

z/OS  
3.2

*Security Server RACF  
Security Administrator's Guide*



**Note**

Before using this information and the product it supports, read the information in [“Notices” on page 743](#).

This edition applies to IBM® z/OS® 3.2 (5655-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2025-09-30

© **Copyright International Business Machines Corporation 1994, 2025.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>Figures.....</b>	<b>xix</b>
<b>Tables.....</b>	<b>xxiii</b>
<b>About this document.....</b>	<b>xxvii</b>
Who should use this document.....	xxvii
How to use this document.....	xxvii
Where to find more information.....	xxvii
RACF courses.....	xxvii
Other sources of information.....	xxviii
Internet sources.....	xxviii
<b>Summary of changes.....</b>	<b>xxix</b>
Summary of changes for z/OS 3.2.....	xxix
Summary of changes for z/OS 3.1.....	xxx
<b>Chapter 1. Introduction.....</b>	<b>1</b>
How RACF meets security needs.....	1
User identification and verification.....	1
Authorization checking.....	2
Logging and reporting.....	2
User accountability.....	3
Flexibility.....	7
RACF transparency.....	7
Implementing multilevel security.....	8
Multilevel security.....	8
Characteristics of a multilevel-secure environment.....	8
Administering security.....	9
Delegating administration tasks.....	9
As of z/VM 7.3, sharing of the RACF database between z/OS and z/VM is not allowed.....	10
Using RACF commands or panels.....	12
RACF group and user structure.....	13
Defining users and groups.....	13
Protecting resources.....	18
Security classification of users and data.....	21
Selecting RACF options.....	21
Using RACF installation exits to customize RACF.....	21
The RACROUTE REQUEST=VERIFY, VERIFYX, AUTH, and DEFINE exits.....	21
The RACROUTE REQUEST=LIST exits.....	22
The RACROUTE REQUEST=FASTAUTH exits.....	22
The RACF command exits.....	22
The RACF password processing exits.....	22
The RACF password authentication exits.....	22
Tools for the security administrator.....	23
Using RACF utilities.....	23
RACF block update command (BLKUPD).....	25
Using the RACF report writer.....	25
Using the data security monitor.....	25
Recording statistics in RACF profiles.....	26

Listing information from RACF profiles.....	26
Searching for RACF profile names.....	28
Using the LIST and SEARCH commands effectively.....	29
<b>Chapter 2. Organizing for RACF implementation.....</b>	<b>33</b>
Ensuring management commitment.....	33
Selecting the security implementation team.....	33
Responsibilities of the implementation team.....	34
Defining security objectives and preparing the implementation plan.....	34
Deciding what to protect.....	35
Protecting existing data.....	35
Protecting new data.....	36
Allowing a warning period.....	37
Establishing ownership structures.....	38
Selecting user IDs and group names.....	38
Establishing your RACF group structure.....	39
Educating the system users.....	41
Summary.....	42
<b>Chapter 3. Defining users.....</b>	<b>45</b>
User profiles.....	45
The base segment in user profiles.....	47
The CICS segment in user profiles.....	48
The CSDATA segment in user profiles.....	49
The DCE segment in user profiles.....	49
The DFP segment in user profiles.....	49
The KERB segment in user profiles.....	50
The LANGUAGE segment in user profiles.....	50
The LNOTES segment in user profiles.....	51
The NDS segment in user profiles.....	51
The NETVIEW segment in user profiles.....	51
The OMVS segment in user profiles.....	52
The OPERPARM segment in user profiles.....	52
The PROXY segment in user profiles.....	54
The TSO segment in user profiles.....	54
The WORKATTR segment in user profiles.....	55
User naming conventions.....	56
Suggestions for defining user IDs.....	56
Migrating existing user IDs to RACF.....	56
Creating new user IDs from scratch.....	56
Creating user IDs for system operators.....	57
Creating user IDs for RRSF users.....	57
Ownership of a RACF user profile.....	57
User attributes.....	57
The SPECIAL attribute.....	57
The AUDITOR attribute.....	58
The ROAUDIT attribute.....	58
The OPERATIONS attribute.....	59
The CLAUTH (class authority) attribute.....	60
The REVOKE attribute.....	61
The GRPACC (group access) attribute.....	61
The ADSP (automatic data set protection) attribute.....	62
The RESTRICTED attribute.....	62
User attributes at the group level.....	62
The scope of authority for the users with group-level attributes.....	63
Suggestions for assigning user attributes.....	66
Verifying user attributes.....	67

Default universal access authority (UACC).....	67
Assigning security categories, levels, and labels to users.....	67
Passwords and password phrases.....	68
Assigning password phrases.....	69
Multi-Factor Authentication for z/OS.....	70
What is Multi-Factor Authentication?.....	70
IBM MFA on z/OS.....	71
Configuring RACF for IBM MFA.....	71
MFA application bypass.....	72
MFA policy.....	72
MFA compound In-Band.....	72
Limiting when a user can access the system.....	73
Time-of-day and day-of-week checking for users and terminals.....	73
Defining protected user IDs.....	74
Restrictions for using protected user IDs with z/VM systems.....	74
Defining restricted user IDs.....	75
Using restricted user IDs for digital certificate users.....	75
Using restricted user IDs for distributed identity users.....	75
Summary of steps for defining users.....	75
Summary of steps for deleting users.....	77
General considerations for user ID delegation.....	79
RACF user ID containment.....	81

## **Chapter 4. Defining groups.....85**

Types of groups.....	85
Administrative groups.....	85
Holding groups.....	85
Data control groups.....	85
Functional groups.....	86
User groups.....	86
Group profiles.....	86
The Base segment in group profiles.....	86
The CSDATA segment in group profiles.....	87
The DFP segment in group profiles.....	87
The OMVS segment in group profiles.....	87
The TME segment in group profiles.....	88
Defining large groups with the UNIVERSAL attribute.....	88
Group naming conventions.....	89
Benefits of using RACF groups.....	89
Group ownership and levels of group authority.....	90
Summary of steps for defining a RACF group.....	93
Summary of steps for deleting groups.....	94

## **Chapter 5. Classifying users and data.....95**

Security classification of users and data.....	95
Effect on RACF authorization checking.....	95
Understanding security levels and security categories.....	96
CATEGORY and SECLEVEL information in profiles.....	97
Converting from LEVEL to SECLEVEL.....	97
Deleting UNKNOWN categories.....	97
Maintaining categories in an RRSF environment.....	97
Understanding security labels.....	98
Comparing security labels.....	98
Considerations related to security labels.....	99
How users specify current security labels.....	100
Listing security labels.....	100
Finding out which security labels a user can use.....	101

Searching by security labels.....	101
Restricting security label changes.....	101
Requiring security labels.....	101
Controlling the write-down privilege.....	101
Planning considerations for security labels.....	103

## **Chapter 6. Specifying RACF options.....105**

Using the SETROPTS command.....	105
SETROPTS options for initial setup.....	106
Allowing mixed-case passwords (PASSWORD option).....	106
Allowing special characters in passwords (PASSWORD option).....	106
Establishing password syntax rules (PASSWORD option).....	108
Setting the maximum and minimum change interval (PASSWORD option).....	108
Extending password and user ID processing (PASSWORD option).....	109
Revoking unused user IDs (INACTIVE option).....	111
Activating list-of-groups checking (GRPLIST option).....	111
Setting the RVARYPW passwords (RVARYPW option).....	112
Restricting the creation of general resource profiles (GENERICOWNER and ENHANCEDGENERICOWNER options).....	113
Activating general resource classes (CLASSACT option).....	114
Activating generic profile checking and generic command processing.....	115
Activating statistics collection (STATISTICS option).....	115
Activating global access checking (GLOBAL option).....	118
RACF-protecting all data sets (PROTECTALL option).....	119
Activating JES2 RACF support.....	120
Preventing access to uncataloged data sets (CATDSNS option).....	120
Activating enhanced generic naming for the DATASET class (EGN option).....	121
Controlling data set modeling (MODEL option).....	121
Bypassing automatic data set protection (NOADSP option).....	122
Displaying and logging real data set names (REALDSN option).....	122
Protecting data sets with single-qualifier names (PREFIX option).....	122
Activating tape data set protection (TAPEDSN option).....	123
Activating tape volume protection (TAPEVOL option).....	123
Establishing a security retention period for tape data sets (RETPD option).....	123
Erasing scratched or released data (ERASE option).....	124
Establishing national language defaults (LANGUAGE option).....	125
SETROPTS options to activate in-storage profile processing.....	125
SETROPTS GENLIST processing.....	126
SETROPTS RACLIST processing.....	127
SETROPTS REFRESH option for special cases.....	129
Refreshing in-storage generic profile lists (GENERIC REFRESH option).....	130
Refreshing global access checking lists (GLOBAL REFRESH option).....	130
Refreshing shared systems (REFRESH option).....	131
SETROPTS options for special purposes.....	131
Protecting undefined terminals (TERMINAL option).....	131
Activating the security classification of users and data.....	131
Establishing the maximum VTAM session interval (SESSIONINTERVAL option).....	132
Activating program control (WHEN(PROGRAM) option).....	132
SETROPTS options related to security labels.....	133
Restricting changes to security labels (SECLABELCONTROL option).....	133
Preventing changes to security labels (MLSTABLE option).....	133
Quiescing RACF activity (MLQUIET option).....	134
Preventing the copying of data to a lower security label (SETROPTS MLS option).....	134
Activating compatibility mode for security labels (COMPATMODE option).....	135
Enforcing multilevel security (MLACTIVE option).....	135
Restricting access to z/OS UNIX files and directories (MLFSOBJ option).....	137
Restricting access to interprocess communication objects (MLIPCOBJ option).....	137

Using name-hiding (MLNAMES option).....	138
Activating security labels by system image (SECLBYSYSTEM option).....	138
SETROPTS options for automatic control of access list authority.....	139
Automatic addition of creator's user ID to access list.....	139
Automatic omission of creator's user ID from access list.....	139
Specifying the encryption method for user passwords.....	139
Using started procedures.....	141
Assigning RACF user IDs to started procedures.....	141
Authorizing access to resources.....	142
Setting up the STARTED class.....	142
Using the started procedures table (ICHRIN03).....	144
Started procedure considerations.....	144

## **Chapter 7. Protecting data sets on DASD and tape..... 147**

Protecting data sets.....	147
Rules for defining data set profiles.....	147
Controlling the creation of new data sets.....	149
Data set profile ownership.....	150
Data set profiles.....	150
Rules for generic data set profile names.....	151
Automatic profile modeling for data sets.....	158
Password-protected data sets.....	160
Protecting GDG data sets.....	161
Protecting data sets that have duplicate names.....	161
Disallowing duplicate names for data set profiles.....	162
Using the PROTECT operand or SECMODEL for non-VSAM data sets.....	162
Protecting multivolume data sets with discrete profiles.....	162
Protecting DASD data sets.....	163
Access authorities for DASD data sets.....	163
Erasing of scratched (deleted) DASD data sets.....	165
Comparison of password and RACF authorization requirements for VSAM.....	165
Protecting catalogs.....	166
Protecting DASD system data sets.....	166
DASD volume authority.....	167
DFSMSdss storage administration.....	168
Protecting data on tape.....	168
Using DFSMSrmm with RACF.....	168
Choosing which tape-related options to use.....	169
Protecting existing data on tape (SETROPTS TAPEDSN in effect).....	171
Protecting new data on tape.....	172
Security levels and security categories for tapes.....	174
Security labels for tapes.....	175
Tape volume profiles that contain a TVTOC.....	175
Predefining tape volume profiles for tape data sets.....	177
RACF security retention period processing (TAPEDSN must be active).....	178
Authorization requirements for tape data sets when both TAPEVOL and TAPEDSN are active.....	179
Authorization requirements for tape data sets when TAPEVOL is inactive and TAPEDSN is active.....	180
Authorization requirements for tape data sets when TAPEVOL is active and TAPEDSN is inactive.....	180
JCL changes.....	180
Installations with DFSMSHsm.....	180
IEC.TAPERING profile in the FACILITY class.....	180
Password-protected tape data sets.....	181
Using the PROTECT parameter for tape data set or tape volume protection.....	181
Multivolume tape data sets.....	181
RACF authorization of bypass label processing (BLP).....	182
Authorization requirements for labels.....	182
Tape data set and tape volume protection with nonstandard labels (NSL).....	182

Tape data set and tape volume protection for nonlabeled (NL) tapes.....	183
<b>Chapter 8. Protecting general resources.....</b>	<b>185</b>
Defining profiles for general resources.....	185
Summary of steps for defining general resource profiles.....	185
Choosing between discrete and generic profiles in general resource classes.....	187
Disallowing generic profile names for general resources.....	188
Choosing among generic profiles, resource group profiles, and RACFVARS profiles.....	188
Rules for generic profile names.....	188
Generic profile checking of general resources.....	190
Generic profile performance.....	192
Granting access authorities.....	192
Conditional access lists for general resource profiles.....	193
Setting up the global access checking table.....	194
How global access checking works.....	195
Candidates for global access checking.....	195
Creating global access checking table entries.....	195
Stopping global access checking for a specific class.....	199
Listing the global access checking table.....	199
Special considerations for global access checking.....	199
Field-level access checking.....	200
Planning for profiles in the FACILITY class.....	211
Delegating help desk functions.....	211
Delegating authority to profiles in the FACILITY class.....	211
Creating resource group profiles.....	212
Adding a resource to a profile.....	213
Deleting a resource from a profile.....	213
Which profiles protect a particular resource?.....	213
Resolving conflicts among grouping profiles.....	214
Considerations for resource group profiles.....	215
Using RACF variables in profile names (RACFVARS class).....	216
Defining RACF variables.....	216
Example of protecting several tape volumes using the RACFVARS class.....	217
Using RACF variables.....	217
How RACF uses the RACFVARS member list.....	218
Using RACFVARS with mixed-case classes.....	220
Controlling VTAM LU 6.2 bind.....	221
Protecting applications.....	223
Protecting DFP-managed temporary data sets.....	224
Protecting file services provided by LFS/ESA.....	224
Protecting terminals.....	225
Creating profiles in the TERMINAL and GTERMINL classes.....	225
Controlling the use of undefined terminals.....	226
Limiting specific groups of users to specific terminals.....	227
Limiting the times that a terminal can be used.....	227
Using security labels to control terminals.....	227
Using the TSO LOGON command with the RECONNECT operand.....	227
Protecting consoles.....	228
Using security labels to control consoles.....	229
Protecting the vector facility.....	229
Controlling access to program dumps.....	230
Using RACF to control access to program dumps.....	230
Using non-RACF methods to control access to program dumps.....	232
Controlling the allocation of devices.....	232
Protecting LLA-managed data sets.....	234
Controlling data lookaside facility (DLF) objects (Hiperbatch).....	235
Using RACROUTE REQUEST=LIST,GLOBAL=YES support.....	237



The RACGLIST class.....	238
Administering the use of operator commands.....	239
Authorizing the use of operator commands.....	239
Command authorization in an MCS sysplex.....	240
Controlling the use of operator commands.....	240
Establishing security for the RACF parameter library.....	245
Controlling message traffic.....	245
Controlling the opening of VTAM ACBs.....	246
RACF and PSF (Print Services Facility).....	247
Auditing when users receive message traffic.....	248
RACF and APPC.....	248
User verification during APPC transactions.....	248
Protection of APPC/MVS transaction programs (TPs).....	249
LU security capabilities.....	250
Origin LU authorization.....	250
Protection of APPC server IDs (APPCSERV).....	250
RACF and CICS.....	250
RACF and Db2.....	251
RACF and IMS.....	251
RACF and ICSF.....	251
RACF and z/OS UNIX.....	251
RACF support for NDS and Lotus Notes for z/OS.....	251
Administering application user identities.....	251
System considerations.....	252
Authorizing applications to use identity mapping.....	254
Considerations for application user names.....	255
Storing encryption keys using the KEYSMSTR class.....	255
Steps for storing a key in a KEYSMSTR profile.....	256
Defining delegated resources.....	257
Steps for authorizing daemons to use delegated resources.....	257
<b>Chapter 9. Limiting ALTER access authority in discrete profiles.....</b>	<b>259</b>
Summary of ALTER access.....	259
Restricting ALTER access.....	260
Logging the use of ALTER access for profile management.....	261
<b>Chapter 10. Administering the dynamic class descriptor table (CDT).....</b>	<b>263</b>
Overview of the class descriptor table.....	263
Restrictions for applications and vendor products.....	263
Using the dynamic CDT.....	264
Profiles in the CDT class.....	264
Adding a dynamic class with a unique POSIT value.....	265
Steps for adding a dynamic class with a unique POSIT value.....	266
Adding a dynamic class that shares a POSIT value.....	267
Processing options that are controlled by a shared POSIT value.....	267
Rules about disallowing generics when sharing a POSIT value.....	268
Steps for adding a dynamic class with a shared POSIT value.....	268
Changing a POSIT value for a dynamic class.....	269
Steps for changing a POSIT value of an existing dynamic class.....	269
Guidelines for changing dynamic CDT entries.....	270
Defining a dynamic class with generics disallowed.....	272
Steps for changing a dynamic class to disallow generic profiles.....	272
Deleting a class from the dynamic CDT.....	273
Steps for deleting a dynamic CDT class.....	273
Disabling the dynamic CDT.....	275
Re-enabling a previously defined dynamic class.....	276
Steps to re-enable a previously defined dynamic class.....	276

Migrating to the dynamic CDT.....	276
Sysplex considerations for the dynamic CDT.....	278
Shared system considerations for the dynamic CDT.....	278
RRSF considerations for the dynamic CDT.....	279
<b>Chapter 11. Using PassTickets.....</b>	<b>281</b>
The RACF PassTicket.....	281
Activating the PTKTDATA class.....	282
Defining profiles in the PTKTDATA class.....	282
Determining PTKTDATA profile names.....	283
Protecting PassTicket keys.....	286
Storing legacy PassTicket Keys Masked in RACF.....	286
Storing PassTicket keys encrypted in ICSF.....	287
Converting legacy PassTicket masked keys to encrypted keys.....	288
Authorization requirements for managing PassTicket keys.....	288
Examples of defining PTKTDATA class profiles.....	289
When the profile definitions are complete.....	289
How RACF processes the PassTicket.....	289
Bypassing PassTicket replay protection.....	290
Bypassing legacy PassTicket replay protection.....	291
Bypassing enhanced PassTicket replay protection.....	291
Enabling the use of PassTickets.....	291
Verifying the PassTicket environment.....	292
<b>Migrating from legacy PassTickets to enhanced PassTickets.....</b>	<b>292</b>
Preventing errors.....	293
Auditing the use of PassTickets.....	294
Allowing password changes when authenticating with a PassTicket.....	295
<b>Chapter 12. Detecting ACEE modifications.....</b>	<b>297</b>
<b>Chapter 13. Protecting programs.....</b>	<b>301</b>
Overview of protecting programs.....	301
Program security modes.....	302
Simple program protection in BASIC or ENHANCED mode.....	303
Program control by SMFID in BASIC or ENHANCED mode.....	305
Maintaining a clean environment in BASIC or ENHANCED mode.....	306
More complex controls: Using EXECUTE access for programs or libraries (BASIC mode).....	307
Migrating from BASIC to ENHANCED program security mode.....	308
Protecting program libraries.....	309
Program access to data sets (PADS) in BASIC mode.....	310
Choosing between the PADCHK and NOPADCHK operands.....	314
Program access to SERVAUTH resources in BASIC or ENHANCED mode.....	314
ENHANCED program security mode.....	315
Program access to data sets (PADS) in ENHANCED mode.....	315
Using EXECUTE access for programs and libraries in ENHANCED mode.....	316
When to use MAIN or BASIC.....	316
Defining programs as MAIN or BASIC.....	317
How protection works for programs and PADS.....	318
How program control works.....	318
Informational messages for program control.....	319
Authorization checking for access control to load modules.....	319
Authorization checking for access control to data sets.....	320
Processing for execute-controlled libraries.....	321
Examples of controlling programs and using PADS.....	322
Examples of defining load modules as controlled programs.....	322
Examples of setting up program access to data sets.....	323
Example of setting up an execute-controlled library.....	324

Example of setting up program control by system ID.....	325
<b>Chapter 14. Program signing and verification.....</b>	<b>327</b>
Overview of program signing and verification.....	327
Terms to know.....	327
Related information.....	327
Task roadmap for program signing and signature verification.....	328
Enabling a user to sign a program.....	328
Overview of enabling a user to sign a program.....	328
Steps for enabling a user to sign a program using RACF code-signing certificates.....	331
Steps for enabling a user to sign a program using external code-signing certificates.....	333
Enabling RACF to verify signed programs.....	335
Overview of enabling RACF to verify signed programs.....	335
Steps for discovering if signed programs currently execute on your systems (optional).....	338
Steps for preparing RACF to verify signed programs (one-time setup).....	340
Steps for verifying a signed program.....	342
IPL data signing for Validated Boot for z/OS .....	344
Overview of enabling your system for signed IPL data.....	344
Certificate requirements for signing IPL data.....	344
Defining the IRR.PROGRAM.V2.SIGNING profile.....	345
Enabling IPL data signing for Validated Boot for z/OS.....	347
<b>Chapter 15. Operating considerations.....</b>	<b>355</b>
Coordinating profile updates.....	355
RACF commands for flushing a VLF cache.....	356
Getting started with RACF (after first installing RACF).....	356
Logging on as IBMUSER and checking initial conditions.....	357
Defining administrator user IDs for your own use.....	358
Defining at least one user ID to be used for emergencies only.....	358
Logging on as RACFADM, checking groups and users, and revoking IBMUSER.....	358
Defining the groups needed for the first users.....	358
Defining a system-wide auditor.....	359
Defining a system-wide read-only auditor.....	359
Defining users and groups.....	359
Defining group administrators, group auditors, and data managers.....	359
Protecting system data sets.....	360
Setting RACF options.....	361
Using the data security monitor (DSMON).....	361
JCL parameters related to RACF.....	364
Restarting jobs.....	365
Bypassing password protection.....	365
Controlling access to RACF passwords.....	365
Authorizing only RACF-defined users to access RACF-protected resources.....	366
Using the TSO or ISPF editor.....	367
Service by IBM personnel.....	367
Failsoft processing.....	367
Failsoft processing with tape data sets.....	368
Considerations for RACF databases.....	368
Backup RACF database.....	368
Multiple data set support.....	368
Protecting the RACF database.....	368
Encrypting a RACF VSAM data set.....	369
Using RACF data sharing.....	370
Sharing data without sharing a RACF database.....	370
Number of resident data blocks.....	370
<b>Chapter 16. Working with the RACF database.....</b>	<b>371</b>

Using the RACF database unload utility (IRRDBU00).....	371
Diagnosis.....	371
Performance considerations.....	371
Operational considerations.....	372
Running the database unload utility.....	373
Allowable parameters.....	374
Using the database unload utility output effectively.....	375
Using the RACF remove ID (IRRRID00) utility.....	391
IRRRID00 job control statements.....	394
IRRRID00 return codes.....	396
Finding residual IDs.....	396
Creating commands to remove IDs.....	397
Using IRRRID00 output.....	398
Processing profiles and resources.....	401
What IRRRID00 verifies.....	401
Database objects that are not processed.....	402
Processing a hierarchy of groups.....	402
Processing global profiles.....	403
Processing general resource profiles.....	403
Processing MEMBER data.....	403
Processing universal groups.....	403
IRRRID00 and Tivoli.....	403
Time required to run IRRRID00.....	404
<b>Chapter 17. The RACF remote sharing facility (RRSF).....</b>	<b>405</b>
The RRSF network.....	405
RRSF nodes.....	406
Establishing user ID associations in the RRSF network.....	407
Types of user ID associations.....	407
Password synchronization.....	408
User ID associations.....	409
Defining user ID associations.....	409
Approving user ID associations.....	410
Deleting user ID associations.....	411
Listing user ID associations.....	411
Command direction.....	411
Commands that are not eligible for command direction.....	412
Directing commands using the AT option.....	412
Directing commands using the ONLYAT option.....	415
Order considerations for directed commands and application updates.....	416
Directing commands to incompatible systems.....	417
Automatic direction.....	417
Preparing to use automatic direction.....	418
Output processing.....	421
Interactions among automatic direction functions and password synchronization.....	425
Using automatic direction of commands.....	427
Using automatic direction of application updates.....	430
Using automatic password direction.....	433
Synchronizing database profiles.....	434
Controlling the use of remote sharing functions.....	434
Controlling access to the RACLINK command.....	435
Controlling password synchronization.....	435
Controlling the use of the AT operand.....	436
Controlling the use of the ONLYAT operand.....	437
Controlling automatic direction.....	437
Establishing RACF security for RRSF TCP/IP connections.....	441
Task roadmap for establishing RACF security for RRSF TCP/IP connections.....	442

Administer profiles in the SERVAUTH class to enable RRSF to use TCP/IP node connections.....	442
Implementing an RRSF trust policy.....	444
<b>Chapter 18. Providing security for JES.....</b>	<b>453</b>
Planning for security.....	453
How JES and RACF work together.....	453
Defining JES as a RACF started procedure.....	453
Forcing batch users to identify themselves to RACF.....	454
Support for execution batch monitor (XBM).....	454
Defining and grouping operators.....	454
JES user ID early verification.....	455
User ID propagation when jobs are submitted.....	455
Allowing surrogate job submission.....	455
Controlling user ID propagation in a local environment.....	457
Using protected user IDs for batch jobs.....	458
Propagating protected user IDs.....	458
Using protected user IDs for surrogate job submission.....	458
Where NJE jobs are verified.....	458
How SYSOUT requests are verified.....	459
Security labels for JES resources.....	459
Controlling access to data sets JES uses.....	460
Controlling input to your system.....	460
How RACF validates users.....	460
Controlling the use of job names.....	461
Authorizing the use of input sources.....	467
Authorizing network jobs and SYSOUT (NJE).....	468
Authorizing inbound work.....	468
Authorizing outbound work.....	483
Controlling access to spool data.....	483
Protecting data sets on spools.....	483
Defining profiles for SYSIN and SYSOUT data sets.....	484
Letting users create their own JESSPOOL profiles.....	486
Protecting JESNEWS.....	487
Protecting trace data sets (JES2 only).....	488
Protecting SYSLOG.....	488
Spool offload considerations (JES2 only).....	488
Dumping jobs.....	489
Restoring jobs.....	489
Controlling access to key labels for spool data sets.....	489
Authorizing console access.....	491
MCS consoles.....	491
Remote workstations (RJP/RJE consoles).....	491
Controlling where output can be processed.....	493
Authorizing the use of your installation's printers.....	494
Authorizing the use of operator commands.....	495
Commands from RJE work stations.....	495
Commands from NJE nodes.....	495
Who authorizes commands when RACF is active.....	496
<b>Chapter 19. RACF and Storage Management Subsystem (SMS).....</b>	<b>497</b>
Overview of RACF and SMS.....	497
RACF general resource classes for protecting SMS classes.....	497
Controlling the use of SMS classes.....	497
Refreshing profiles for SETROPTS RACLIST processing for MGMTCLAS and STORCLAS.....	498
DFP segment in RACF profiles.....	499
DFP segment in user and group profiles.....	499
DFP segment in data set profiles.....	500

How RACF uses the information in the DFP segments.....	501
Controlling access to the DFP segment.....	501
Controlling the use of other SMS resources.....	504
<b>Chapter 20. RACF and TSO/E.....</b>	<b>507</b>
TSO/E administration considerations.....	507
Protecting TSO resources.....	507
Authorization checking for protected TSO resources.....	510
Field-level access checking for TSO.....	510
Controlling the use of the TSO SEND command.....	510
Restricting spool access by TSO users.....	511
TSO commands that relate to RACF.....	511
Using TSO when RACF is deactivated.....	512
<b>Chapter 21. RACF and z/OS UNIX.....</b>	<b>513</b>
Defining group identifiers (GIDs).....	513
Defining user identifiers (UIDs).....	514
Listing UIDs and GIDs.....	514
Superuser authority.....	515
Setting z/OS UNIX user limits.....	515
Protected user IDs.....	516
Controlling the use of shared UNIX identities.....	516
Sharing IDs.....	516
Defining the SHARED.IDS profile in the UNIXPRIV class.....	517
Using the SHARED operand.....	517
Enabling automatic assignment of unique UNIX identities.....	518
Automatically assigning unique IDs using RACF commands.....	518
Automatically assigning unique IDs through UNIX services.....	520
RRSF considerations for automatic ID assignment.....	522
z/OS UNIX performance considerations.....	524
Converting to stage 3 of application identity mapping.....	524
Using the UNIXMAP class and Virtual Lookaside Facility (VLF).....	524
Using UNIXPRIV class profiles to manage z/OS UNIX privileges.....	527
Example of authorizing superuser privileges.....	527
Allowing z/OS UNIX users to change file ownerships.....	527
Allowing z/OS UNIX users to read or search directories.....	528
Configuring the group owner for new UNIX files.....	529
Protecting file system resources.....	530
Administering ACLs.....	530
Restricting access to a zFS file system.....	532
Steps for restricting access to a zFS file system.....	533
Restricting execute access in a zFS or TFS file system.....	534
Steps for restricting execute access in a zFS or TFS file system.....	534
z/OS UNIX application considerations.....	535
Threads and security.....	535
Application services and security.....	536
Restrictions of RACF client ACEE support.....	537
Auditing z/OS UNIX security events.....	537
<b>Chapter 22. RACF and digital certificates.....</b>	<b>539</b>
Overview of digital certificates.....	539
Public and private keys.....	539
X.509 certificates.....	539
Certificate hierarchies.....	540
Certificate formats.....	541
Using certificates with z/OS client/server applications.....	542
Enabling client login using certificates.....	545

Using RACF to manage digital certificates.....	546
Size considerations for public and private keys.....	547
Using the RACDCERT command to administer certificates.....	548
Controlling the use of the RACDCERT command.....	549
Examples of adding digital certificate information.....	552
Examples of listing digital certificate information.....	552
Examples of listing digital certificate chain information.....	556
Examples of checking digital certificate information.....	559
Examples of altering digital certificate information.....	564
Examples of deleting digital certificates.....	564
DIGTCERT general resource profiles.....	565
DIGTCERT profile names.....	565
Ownership of DIGTCERT profiles.....	566
RACLISTing the DIGTCERT class.....	566
RACF and key rings.....	567
DIGTRING general resource profiles.....	568
Sharing a private key in a key ring.....	568
Using a virtual key ring.....	568
RACF and z/OS PKCS #11 tokens.....	569
Creating and populating PKCS #11 tokens.....	569
Certificate name filtering.....	571
Interpreting the X.500 directory information tree.....	571
Creating certificate name filters.....	572
Types of certificate name filters.....	574
How RACF processes certificate name filters.....	577
Using an existing certificate as a model.....	577
Excluding a certificate by using the NOTRUST option.....	578
Mapping multiple user IDs using additional criteria.....	578
Automatic registration of digital certificates.....	582
ICSF considerations for keys in the PKA key data set (PKDS).....	582
Using a PCI cryptographic coprocessor to generate private keys.....	582
Migrating an ICSF private key in the PKDS from one system to another.....	582
The irrcerta, irrmulti, and irrsitec user IDs.....	584
Renewing an expiring certificate.....	584
Renewing a certificate with the same private key.....	585
Renewing (rekeying) a certificate with a new private key .....	586
Supplied digital certificates.....	589
Steps to begin using a supplied CA certificate.....	589
Implementation scenarios.....	590
Scenario 1: Secure server with a certificate signed by a certificate authority.....	590
Scenario 2: Secure server with a locally signed certificate.....	592
Scenario 3: Migrating an ikeyman or gskkyman certificate.....	593
Scenario 4: Secure server-to-server session enablement.....	594
Scenario 5: Creating client browser certificates with a locally signed certificate.....	596
Scenario 6a: Enabling secure outbound FTP using a shared virtual key ring.....	597
Scenario 6b: Enabling secure outbound FTP using a shared real key ring.....	598
Scenario 7: Sharing one certificate among multiple servers.....	599
Scenario 8: Using the IBM Encryption Facility for z/OS.....	601
<b>Chapter 23. Controlling applications that invoke callable services.....</b>	<b>603</b>
Authorizing applications.....	603
Defining applications as RACF users.....	603
Defining resources that control callable services.....	603
Activating your authorizations.....	603
initACEE (IRRSIA00) callable service.....	604
Registering user certificates.....	604
Deregistering user certificates.....	604

Replacing certificate-authority certificates.....	604
Using a hostIdMappings extension.....	605
R_admin (IRRSEQ00) callable service.....	606
R_auditx (IRRSAX00) callable service.....	606
R_cacheserv (IRRSCH00) callable service.....	606
R_datalib (IRRSDL00 or IRRSDL64) callable service.....	606
Extracting private keys.....	607
Managing certificate serial numbers.....	607
R_dcekey (IRRSDK00) callable service.....	607
R_GetInfo (IRRSGIO0) callable service.....	608
R_dceruid (IRRSUD00) callable service.....	608
R_PKIServ (IRRSPX00) callable service.....	608
Authorizing end-user functions.....	608
Authorizing administrative functions.....	611
R_proxyserv (IRRSFY00) callable service.....	614
R_ticketserv (IRRSFK00) callable service.....	614
Permitting access to the IRR.RTICKETSERV resource.....	615
<b>Chapter 24. RACF and the z/OS LDAP server.....</b>	<b>617</b>
Defining an LDAPBIND class profile.....	617
LDAP event notification.....	618
LDAP change log entries.....	619
LDAP notification occurs in real-time only.....	620
RRSF considerations for applications that exploit enveloping.....	620
Activating LDAP change notification.....	620
Disabling LDAP change notification.....	621
<b>Chapter 25. Password and password phrase enveloping.....</b>	<b>623</b>
Overview of enveloping.....	623
Resources that control enveloping.....	623
Signing hash algorithm and encryption strength used to create the envelope.....	624
The IRR.PWENV.KEYRING key ring.....	625
Controlling envelope retrieval.....	625
The NOTIFY.LDAP.USER resource.....	626
Setting up enveloping.....	626
Preparing the address space of the RACF subsystem.....	626
Generating a local CA certificate using RACF as the CA.....	627
Generating an X.509 V3 certificate for the RACF address space.....	627
Generating an X.509 V3 certificate for the envelope recipient.....	628
Copying the certificates to the host system (if generated elsewhere).....	630
Exporting RACF's certificate to the recipient key database.....	631
Authorizing the envelope recipient.....	631
Activating enveloping.....	632
Disabling enveloping.....	633
Steps for disabling enveloping and deleting existing envelopes.....	634
Planning considerations for heterogeneous password synchronization.....	635
<b>Chapter 26. Defining and using custom fields.....</b>	<b>637</b>
Overview of custom fields.....	637
Task roadmap for defining and using custom fields.....	638
Defining a custom field and its field attributes.....	638
Profiles in the CFIELD class.....	638
Steps for defining a custom field and its attributes.....	640
Activating a custom field.....	642
Steps for activating a custom field.....	642
Adding data to a custom field.....	643
Steps for adding data to a custom field.....	643



Authorizing users to define custom fields.....	644
Steps for authorizing users to define custom fields.....	645
Authorizing users to update data in a custom field.....	645
Authorizing users for the ISPF panels to update custom field data.....	646
Steps for authorizing users to update data in a custom field.....	646
Changing attributes of an existing custom field.....	647
When you need to change the data type.....	648
When you need to change the MAXLENGTH of a numeric field.....	649
Removing a custom field.....	650
Steps for removing a custom field.....	650
Common errors when defining and using custom fields.....	651
Errors defining a custom field.....	651
Errors adding data to a custom field.....	651
RRSF considerations for custom fields.....	653
<b>Chapter 27. Authorizing help desk functions.....</b>	<b>655</b>
Delegating the authority to list user information.....	655
Delegating the authority to list user information in any user profile.....	655
Delegating the authority to list user information in only selected user profiles.....	656
Delegating the authority to list user information by owner.....	657
Delegating the authority to list user information by group tree.....	658
Excluding selected user profiles.....	659
Delegating the authority to reset passwords and password phrases.....	660
Levels of authority.....	661
Delegating the authority to reset the password for any user.....	662
Delegating the authority to reset passwords for only selected users.....	663
Delegating the authority to reset passwords by owner.....	663
Delegating the authority to reset passwords by group tree.....	664
Excluding selected users.....	666
Delegating both by owner and by group tree.....	667
Examples of delegating help desk authorities.....	668
Delegating help desk authorities by owner.....	668
Delegating help desk authorities by group tree.....	669
Delegating help desk authorities for all users, excluding selected users.....	669
<b>Chapter 28. Distributed identity filters.....</b>	<b>671</b>
Overview of distributed identity filters.....	671
What is a distributed identity filter?.....	671
Applications that support distributed identity filters.....	671
Overview of the RACMAP command.....	671
Profiles in the IDIDMAP class.....	672
RACMAP command updates to user profiles.....	672
DELUSER processing with distributed identity filters.....	673
IRRRID00 considerations for distributed identity filters.....	673
Details about specifying user and registry names.....	673
Restrictions for UTF-8 data values.....	677
Defining a filter for a non-LDAP user name.....	677
Steps for defining a filter for a non-LDAP user name.....	677
Defining a filter for an X.500 user identity.....	678
Steps for defining a filter for a full X.500 DN.....	679
Steps for defining a filter using selected RDNs.....	680
Deleting a distributed identity filter.....	681
Steps for deleting a distributed identity filter.....	682
<b>Appendix A. Supplied RACF resource classes.....</b>	<b>683</b>
Supplied resource classes for z/OS systems.....	683

<b>Appendix B. Summary of RACF commands and authorities.....</b>	<b>693</b>
Summary of commands and their functions.....	693
Summary of authorities and commands.....	699
The SPECIAL or group-SPECIAL attribute.....	699
The AUDITOR or group-AUDITOR attribute.....	700
The ROAUDIT attribute.....	701
The OPERATIONS or group-OPERATIONS attribute.....	701
The CLAUTH attribute.....	701
Group authority.....	702
Access authority.....	703
Profile ownership authority.....	704
Other authorities.....	706
<b>Appendix C. Listings of RACF supplied certificates.....</b>	<b>709</b>
<b>Appendix D. Security for system data sets.....</b>	<b>711</b>
<b>Appendix E. Debugging problems in the RACF database.....</b>	<b>715</b>
Checklist: Resolving problems when access is denied unexpectedly.....	715
Checklist: Resolving problems when access is allowed incorrectly.....	716
When changes to data set profiles take effect.....	717
Authorization checking for RACF-protected resources.....	718
When authorization checking takes place and why.....	718
Authorizing access to RACF-protected resources.....	719
Pictorial view of RACF authorization checking.....	723
Authorizing access to z/OS UNIX files and directories.....	728
Authorizing access to RACF-protected terminals.....	730
Authorizing access to consoles, JES input devices, APPC partner LUs, or IP addresses.....	731
Authorization checking for RACROUTE REQUEST=FASTAUTH requests.....	732
Authorizing access to RACF-protected applications.....	733
Security label authorization checking.....	733
Relationships among the SECLABEL class, SETROPTS MLS(FAILURES), SETROPTS	
MLACTIVE(FAILURES) and SETROPTS MLQUIET.....	737
Problems with user ID authentication.....	738
When logon or job initialization processing takes place and why.....	738
Logon/job initialization processing.....	739
<b>Appendix F. Accessibility.....</b>	<b>741</b>
<b>Notices.....</b>	<b>743</b>
Terms and conditions for product documentation.....	744
IBM Online Privacy Statement.....	745
Policy for unsupported hardware.....	745
Minimum supported hardware.....	745
RSA Secure code.....	746
Trademarks.....	746
<b>Glossary.....</b>	<b>747</b>
<b>Index.....</b>	<b>773</b>

---

# Figures

1. RACF authorization checking.....	2
2. Sample ISPF panel for RACF.....	13
3. Scope of control of an attribute assigned at the group level.....	15
4. User and group relationships.....	40
5. Group-level authority structure.....	65
6. Scope of authority for a group-SPECIAL user.....	66
7. Delegating authority (user profiles).....	80
8. Example of two network LU partners.....	223
9. Reports produced by DSMON.....	362
10. Member UGRP: Users with extraordinary group authorities: report format statements.....	376
11. Member UGRPCNTL: Users with extraordinary group authorities: record selection statements.....	376
12. Report of all users with extraordinary group authorities.....	377
13. Customized record selection criteria.....	380
14. Customized report format.....	381
15. Customized report JCL.....	381
16. Sample SQL utility statements: Defining a table space.....	383
17. Sample SQL utility statements: Creating a table.....	383
18. Sample SQL utility statements: Creating indexes.....	384
19. Db2 utility statements required to load the tables.....	384
20. Db2 utility statements required to delete the group records.....	385
21. Sample SQL to process revoke and resume dates.....	389
22. A sample SQL query.....	390
23. A sample QMF form.....	391

24. A sample report.....	391
25. Using the remove ID utility.....	392
26. Searching for all residual references.....	395
27. Searching for specific references.....	395
28. Specifying a replacement ID.....	396
29. Running IRRRID00 with an empty SYSIN: Sample input.....	397
30. Running IRRRID00 with an empty SYSIN: Sample output.....	397
31. Running IRRRID00 with data in SYSIN: Sample input.....	398
32. Running IRRRID00 with data in SYSIN: Sample output.....	398
33. Sample output from the IRRRID00 utility.....	400
34. Running IRRRID00 CLIST using TMP: Sample JCL statements.....	400
35. An RRSF network.....	406
36. Captured output from a password synchronization request.....	409
37. RACLINK ID(userid) LIST(*.*) output.....	411
38. Captured output from a directed LISTGRP command.....	415
39. Captured output from a directed ADDSD command.....	415
40. Which NODES profiles are used?.....	472
41. Example: Simple NJE user translation.....	479
42. Example: Simple NJE user translation using &SUSER.....	479
43. Example: Trusted, semitrusted, and untrusted nodes.....	480
44. Example of a simple certificate hierarchy.....	540
45. A high-level view of a secure z/OS handshake using a public key network protocol.....	543
46. Controlling access to RACDCERT functions under the FACILITY class.....	551
47. Output from the RACDCERT LIST command.....	553
48. Output from the RACDCERT LISTRING command.....	554

49. Output from the RACDCERT LIST command with LABEL.....	554
50. Output from the RLIST DIGTCERT command.....	555
51. Output from the SEARCH CLASS(DIGTCERT) command.....	556
52. Example of an X.500 directory information tree.....	572
53. Sample RACDCERT MAP command for creating an issuer's name filter.....	573
54. Sample output from the LISTMAP command for an issuer's name filter.....	574
55. Sample RACDCERT MAP commands for creating subject's name filters.....	574
56. Sample RACDCERT MAP command for creating a subject's and issuer's name filter.....	576
57. Sample RACDCERT MAP commands using a model certificate.....	578
58. Sample RACDCERT MAP commands not using a model certificate.....	578
59. Sample RACDCERT MAP command using the NOTRUST option.....	578
60. Sample RACDCERT MAP and RDEFINE commands for mapping multiple user IDs.....	579
61. Sample output from the LISTMAP command for a MULTIID filter.....	580
62. Sample RACDCERT MAP and RDEFINE commands using multiple criteria.....	581
63. Sample group and user structure for delegating help desk authorities.....	668
64. Process flow of callers of RACF for RACROUTE REQUEST=AUTH requests.....	724
65. Process flow of SAF router for RACROUTE REQUEST=AUTH requests.....	724
66. Process flow of RACF router.....	725
67. Process flow of RACF authorization checking, part 1 of 3.....	726
68. Process flow of RACF authorization checking, part 2 of 3.....	727
69. Process flow of RACF authorization checking, part 3 of 3.....	728



---

# Tables

1. User attributes.....	15
2. Commands to list profile contents.....	26
3. Command to search for profile names.....	29
4. Participants of the security implementation team.....	34
5. Checklist for implementation team activities.....	43
6. Scope of authority for user attributes at the group level.....	64
7. Group authorities.....	91
8. Sample profile names for STARTED class resources.....	143
9. Sample data set profile names in order from most specific to least specific (EGN off).....	153
10. Sample data set profile names in order from most specific to least specific (EGN on).....	154
11. Protecting GDG data sets using generic profiles.....	161
12. Access authorities for DASD data sets.....	164
13. RACF commands used with general resource profiles.....	185
14. Choosing among generic profiles, resource group profiles, and RACFVARS profiles.....	188
15. Sample general resource profile names in order from most specific to least specific.....	190
16. ALTER, NONE, and CONTROL, UPDATE, and READ access authorities for general resources.....	193
17. Comparison of GRPACC attribute with &RACGPID.** entry in global access checking table.....	198
18. Fields in RACF segments that correspond to RACF command operands.....	205
19. Delegating authority in the FACILITY class.....	212
20. Rules for merging conflicting profiles.....	214
21. RACF classes used to authorize operator commands.....	239
22. RACF operator command profiles: Naming conventions.....	242
23. RACF TSO commands entered as operator commands: Naming conventions.....	244

24. KEYSMSTR class profiles.....	255
25. Processing options controlled simultaneously for classes sharing a POSIT value.....	268
26. ICHERCDE macro operands and the corresponding operands for the RDEFINE and RALTER commands.....	277
27. Scenarios for signing IPL data.....	347
28. Correlation of record type, record name, and Db2 table name.....	385
29. Return codes for the remove ID utility (IRRRID00).....	396
30. RRSFDATA resources to control propagation of certificate information.....	432
31. Automatic command direction: Resource names.....	437
32. Resource names in the JESJOBS class for the Job Modify SSI.....	465
33. NODES class operands and the UACC meaning for inbound jobs.....	473
34. NODES class operands, UACC, and SYSOUT ownership when node is not defined to &RACLNDE.....	477
35. TSO command usage when RACF protection is enabled.....	511
36. The UNIXMAP class and VLF: Effects on performance for installations that have not reached stage 3 of application identity mapping.....	524
37. Subject's and issuer's distinguished names.....	571
38. Summary of access authorities required for PKI Services requests.....	610
39. LDAP event notification of RACF profile changes.....	618
40. Authorities you can delegate based on the access level to the IRR.PASSWORD.RESET, IRR.PWRESET.OWNER, IRR.PWRESET.TREE, and IRR.PWRESET.EXCLUDE resources.....	661
41. Resource classes for z/OS systems.....	683
42. Functions of RACF commands.....	693
43. Commands and operands you can issue if you have the SPECIAL or group-SPECIAL attribute.....	699
44. Commands and operands you can issue if you have the AUDITOR or group-AUDITOR attribute.....	700
45. Commands and operands you can issue if you have the ROAUDIT attribute.....	701
46. Commands and operands you can issue if you have the OPERATIONS or group-OPERATIONS attribute.....	701
47. Commands and operands you can issue if you have the CLAUTH attribute.....	701



48. Commands and operands you can issue if you have a group authority.....	702
49. Commands and operands you can issue if you have an access authority.....	703
50. Commands and operands you can issue if you own a profile.....	704
51. Commands and operands you can issue for miscellaneous reasons.....	706
52. UACC values for system data sets.....	711
53. Required relationship between security levels for each MAC checking type.....	735
54. Security label authorization checking when SECLABEL class is active and either SETROPTS MLS(FAILURES) or MLS(WARNING) is in effect.....	735
55. Security label authorization checking when SECLABEL class is active and either SETROPTS NOMLS is in effect or the user is in "writedown" mode.....	736
56. Effects of MLACTIVE settings on security label authorization.....	737
57. Relationships among the SECLABEL class, SETROPTS MLS(FAILURES), SETROPTS MLACTIVE(FAILURES), and SETROPTS MLQUIET.....	737
58. Resource classes checked for logon and job initialization requests.....	740



## About this document

---

This document supports z/OS (5655-ZOS) and contains information about Resource Access Control Facility (RACF), which is part of z/OS Security Server. This document provides information to help the security administrator plan for and administer the RACF® component of z/OS Security Server.

## Who should use this document

---

Security administrators, group administrators, and other administrators who are responsible for system data security and integrity on a z/OS system should use this document for such tasks as:

- Planning how to use RACF to increase the security of the system
- Deciding which resources to protect
- Performing administration tasks
- Coordinating with administrators of other products

Readers should be familiar with RACF concepts and terminology. The readers of this document should also be familiar with z/OS systems.

RACF overview information can be obtained from the [RACF home page \(www.ibm.com/products/resource-access-control-facility/resources\)](http://www.ibm.com/products/resource-access-control-facility/resources).

## How to use this document

---

Much of this document describes how to protect resources, such as data sets, terminals, and others. In general, you first need to define users to RACF and set some RACF options. Then, depending on your security plan, you select classes of resources to protect and create resource profiles for them.

If you are reading this document for the first time, consider reading the following parts first:

- [Chapter 1, “Introduction,” on page 1](#)
- [Chapter 2, “Organizing for RACF implementation,” on page 33](#)
- [Chapter 3, “Defining users,” on page 45](#)
- [Chapter 4, “Defining groups,” on page 85](#)
- [“Defining profiles for general resources” on page 185](#)
- [“Setting up the global access checking table” on page 194](#)
- [“Getting started with RACF \(after first installing RACF\)” on page 356](#)
- Appropriate portions of [Chapter 6, “Specifying RACF options,” on page 105](#)

## Where to find more information

---

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

To find the complete z/OS library, including the z/OS Documentation, see the [z/OS Internet library \(www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary\)](http://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary).

To find educational material, see the [IBM Education home page \(www.ibm.com/training\)](http://www.ibm.com/training).

## RACF courses

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

**ES191**

*Basics of z/OS RACF Administration*

**BE870**

*Effective RACF Administration*

**ES885**

*Exploiting the Advanced Features of RACF*

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

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

## Other sources of information

---

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

### Internet sources

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

- z/OS Internet library ([www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary](http://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary))
- IBM Redbooks ([www.ibm.com/redbooks](http://www.ibm.com/redbooks))
- Enterprise security ([www.ibm.com/systems/z/solutions/enterprise-security.html](http://www.ibm.com/systems/z/solutions/enterprise-security.html))
- RACF home page ([www.ibm.com/products/resource-access-control-facility/resources](http://www.ibm.com/products/resource-access-control-facility/resources))
- RACF download page ([github.com/IBM/IBM-Z-zOS/tree/master/zOS-RACF/Downloads](https://github.com/IBM/IBM-Z-zOS/tree/master/zOS-RACF/Downloads))

## Summary of changes

---

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.

**Note:** IBM z/OS policy for the integration of service information into the z/OS product documentation library is documented on the z/OS Internet Library under [IBM z/OS Product Documentation Update Policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy) ([www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy](http://www.ibm.com/docs/en/zos/latest?topic=zos-product-documentation-update-policy)).

## Summary of changes for z/OS 3.2

---

The following content is new, changed, or no longer included in z/OS 3.2.

### New

The following content is new.

#### September 2025 release

- **RACF user ID containment:** RACF provides the ability for the administrator to contain a user ID, stopping them from accessing RACF resources from active sessions. This function allows the administrator to immediately revoke the privileges of a user to protect critical resources on the system with one command. A contained user has no ability to access RACF-protected resources, even in active sessions. The user is effectively quarantined.

This functionality is available in z/OS 3.2 and z/OS 3.1 when you install the PTF for [APAR OA67286](http://www.ibm.com/support/pages/apar/OA67286) ([www.ibm.com/support/pages/apar/OA67286](http://www.ibm.com/support/pages/apar/OA67286)).

For more information, see “RACF user ID containment” on page 81, which is a new topic. Updated topics include “The base segment in user profiles” on page 47 and “Authorizing access to RACF-protected resources” on page 719.

- **RACF RACDCERT certificate generation support of multiple alternate names:** It is now possible to issue RACF-managed certificates with multiple subject alternate names of the same type. Your certificates can contain multiple IP addresses, domain names, URLs, and email addresses. With this support, you can minimize the number of certificates that are needed to secure access to a server that can be accessed multiple ways.

For example, certificates can contain multiple host names (such as `www.mycompany.com` and `info.mycompany.com` and `support.mycompany.com`) and multiple IP addresses, such as `9.114.118.56` and `9.114.118.58` and `2001:0db8:85a3::8a2e:0370:7334`.

“Scenario 1: Secure server with a certificate signed by a certificate authority” on page 590 is updated to show this new capability.

More information is provided in *z/OS Security Server RACF Command Language Reference*. See the description of the `ALTNAME` parameter, which is used to specify values for the certificate subject alternate name (SAN) extension. The new keywords are `AIP`, `ADOMAIN`, `AEMAIL`, and `AURI`.

- RACF has added the **DSNRAUTH** class to the `SETROPTS RACLIST` keyword. See “Supplied resource classes for z/OS systems” on page 683 for more information.

### Changed

The following content is changed.

## September 2025 release

- The R\_PgmSignVer function no longer requires the use of the nonrepudiation key usage bit. Also, this service supports the use of signing keys that are stored in ICSF. The following topics were updated:
  - [“Certificate objects required for program signing” on page 328](#)
  - [“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331](#)
  - [“Steps for enabling a user to sign a program using external code-signing certificates” on page 333](#)

## Deleted

The following content is deleted.

## September 2025 release

- None.

# Summary of changes for z/OS 3.1

---

The following content is new, changed, or no longer included in z/OS 3.1.

## New

### May 2025

- [“Authorizing access to RACF-protected resources” on page 719](#) is modified based on using the NOGENERIC operand of the RDELETE command. Specifically, it is modified where it describes the steps to take if you expect generic profiles to be used by RACF authorization checking.

### April 2025

- The following fields were added to the IDTPARMS segment in the IDTDATA class profile: SIGLABELPRIMARY and SIGKIDPRIMARY. See [“Field-level access checking” on page 200](#).

### March 2025

- The use of AES encryption for the LDAPBIND and KEYSMSTR profiles is supported when you apply the PTF for APAR OA66458. For information, see [“Steps for storing a key in a KEYSMSTR profile” on page 256](#)
- The ENCRYPTTYPES operand and ENCTYPES field are added to the DFP segment in data set profiles; see [Table 18 on page 205](#).

### November 2024

- The example was corrected in [“Field-level access checking” on page 200](#).

### August 2024

- Information is added to [“Activating the PTKTDATA class” on page 282](#).

### June 2024

- RACF AES Password Enveloping support is added to [“Signing hash algorithm and encryption strength used to create the envelope” on page 624](#).
- A new option is added for handling errors that occur when data is saved to RRSFLIST data sets. This support is added with APAR OA65286, which also applies to z/OS 2.5 and z/OS 2.4.

For more information, see the following topics:

- [“Capturing command output” on page 413](#)
- [“Output processing” on page 421](#)
- [“Output data set names for automatic direction” on page 422](#)
- Information is added to [“Protecting LLA-managed data sets” on page 234](#).

## March 2024

You can disable extra logon attempts for a RACF SPECIAL user after the SETROPTS PASSWORD(REVOKE(*nnn*)) value is exceeded. This support is added with APAR OA63091, which also applies to z/OS 2.5, z/OS 2.4, and z/OS 2.3. For more information, see [“Extending password and user ID processing \(PASSWORD option\)”](#) on page 109.

## September 2023 release

- The ACEE field is added to the CFDEF segment in general resource profiles (CFIELD class). See [“Field-level access checking”](#) on page 200.
- If user-related custom fields (extracted from the CSDATA segment of the USER profile) are available in the user ACEE, problem-state applications can retrieve this information by using the R\_GetInfo callable service. For more information, see [“R\\_GetInfo \(IRRSGL00\) callable service”](#) on page 608.
- SETROPTS APPLAUDIT is extended to include logging successful logons. OPTAUDIT is added to the Supplied resource classes for z/OS systems table. See [Appendix A, “Supplied RACF resource classes,”](#) on page 683 for more information.

## Changed

### November 2023

- Information about certificate extensions for signing IPL data is added to [“Required certificate extensions”](#) on page 345.

### September 2023 release

- None.

## Deleted

### September 2023 release

- The GeoTrust Global Certification Authority certificate is expired and is therefore removed from Appendix C. See [Appendix C, “Listings of RACF supplied certificates,”](#) on page 709.





---

# Chapter 1. Introduction

This topic introduces you to using RACF to administer security on your system.

Over the past several years, it has become much easier to create and access computerized information. No longer is system access limited to a handful of highly skilled programmers; information can now be created and accessed by almost anyone who takes a little time to become familiar with the newer, easier-to-use, high-level inquiry languages. As a result of this improved ease of use, the number of people using computer systems has increased dramatically. More and more people are becoming increasingly dependent on computer systems and the information they store in these systems.

As the general computer literacy and the number of people using computers has increased, the need for data security has taken on a new level of importance. No longer can the installation depend on keeping data secure simply because no one knows how to access the data. Further, making data secure does not mean just making confidential information inaccessible to those who should not see it; it means preventing the inadvertent destruction of files by people who might not even know that they are improperly manipulating data.

As the security administrator, it is your job to ensure that your installation's data is properly protected. RACF can help you do this.

---

## How RACF meets security needs

RACF satisfies the preferences of the end user without compromising any of the concerns raised by security personnel. The RACF approach to data security is to provide an access control mechanism that:

- Offers effective user verification, resource authorization, and logging capabilities
- Supports the concept of user accountability
- Is flexible
- Has little noticeable effect on the majority of end users, and little or no impact on an installation's current operation
- Is easy to install and maintain

## User identification and verification

RACF controls access to and protects resources. For a software access control mechanism to work effectively, it must first *identify* the person who is trying to gain access to the system, and then *verify* that the user is really that person.

RACF uses a *user ID* and a system-encrypted *password* or *password phrase* to perform its user identification and verification. When you define a user to RACF, you assign a user ID and password or a password phrase. The user ID identifies the person to the system as a RACF user. The password or password phrase verifies the user's identity.

The password or password phrase permits initial entry to the system, at which time the person is required to choose a new password or password phrase. Unless the user divulges it, no one else knows the user ID-password or password phrase combination.

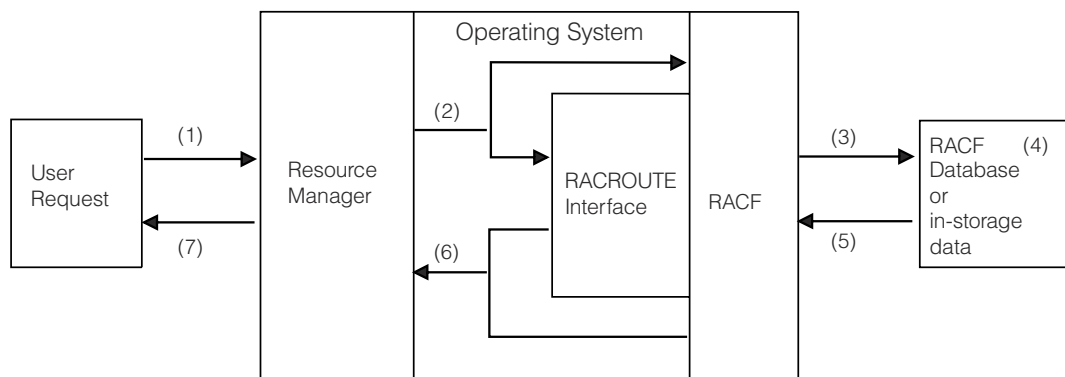
During terminal processing, RACF allows the use of an operator identification card (OIDCARD) in place of, or in addition to, the password or password phrase. (The OI DCARD information is also encrypted.) By requiring a user to know both the correct password and the correct OI DCARD, you have increased assurance that the proper user has entered the user ID.

The secured signon function provides an alternative to the RACF password called a PassTicket, which allows workstations and client machines to communicate with a host without using a RACF password or password phrase. Using this function can enhance security across a network. For more information, see [Chapter 11, "Using PassTickets," on page 281](#).

## Authorization checking

Having identified a valid user, the software access control mechanism must next control interaction between the user and the system resources. It must authorize not only what resources that user can access, but also in what way the user can access them, such as for reading only, or for updating as well as reading. This controlled interaction, or authorization checking, is shown in [Figure 1 on page 2](#). Before this activity can take place, however, someone with the proper authority at the installation must establish the constraints that govern those interactions.

With RACF, you are responsible for protecting the system resources (data sets, tape and DASD volumes, IMS and CICS® transactions, TSO logon information, and terminals) and for issuing the authorities by which those resources are made available to users. RACF records your assignments in *profiles* stored in the RACF database. RACF then refers to the information in the profiles to decide whether a user should be permitted to access a system resource.



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

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

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

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

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

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

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

Figure 1. RACF authorization checking

## Logging and reporting

The ability to log information, such as attempted accesses to a resource, and to generate reports containing that information can prove useful to a resource owner, and is very important to a smoothly functioning security system.

Because RACF can identify and verify a user's user ID and recognize which resources the user can access, RACF can record the events where user-resource interaction has been attempted. This function records actual access activities or variances from the expected use of the system.

RACF has a number of logging and reporting functions that allow a resource owner to identify users who attempt to access the resource. In addition, you and your auditor can use these functions to log all detected successful and unsuccessful attempts to access the RACF database and RACF-protected resources. Logging all access attempts allows you to detect possible security exposures or threats. The logging and reporting functions are:

- **Logging:** RACF writes records to the system management facility (SMF) for detected, unauthorized attempts to enter the system. Optionally, RACF writes records to SMF for authorized attempts and detected, unauthorized attempts to:
  - Access RACF-protected resources

- Issue RACF commands
- Modify profiles on the RACF database

RACF writes these records to an SMF data set. To list SMF records, you can use either the RACF SMF data unload utility (IRRADU00) or the RACF report writer.

- With the SMF data unload utility, you can translate the RACF SMF records into a format you can browse or upload to a database, query, or reporting package, such as Db2®/z/OS.
- With the report writer, you can select RACF SMF records to produce the reports. Because the RACF report writer was stabilized at the RACF 1.9.2 level, it cannot produce reports for all records beyond that release.

You should keep in mind that, for each logging activity that RACF performs, there is a corresponding increase in RACF and SMF processing.

For more information on logging and auditing, see *z/OS Security Server RACF Auditor's Guide*. For information about how to specify logging and auditing functions, see [z/OS Security Server RACF Command Language Reference](#).

- **Sending messages:** RACF sends messages to the security console for detected, unauthorized attempts to enter the system and for detected, unauthorized attempts to access RACF-protected resources or modify profiles on the RACF database.

As well as sending resource access violation messages only to the security console, RACF allows you to send a message to a RACF-defined TSO user. Each resource profile can contain the name of a user to be notified when RACF denies access to the resource. If the user is not logged on to the system at the time of the violation, the user receives the message when logging on.

If you are auditing access attempts, and you have selected the RACF function that issues a warning message instead of failing an invalid access attempt (to allow for a more orderly migration to a RACF-protected system), RACF records each attempted access. For each access attempt that would have failed, RACF sends a warning message (ICH408I) to the accessor, but allows the access. If a *notify user* is specified in the resource profile, RACF also sends a message to that user.

- **Keeping statistical information:** Optionally, RACF can keep selected statistical information, such as the date, time, and number of times that a user enters the system and the number of times a single user accesses a specific resource. This information can help the installation analyze and control its computer operations more effectively. In addition, to allow the installation to track and maintain control over its users and resources, RACF provides commands that enable the installation to list the contents of the profiles in the RACF database.

## User accountability

Individual accountability should probably be one of your installation's prime security objectives. A user who can be held individually accountable for actions is less likely to make mistakes or take other actions that might disrupt or compromise operations at your installation.

When an individual user accesses the system through a terminal, the concept of individual user identity is fairly obvious. With a group of production programs, however, it might be less clear just who the user is. (Is it the application owner, the job scheduling person, or the console operator?)

RACF offers you the ability to assign each user a unique identifier. (Of course, whether you establish this degree of accountability in all cases is an installation decision.)

In addition, RACF permits you to assign each user to one or more groups, which are simply collections of users having common access requirements.

## RACF users

A RACF user is identified by an alphanumeric *user ID* that RACF associates with the user. Note, however, that a RACF user need not be an individual. For example, a user ID can be associated with a started procedure. In addition, in many systems today a "user" is equated with a function, rather than an individual. For example, a service bureau customer might comprise several people who submit work as

a single user. Their jobs are simply charged to a single account number. From the security standpoint, as mentioned before, equating a user ID with anything other than an individual can be undesirable because individual accountability is lost. However, it is up to the installation, through you, to decide how much individual accountability is required.

### RACF groups

A RACF group is normally a collection of users with common access requirements. As such, it is an administrative convenience, because it can simplify the maintenance of access lists in resource profiles. By adding a user to a group, you can give that user access to all of the resources that the group has access to. Likewise, by removing a user from a group, you can prevent the user from accessing those resources. You can also use groups as *holding groups* or *data control groups*. For more information, see [Chapter 4, “Defining groups,”](#) on page 85.

The group concept is very flexible; a RACF group can be equated with almost any logical entity, such as a project, department, application, service bureau customer, operations group, or systems group. Further, individual users can be connected to several groups. Membership and authority in these groups can be used to control the scope of a user's activity.

### What RACF controls

You can use RACF to control access to:

- The system. You can require that all users, including TSO users, MVS system operators, and JES system operators (except operators on locally attached JES3 consoles), log on and supply a password or password phrase when logging on or submitting batch jobs. You can also require that all users supply a security label when logging on or submitting batch jobs.
- Many subsystems and applications, including:
  - JES, including job names, JES commands, consoles, JES input devices, spool, writers (printers) and others
  - TSO
  - IMS
  - CICS
  - Storage Management Subsystem (SMS)
  - Db2z/OS
  - VTAM®
  - APPC sessions
- Terminals
- MVS and JES consoles
- Data, including:
  - Catalogs
  - DASD and tape data sets
  - DASD and tape volumes
  - SYSIN and SYSOUT data on the JES spool
  - Message traffic
- Load modules (programs) for execution only, or copying as well
- IMS/CICS transactions
- CICS files, journals, and so forth
- Installation-defined resources
- APPC transactions and other resources
- Infoprint Server

For more information, see [“Protecting general resources” on page 19](#).

## How users and groups are authorized to access resources

Basically, a user's authority to access a resource while operating in a RACF-protected system at any time is determined by a combination of these factors:

- The user's *identity*
- The user's *attributes*
- The user's *group authorities*
- The *security classification* of the user and the resource profile
- The *access authority* specified in the resource profile

**Identity:** When defining a user, the security administrator assigns a user ID consisting of 1 - 8 characters. This is the user ID with which the user logs on to the system (or submits a batch job). When a user attempts to access RACF-protected resources, RACF uses the user ID to determine the user's access to those resources.

**Attributes:** The security administrator or a delegate can assign attributes to each RACF-defined user. The attributes determine various extraordinary privileges and limitations a user has when using the system. Attributes are classified as either user-level attributes (or, simply, user attributes) or group-level attributes:

- User attributes: You can assign the SPECIAL, AUDITOR, ROAUDIT, OPERATIONS, CLAUTH, GRPACC, ADSP, and REVOKE attributes at the system level. When you assign attributes at the system level, the privileges and limitations apply across the entire system. For detailed information about these attributes, see [“Assigning optional user attributes” on page 14](#) and [“User attributes” on page 57](#).
- Group-level attributes: When you assign an attribute at the group level, RACF confines the privileges or limitations conveyed by the attribute to the group to which it applies (and to resources, users, and groups that fall within the scope of that group). For more information about the group-SPECIAL, group-AUDITOR, and group-OPERATIONS attributes, see [“Assigning optional user attributes” on page 14](#) and [“User attributes” on page 57](#).

**Group authorities:** Each user that you define must be assigned (connected) to at least one group (called the user's default group). The security administrator or group administrator can assign a specific level of “group authority” to each user of a group. The group authorities are USE, CREATE, CONNECT, and JOIN.

If a user has USE group authority within a group, the user can access resources to which the group is authorized.

CREATE, CONNECT, and JOIN also enable the user to access resources to which the group is authorized. However, these group authorities also give the user administrative responsibilities and privileges. The USE, CREATE, CONNECT, and JOIN group authorities are described in detail in [Chapter 4, “Defining groups,” on page 85](#).

If a user is added to a RACF group (via the CONNECT command) after that user has already logged on, that user will have to log off and log back on to have authority based on that group when accessing resources in classes that have been RACLISTed.

If a user is deleted from a RACF group (via the REMOVE command) after that user is already logged on, that user will have to log off and log back on to not have authority based on that group when accessing resources in classes that have been RACLISTed.

**Security classification:** Each user and each resource can have a security classification specified in its profile. The security classification can be a security level, one or more security categories, or both. A security *label* is an installation-defined name that refers to a combination of a security level and zero or more security categories. A security *level* is an installation-defined name that corresponds to a numerical security level (the higher the number, the higher the security level). A security *category* is an installation-defined name corresponding to a department or an area within an organization that has similar security requirements.

When a user requests access to a resource that has a security classification, RACF compares the security classification of the user with the security classification of the resource. For more information on security classifications, see [Chapter 5, “Classifying users and data,” on page 95](#).

**Access authority:** The access authority determines to what extent the specified user or group can use the resource. The owner of a profile protecting a data set or general resource (such as a tape volume or terminal) can grant or deny a user or group access to that resource by including the user ID or group name in the resource profile's access list. Associated with each user ID or group name is an access authority that determines whether the user or group can access the resource, and if they can access the resource, how they can use it.

The access authorities are NONE, EXECUTE, READ, UPDATE, CONTROL, and ALTER (see [Table 49 on page 703](#)).

For data set profiles and profiles in the SERVAUTH class, an entry in the access list might also contain the name of a program that is associated with the user ID and the access authority. In this case, the user must be executing that program to access the resource. For more information, see [“Program access to data sets \(PADS\) in BASIC mode” on page 310](#) and [“Program access to SERVAUTH resources in BASIC or ENHANCED mode” on page 314](#).

For general resource profiles, an entry in the access list might also contain the name of a RACF-defined terminal, console, JES input device, partner LU, SERVAUTH resource (representing the name of a network security zone), or CRITERIA that is associated with the user ID and the access authority. In this case, the user must be using the terminal, console, JES input device, partner LU name, SERVAUTH, or CRITERIA to access the resource. For more information, see [“Conditional access lists for general resource profiles” on page 193](#).

Each resource also has a universal access authority (UACC) associated with it. The UACC can be NONE, EXECUTE, READ, UPDATE, CONTROL, or ALTER. The UACC is the access authority allowed to any group or non-restricted user who is not authorized in the access list. UACC applies to all users, whether or not they are RACF-defined, unless they are defined with the RESTRICTED attribute. For information about assigning the RESTRICTED attribute, see [“Defining restricted user IDs” on page 75](#).

### ***Using ID(\*) on the access list***

If you have some users who are not defined to RACF, you can use the ID(\*) entry on the access list instead of UACC to ensure that only RACF-defined users, except those with the RESTRICTED attribute, can access the resource. The following examples illustrate the difference between UACC(READ) and ID(\*) ACCESS(READ):

- To allow *all* users on the system to use a terminal, specify UACC(READ) for the profile, as follows:

```
RDEFINE TERMINAL profile-name UACC(READ)
```

- To allow only *RACF-defined* users on the system to use a terminal, specify UACC(NONE) for the profile, then issue the PERMIT command with ID(\*) and ACCESS(READ) specified:

```
RDEFINE TERMINAL profile-name UACC(NONE)  
PERMIT profile-name CLASS(TERMINAL) ID(*) ACCESS(READ)
```

Neither the ID(\*) entry on the access list nor the UACC is used to allow a restricted user to access a RACF-protected resource.

## **RACF profiles**

As the security administrator or a delegate defines authorized users, groups, and protected resources, RACF builds *profiles*, which contain the information RACF uses to control access to the protected resources. Each profile is owned by a user or group. (By default, the owner of a profile is the user who creates it.)

You can work with the following types of profiles:

- User profiles

- Group profiles
- Data set profiles
- General resource profiles

User and group profiles contain descriptions of the authorized users of a RACF-protected system. Data set and general resource profiles contain descriptions of the resources and the levels of authority that are necessary to access these resources.

## Flexibility

Because the security requirements at every installation differ, RACF is flexible enough to assist each installation in meeting its own security objectives. There are a number of ways RACF accomplishes this:

- **Administrative control:** RACF allows you a wide range of choices in controlling access to your installation's resources. RACF allows you to use either centralized or decentralized administration techniques by permitting you to delegate authority, establish appropriate group ownership structures, and specify various group-related user attributes. In addition, RACF provides a wide range of processing options and installation exits.

Most RACF command functions, except those performed by RACMAP, RVARY, SET, TARGET, the RACF report writer command (RACFRW), and the block update command (BLKUPD), have Interactive System Productivity Facility (ISPF) entry panels and associated help panels. These panels make it easy to enter command options on TSO.

- **Generic profiles:** RACF generic profiles allow you, your group administrators, and other users to define profiles that consolidate the security requirements of several similarly named resources that have the same access requirements.
- **Protection of installation-defined classes:** RACF allows you to protect your own installation-defined resource classes. To do this, you can add entries to the class descriptor table (CDT) for the new classes, create profiles in the class, and, when a user requests access to a resource (or takes an action you want to control), issue the RACROUTE REQUEST=AUTH macro from your application to check authorization. You can control which users and groups can access each resource in the class by defining profiles in the class. The profiles can include access lists and other information such as auditing, security labels, and so forth, as with profiles in the CDT classes supplied by IBM.

See [Appendix A, “Supplied RACF resource classes,” on page 683](#) for a description of each CDT class supplied by IBM. See [Chapter 10, “Administering the dynamic class descriptor table \(CDT\),” on page 263](#) for details about creating installation-defined resource classes.

- **Installation exits:** RACF installation exits allow you to tailor RACF to specific needs of your installation. For more information, see [“Using RACF installation exits to customize RACF” on page 21](#).

Because of RACF's flexible design, you and your technical support personnel can tailor RACF to operate smoothly within the local operating environment.

## RACF transparency

No users want their data destroyed or altered by other individuals (or themselves) except when they specifically intend it. Unfortunately, users of all types are often reluctant to take steps to protect what they have created. It is not uncommon to see live data used as test data, or to see data deliberately *under-classified* to avoid having to use the security procedures that the appropriate classification would demand. In many cases, people find it easier to ignore security procedures than to use them. Even conscientious users can forget to protect a critical piece of data. The solution to implementing effective security measures, then, is to provide a security system that is transparent (painless) to the user.

With RACF, end users need not be aware that their data is being protected for them. Security and group administrators can use generic profiles to make using RACF transparent to the majority of the installation's end users. Administrators can also use profile modelling to enhance RACF's transparency.



# Implementing multilevel security

If your installation's computing system must implement multilevel security, RACF can help. For detailed information, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

## Multilevel security

---

The United States Department of Defense (DoD) has established security criteria for its computer systems and for those systems that perform government work under contract. Each system is evaluated and awarded a security rating, depending on the extent the system protects resources and its own processing.

Although IBM does not claim compliance with any particular criteria, the multilevel security (MLS) functions of a z/OS Version 1 Release 5 system and higher are designed to provide a high level of security. See [z/OS Planning for Multilevel Security and the Common Criteria](#) for details.

## Characteristics of a multilevel-secure environment

The security policy that you implement in a multilevel-secure environment has as its key feature a system of access controls that not only prevents individuals from accessing information at a classification for which they are not authorized, but also prevents individuals from declassifying information. The system must protect resources of different levels of sensitivity.

These are the key aspects of a multilevel-secure environment:

- Mandatory access control (MAC)
- Security labels
- Discretionary access control (DAC)
- Resource reuse
- Identification and authentication
- Auditing

### Mandatory access control (MAC)

Mandatory access control is a method of limiting access to resources based on the sensitivity of the information that the resource contains and the authorization of the user to access information with that level of sensitivity.

You define the sensitivity of the resource by means of a security label. The security label is composed of a security level and zero or more security categories. The security level indicates a level or hierarchical classification of the information (for example, Restricted, Confidential, or Internal). The security category defines the category or group to which the information belongs (such as Project A or Project B). Users can access only the information in a resource to which their security labels entitle them. If the user's security label does not have enough authority, the user cannot access the information in the resource.

### Security labels

Security labels can be associated with all users and resources in the system. The system uses these labels to determine if access to a resource is allowed under the mandatory access control (MAC) rules. Security labels, maintained in the RACF database, are usually defined by the security administrator and can be changed only by that person.

When a resource is exported to a device attached to a system, the security label of the resource remains in effect. Whether the resource resides on a single-level device, such as a tape drive that does not process information at different levels of security concurrently, or a multilevel device, which is able to process data at different security levels concurrently, the system continues to associate the security label with the resource.



The system provides security labels on each page of print output as a default. The system allows a user to request that no security labels be printed; however, the system is able to audit all such requests.

## Discretionary access control (DAC)

Discretionary access control is a method of limiting access to resources (such as data sets) based on the identity of users or groups to which the users belong. DAC protects all system resources from unauthorized access down to a single user. A user who does not have permission to access a resource can be granted this permission by the resource's owner.

## Resource reuse

Resource reuse is a practice that ensures all system resources (such as tape data sets) that are reused, reassigned, or reallocated are purged of all residual data, including encrypted data, belonging to the former owner.

## Identification and authentication

Identification and authentication is a method of enforcing individual accountability by providing a way for each user to be uniquely identified. Users must then have their identity associated with any security-related, audited action they might take.

## Auditability of security-related events

Auditability of security-related events is the recording of facts that describe a security-related event in a computing system. These facts include the time and date of the event, the name of the event, the name of the system resources affected by the event, the name of the user who invoked the event, and so forth. The following characteristics are also important:

- An audit record contains the audited resource's security label.
- More selective options are available for audit reports.
- The system can audit any override of labeling on printed output.

## Administering security

---

The security administrator's job can range from helping high-level management initially define corporate security policy to authorizing individual end users to access RACF-protected resources. As security administrator, you are responsible for implementing RACF at your installation. You have the authority to review and approve all implementation phases, select the resources to be protected, and plan the order in which protection is implemented. You are the authority for all RACF implementation questions. You decide the degree to which decentralization of security controls takes place. You create profiles for the implementation team, select the team members, and direct their efforts.

## Delegating administration tasks

Although you have responsibility for overall security at your installation, you can decentralize much of the security operation by delegating various RACF security responsibilities to assistants. You can appoint:

- **Group administrators:** Group administrators have many of the duties and responsibilities of a security administrator, but at a less inclusive level. Typically, a group administrator is responsible for defining the access requirements for the resources belonging to a single group. In some cases, the group administrator might delegate responsibilities in the same way as you delegated yours.
- **Technical support:** The technical support person is typically a system programmer whose job is to install operating systems, apply fixes to problems in the operating systems, and write necessary programs to interface between operating system programs and application programs. The technical support person is responsible for providing you with technical assistance, installing and maintaining RACF, and extending RACF to meet installation needs, as you direct. Technical support activities can include maintaining the RACF database.

- **Auditor:** The auditor supports the security implementation by ensuring that the levels of protection are adequate and that security exposures are reduced or eliminated. In addition, the auditor monitors operations to ensure that security procedures are being carried out properly.

In certain installations, it is possible that some of these functions might be combined. Further, the amount of delegation varies from installation to installation. In some installations, there might be much delegation of authority, and there might be more than one technical support person or more than two levels of group administrators. Similarly, other roles might differ somewhat from the way they are described in this document.

For details about defining profiles to delegate administration tasks, see [“Planning for profiles in the FACILITY class” on page 211](#).

## As of z/VM 7.3, sharing of the RACF database between z/OS and z/VM is not allowed

In z/VM® 7.3, the RACF database is changed so that it is identified as a RACF for z/VM 7.3 database. As a result of this change, the sharing of the RACF database between z/OS and z/VM system is no longer supported, as of z/VM 7.3.

No similar change is made to the RACF database for RACF/VM 7.2 and earlier release, nor for any release of RACF for z/OS. Sharing the RACF database between z/OS and z/VM systems is still supported in these configurations.

To prevent the corruption of RACF for z/OS databases, RACF for z/VM 7.3 rejects a RACF for z/OS database.

To prevent the corruption of RACF for z/VM 7.3 databases, RACF for z/OS APAR OA62875 adds support to detect and reject a RACF for z/VM 7.3 database. RACF for z/VM 7.3 databases are not accepted in the following situations:

- IPL
- RVARY ACTIVATE
- IRRMIN00 PARM=UPDATE

**Note:** With the following exception: IRRMIN00 PARM=UPDATE updates a z/VM 7.3 RACF database with template level RPI7300.10000000.00000000 to the current z/OS template level. This capability exists to help simplify migration to z/VM 7.3 and recovery from errors.

Other RACF utilities continue to run against RACF for z/VM 7.3 databases.

PTFs for APAR OA62875 are available for z/OS V2R3 and later.

If the PTF for RACF for z/OS APAR OA62875 is not yet installed, RACF for z/OS accepts and uses the RACF for z/VM 7.3 database. In the short term, RACF for z/OS and RACF for z/VM remain compatible and no corruption occurs. However, RACF database incompatibility will occur over time as RACF for z/OS and RACF for z/VM 7.3 deliver service and features that update the RACF database templates.

RACF for z/OS service that can potentially impact the compatibility between RACF for z/OS and RACF for z/VM will require the installation of the PTF for APAR OA62875. Then, if the database is identified as a RACF for z/VM 7.3 database, RACF for z/OS rejects the database.

For more information, see the following scenarios for configuration details:

- [“Scenario: RACF for z/VM 7.3 was installed prior to the installation of OA62875, and z/OS is using a z/VM 7.3 GA level database” on page 11](#)
- [“Scenario: RACF for z/VM 7.3 is installed prior to the installation of OA62875, and z/OS is using a z/VM 7.3 GA level database, and a RACF for z/VM template update is applied to the z/VM 7.3 database” on page 11](#)
- [“Support for older releases of z/VM and z/OS” on page 11](#)

### Documentation notes:

- In accordance with this change, the RACF publications for z/OS V2R5 are updated to remove references to database sharing with RACF/VM. The RACF publications for z/OS V2R4 and earlier continue to document aspects of sharing the RACF database with RACF for VM 7.2 and earlier. This documentation still applies to all releases of z/OS that share the RACF database with RACF for z/VM release 7.2 and earlier.
- The RACF for z/VM 7.3 publications require that you copy any RACF databases that are shared between z/OS and z/VM and reconfigure z/VM to use the copy. Perform this action before you convert the RACF database into a z/VM 7.3 database.

### **Scenario: RACF for z/VM 7.3 was installed prior to the installation of OA62875, and z/OS is using a z/VM 7.3 GA level database**

If the PTF for RACF APAR OA62875 is installed after z/VM is upgraded to release 7.3, and the database in use by z/OS is marked as a RACF for z/VM 7.3 database, RACF for z/OS no longer accepts the RACF for z/VM 7.3 database.

Ensure that the RACF database is no longer shared with RACF for z/VM 7.3. Then, use IRRMIN00 PARM=UPDATE to update the templates to the latest z/OS template level and mark the RACF database as a RACF for z/OS database.

IRRMIN00 PARM=UPDATE works only for the RACF for z/VM 7.3 GA level of the database. To prevent database corruption, this utility does not work after template updates are applied beyond the RACF for z/VM 7.3 GA level.

By using IRRMIN00 PARM=UPDATE, you can recover the database without losing updates that were made after z/VM is migrated to Release 7.3.

Alternatively, after you ensure that the RACF database is no longer shared with RACF for z/VM, you can restore a backup of the RACF database.

### **Scenario: RACF for z/VM 7.3 is installed prior to the installation of OA62875, and z/OS is using a z/VM 7.3 GA level database, and a RACF for z/VM template update is applied to the z/VM 7.3 database**

If the PTF for APAR OA62875 is not installed on z/OS, z/OS does not prevent sharing the RACF database with z/VM 7.3. This configuration is not supported. If template updates are applied to the shared RACF database by RACF for z/VM starting in release 7.3, the RACF database might become irrevocably unusable by z/OS when the PTF for APAR OA62875 is applied.

IRRMIN00 PARM=UPDATE cannot be used to update the z/VM RACF database because its templates are beyond the RACF for z/VM 7.3 GA level. You might need to restore a copy of the RACF database that predates the installation of RACF database templates beyond the z/VM 7.3 GA level.

### **Support for older releases of z/VM and z/OS**

Sharing of the RACF database at all releases of z/OS with releases of z/VM before 7.3 is unaffected and continues to be supported. This is true regardless of whether the PTF for APAR OA62875 is applied to z/OS. Future database template updates to RACF for z/OS after APAR OA62875 might render a RACF database that is shared with z/VM 7.2 (and earlier) non-migratable to z/VM 7.3.

Consider separating your z/VM and z/OS RACF databases now if you plan to upgrade your z/VM system to release 7.3 in the future.

APAR OA62875 is not available on releases earlier than z/OS V2R3. Though these earlier z/OS releases are not prevented from sharing the RACF database with z/VM 7.3, this configuration is not supported. If template updates are applied to the shared RACF database by RACF for z/VM 7.3, the RACF database becomes irrevocably unusable on a supported release of z/OS.

IRRMIN00 PARM=UPDATE cannot be used to update the z/VM RACF database because its templates are beyond the RACF for z/VM 7.3 GA level. Restore a copy of the RACF database that predates the installation of RACF database templates beyond the z/VM 7.3 GA level in this situation.

Consider separating your z/VM and z/OS RACF databases now if you plan to upgrade your z/OS system to a supported release in the future.

## Using RACF commands or panels

After you have planned for RACF implementation (see [Chapter 2, “Organizing for RACF implementation,”](#) on page 33), you can perform security and group administration tasks by using various RACF commands. For example, you can use the ADDGROUP command to define a new group as a subgroup of an existing group; you can use the ADDUSER command to define a new user and connect the user to the user's default group; you can use the ADDSD command to protect a DASD data set, and so on. (Sample command sequences for administrative tasks are given throughout this document. See [z/OS Security Server RACF Command Language Reference](#) for the attributes and authorities you need to use RACF commands.)

The RACF commands include operands with which you specify the various user attributes, group authorities, and access authorities. RACF places the information it receives from the commands into various profiles (user, group, data set, and general resource profiles), which it keeps in the RACF database and uses to control subsequent access to resources.

As an alternative to using RACF commands to perform administration tasks, you can use RACF's ISPF panels if the ISPF product is installed at your location. If you use the panels, you do not need to memorize command or operand names; you only need to complete the appropriate information on the proper panels.

## Using the ISPF panels

In general, you can perform the same RACF functions using the ISPF panels.

The **ISPF panels** provide the following advantages:

- When you use the panels, you avoid having to memorize a command and type it correctly. Panels can be especially useful if the command is complex or you perform a task infrequently.
- ISPF creates in the ISPF log a summary record of the work that you do. Unless you use the TSO session manager, the RACF commands do not create such a record.
- From the panels, you can press the HELP key to display brief descriptions of the fields on the panels.
- The options chosen when installing the RACF panels determine whether output (for example, profile listings, search results, and RACF options) is displayed in a scrollable form.
- The ISPF panels for working with password rules allow you to enter all of the password rules on one panel. The figure below shows one of these panels.
- When you use the ISPF panels to update a custom field definition in the CFDEF segment, the current values are displayed. You can then overwrite the values to make changes.
- When you use the ISPF panels to add, update, or delete custom field information (CSDATA segment fields), the panels are primed with the custom field names and values. You can then make additions, changes, and deletions.

**Limitations:** The following limitations apply to the use of the ISPF panels:

- The ISPF panels do not support all options of all commands. For example, the SETROPTS PASSWORD option to activate and deactivate mixed-case password support is not available through the RACF panels.
- The ISPF RACF panels are limited to 32000 lines of command output. If the output listing for a command (most commonly, the RLIST command) exceeds 32000 lines, the output is truncated at the 32000 line limit and an error is likely to occur. To avoid this limitation, use one of the following alternate methods:
  - Issue the command using a batch execution of the terminal monitor program (TMP) and use the SDSF **XD** command to store the output in a data set.
  - Create a report using output from the RACF database unload (IRRDBU00) utility.

```

                                RACF - SET PASSWORD FORMAT RULES
COMMAND ===>

Enter PASSWORD FORMAT RULES:
                                MINIMUM  MAXIMUM
                                LENGTH    LENGTH    FORMAT
RULE 1:  --                    --          -----
RULE 2:  --                    --          -----
RULE 3:  --                    --          -----
RULE 4:  --                    --          -----
RULE 5:  --                    --          -----
RULE 6:  --                    --          -----
RULE 7:  --                    --          -----
RULE 8:  --                    --          -----
To cancel an existing rule, enter NO for MINIMUM LENGTH.
To specify FORMAT, use the following codes for each character position:
* = Any Character      $ = National      V = Vowel      N = Numeric
C = Consonant          A = Alphabetic     v = Mixed Vowel  m = Mixed Numeric
c = Mixed Consonant    L = Alphanumeric  W = No Vowel    s = Special
x = Mixed All

```

Figure 2. Sample ISPF panel for RACF

## Additional authorization for using the ISPF panels

You must authorize general users to use ISPF panels to add data to custom fields in user and group profiles. For details, see [“Authorizing users to update data in a custom field”](#) on page 645.

## RACF group and user structure

Two of the fundamental elements of RACF are users and groups. Users, of course, are the many people who log on to a system, each with a unique user ID. Administration of a small number of users is not too difficult. However, when there are thousands of users, administration becomes a very large task. To make this task more manageable, the concept of groups was developed.

A group is a RACF entity with which any number of users are associated. Usually, the users in a group have some logical relationship to one another. The relationship used most frequently is members of a department. Many installations pattern their group-user structure after their organization charts.

At the top of the RACF group-user structure is a group called SYS1. When you install RACF, it defines this group for you. The SYS1 group is the highest group in the total RACF group-user structure. You can define your system administrator and system auditor as members of this group. The system administrator has the SPECIAL attribute and the system auditor has the AUDITOR attribute. The significance of SPECIAL and group-SPECIAL, AUDITOR and group-AUDITOR, ROAUDIT, and the differences between them, are described in later sections.

## Defining users and groups

You define users to RACF by issuing RACF commands that include various user attributes, as well as other control information that RACF uses. The following are some of the commands you might use in your user-definition tasks. For a more complete description of the process of defining users, see [Chapter 3, “Defining users,”](#) on page 45. For complete descriptions of RACF commands, see the [z/OS Security Server RACF Command Language Reference](#).

### Commands for user administration:

#### ADDUSER

Add a user profile to RACF.

#### ALTUSER

Change a user's RACF profile.

#### CONNECT

Connect a user to a group.

#### DELUSER

Delete a user profile from RACF and remove connection to all groups.

### **REMOVE**

Remove a user from a group and assign a new owner for group data sets owned by the removed user.

### **LISTUSER**

Display the contents of a user's profile.

### **PERMIT**

Permit a user to access a resource (or deny access to a resource).

### **PASSWORD or PHRASE**

Change a password or password phrase.

In addition to defining individual users, you can define groups of users. Group members can share common access authorities to a protected resource.

One benefit of grouping users is that you can authorize the entire group, as a single unit, to access a protected resource. Another benefit is that attributes such as OPERATIONS can be assigned so that a given user has that attribute only when connected to a specific group, and the attribute is only effective for resources within the scope of that group.

The following are some of the commands you might use in your group-definition tasks.

### **Commands for group administration:**

#### **ADDGROUP**

Define a new group (a subgroup of an existing group).

#### **ALTGROUP**

Assign a subgroup to a new superior group.

#### **DELGROUP**

Delete one or more groups.

#### **LISTGRP**

Display the contents of a group profile.

#### **CONNECT**

Connect a user to a group.

#### **REMOVE**

Remove a user from a group and assign a new owner for group data sets owned by the removed user.

#### **PERMIT**

Permit a group of users to access a resource (or deny them access to a resource).

## **Assigning optional user attributes**

You can assign user attributes by specifying operands on RACF commands. User attributes describe various extraordinary privileges, limitations, and processing environments that can be assigned to specified users in a RACF-protected system.

You can assign user attributes at either the system level or at the group level. When assigned at the system level, attributes are effective for the entire RACF-protected system. When assigned at the group level, their effect is limited to profiles that are within the scope of the group.

The scope of control of a group-level attribute percolates down through a group-ownership structure from group to subgroup to subgroup, and so on. Percolation is halted (and therefore the scope of control of the group-level attribute is ended) when a subgroup is owned by a user instead of a superior group. [Figure 3 on page 15](#) shows an example of the scope of control of an attribute assigned at the group level.

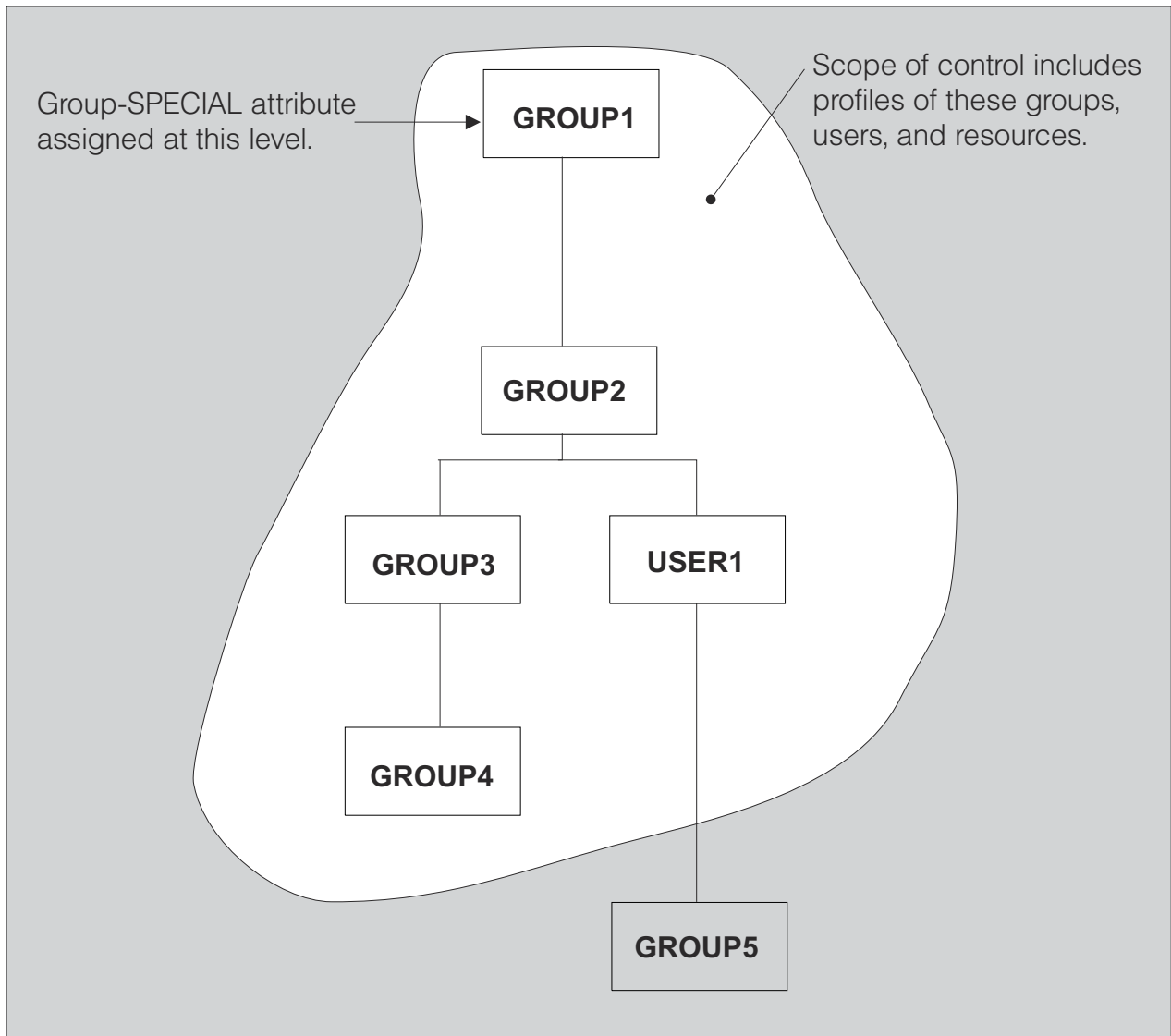


Figure 3. Scope of control of an attribute assigned at the group level

Figure 3 on page 15 shows a group ownership structure. In this figure, GROUP1 owns GROUP2, GROUP2 owns GROUP3 and USER1, and so on. A user who is connected to GROUP1 with the group-SPECIAL attribute has an explicit scope of control as shown in the figure. That is, the user cannot modify any profiles owned by GROUP5. Table 1 on page 15 lists and describes attributes that can be assigned at the user and group level. For a more complete description, see Chapter 4, “Defining groups,” on page 85.

Table 1. User attributes

User attribute	Description
SPECIAL	<p>When you assign it at the system level, the SPECIAL attribute gives the user full control over all of the RACF profiles in the RACF database. At the system level, the SPECIAL attribute allows the user to issue all RACF commands.</p> <p>When you assign the SPECIAL attribute at the group level, the group-SPECIAL user has full control over all of the resources that are within the scope of the group but cannot issue RACF commands that would have a global effect on RACF processing.</p>

Table 1. User attributes (continued)

User attribute	Description
AUDITOR	<p>When you assign it at the system level, the AUDITOR attribute gives the user full responsibility for auditing the security controls and use of system resources across the entire system. With the AUDITOR attribute at the system level, the user can specify logging options on the RACF commands and list the auditing options of any profiles using the RACF commands. In addition, the user can control additional logging to SMF for detecting changes and attempts to change the RACF database and for detecting accesses and attempts to access RACF-protected resources.</p> <p>When you assign the AUDITOR attribute at the group level (that is, when you assign the group-AUDITOR attribute), auditing authority is limited to resources that are within the scope of the group.</p>
ROAUDIT	<p>When you assign it at the system level, the ROAUDIT attribute gives the user responsibility for auditing the security controls and use of system resources across the entire system without the authority to alter the system logging options. With the ROAUDIT system level attribute, the user can list the auditing options of any profiles using the RACF commands.</p>
OPERATIONS	<p>When you assign the OPERATIONS attribute at the system level, the user can perform any maintenance operations on RACF-protected resources, such as copying, reorganizing, cataloging, and scratching data.</p> <p>At the group-OPERATIONS level, authority to perform these operations is limited to resources that are within the scope of the group.</p>
CLAUTH	<p>The CLAUTH (class authority) attribute allows the user to define profiles in a specific RACF class. A user can have class authority for the USER class and any of the classes that are defined in the class descriptor table (CDT). Examples of classes that IBM supplies in the CDT are the TERMINAL class (for terminals) and the TAPEVOL class (for tape volumes). For a list of valid class names, see <a href="#">Appendix A, “Supplied RACF resource classes,”</a> on page 683.</p> <p>For a list of the RACF commands that the CLAUTH attribute allows users to issue, see <a href="#">Table 47</a> on page 701.</p> <p>If the SETROPTS GENERICOWNER option is in effect, this authority is limited. See <a href="#">“Restricting the creation of general resource profiles (GENERICOWNER and ENHANCEDGENERICOWNER options)”</a> on page 113.</p>
GRPACC	<p>When a user with the GRPACC attribute creates a data set profile for a group data set, RACF gives UPDATE access authority to other users in the group (if the user defining the profile is a member of that group). A group data set is a data set whose high-level qualifier, or the qualifier derived from the RACF naming convention table, is a RACF-defined group name.</p>
ADSP	<p>The ADSP attribute establishes an environment in which all permanent DASD data sets created by this user are automatically defined to RACF and protected with a discrete profile. ADSP can be assigned at the group level, in which case it is effective only when the user is connected to that group.</p>
REVOKE	<p>The REVOKE attribute prevents the RACF-defined user from entering the system. REVOKE can be assigned at the group level, in which case the user cannot enter the system and connect to that group.</p>



Table 1. User attributes (continued)

User attribute	Description
RESTRICTED	<p>The RESTRICTED attribute prevents a user from gaining access to a protected resource, other than a z/OS UNIX file system resource, unless the user is specifically authorized on the access list. Global access checking, the ID(*) entry on the access list, and the UACC will not be used to allow a restricted user to access a protected resource.</p> <p>To prevent a restricted user from gaining access to a z/OS UNIX file system resource unless specifically authorized, see <a href="#">“Controlling access to file system resources for restricted users”</a> on page 531.</p>

**Guideline:** You and your delegates should assign the SPECIAL, AUDITOR, ROAUDIT, and OPERATIONS attributes to the minimum number of people necessary to administer security at your installation.

## Assigning group authorities

Each user in a group might have different responsibilities for the group. These responsibilities might include creating resource profiles to be used by the group and adding new members to the group. You should assign a specific level of group authority to the user that is based on the user's responsibilities for administering and maintaining the group to which the user is connected. You can do this with the ADDUSER, ALTUSER, or CONNECT command.

The group authorities you can assign to a user are (in order of least to most authority): USE, CREATE, CONNECT, and JOIN. Each higher level authority includes the lower levels of authority. The group authorities are defined generally as follows:

- The USE authority permits the user to access resources to which the group is authorized.
- The CREATE authority permits the user to create group data set profiles.
- The CONNECT authority enables the user to add RACF-defined users to the group.
- The JOIN authority enables the user to define new users and new groups.

For the specific details of each group authority, see [“Group authorities”](#) on page 91.

## Profiles associated with users and groups

When you use RACF commands to define users and groups, the information RACF gathers from these commands is stored in profiles and placed in the RACF database. A general description of user and group profiles follows:

### The user profile

The user profile defines an individual user. Some of the things the user profile can contain are:

- Information about the user's identity, such as name, password or password phrase
- System-wide user attributes
- The name of the user's default group
- Whether the user's security-related activities should be logged
- How often the user's password or password phrase must be changed
- The user's security categories, security level, and default security label
- The name of an optional model profile RACF uses when a user creates new data set profiles
- A TSO segment, which contains TSO logon information
- A DFP segment, which contains default DFP information for the user
- An OPERPARM segment, which contains initial information used when the user enters an extended MCS console session

- A CICS segment, which contains initial information used when the user signs on to CICS
- For z/OS UNIX, an OMVS segment, which contains z/OS UNIX information about the user

### **The group profile**

The group profile defines a group. Some of the things the group profile can contain are:

- Information about the group, such as who owns it and what subgroups it has
- Whether the group is a universal group
- For non-universal groups, a list of all connected users (members)

**Note:** Member lists for universal groups do not contain information about all connected users, only those users with group authorities higher than USE, and those with the group-SPECIAL, group-OPERATIONS, or group-AUDITOR attributes. For more information, see [“Defining large groups with the UNIVERSAL attribute” on page 88.](#)

- For non-universal groups, the group authorities of each member

**Note:** Information about members with group authority of USE is not available for universal groups.

- The name of an optional model profile RACF uses when a user creates new group data set profiles
- A DFP segment, which contains default DFP information for the group
- For z/OS UNIX, an OMVS segment, which contains z/OS UNIX information about the group

## **Protecting resources**

In the early releases of RACF, the only resources that were protected were data sets. Over the years, enhancements to RACF, applications have broadened the meaning of the term *resource* to include the following:

- Places in the system where data resides (such as data sets)
- Places in the system through which data passes during processing (such as terminals)
- The functions by which users work with data (such as commands)

Using RACF, you can protect resources so that only authorized users can access the resource in approved ways.

In general, you control access to a protected resource by creating a *discrete* or *generic* profile.

Discrete profiles protect only one resource. The name of the profile identifies to RACF which resource is protected. For example, a profile called SMITH.REXX.EXEC in class DATASET would protect the data set named SMITH.REXX.EXEC.

Generic profiles protect one or more resources that have the same security requirements. In many cases, some of the characters in the resource names are the same. For example, a profile called SMITH.\*\* in class DATASET would protect all of SMITH's data sets that did not have a more specific profile defined.

In most general resource classes, you can also provide a *top* generic profile that protects all of the resources that are not otherwise protected.

**Tip:** A top generic profile for a class should have a profile name of \*\* (rather than \*) so that you can issue the RLIST command to display the profile itself.

Using generic profiles can greatly reduce the amount of RACF profile maintenance done by a RACF administrator.

Examples of discrete and generic profiles are shown throughout this document.

### **Protecting data sets**

RACF can protect the following kinds of data sets:

- VSAM data sets

- Data sets managed by the Storage Management Subsystem (SMS)
- Cataloged and uncataloged non-VSAM DASD data sets
- Tape data sets with standard labels
- Data sets that have the same name but reside on different volumes
- Generation data group (GDG) data sets

RACF protects data sets whether or not they are protected by passwords. When both RACF protection and password protection are applied to a data set, access to the data set is determined only through RACF authorization checking. That is, password protection is bypassed.

RACF protection has an advantage over password protection. With RACF protection, only authorized users can access the data set. With password protection, any user who knows the password can access the data set. Also, users can run jobs more easily using RACF protection because the system operator is not prompted for data set passwords for RACF-protected data sets that are accessed during a job.

To protect either a DASD or tape data set, a user issues the ADDSD command, which creates a data set profile and stores it in the RACF database. Alternatively, the user can specify the PROTECT=YES operand in the JCL or the PROTECT operand on the TSO ALLOCATE command. For tape data sets, the user can also predefine the tape volume using the RDEFINE command. (When protecting a tape data set, RACF also creates, under certain circumstances, a profile for the tape volume that contains the tape data set.)

You can protect data sets with either discrete or generic profiles. If a data set has unique access-authorization or logging requirements, you should define a discrete profile for it. If the requirements are the same for several data sets that share a common name structure, you can define a generic profile that applies to all of the data sets.

For information about protecting z/OS UNIX files, see [“Protecting file system resources” on page 530](#).

## Protecting general resources

To protect a general resource, create a general resource profile using the RDEFINE command. When you create a general resource profile, you must specify a general resource class for the profile.

See Appendix A, “Supplied RACF resource classes,” on page 683 for a list of the general resource classes that IBM supplies in the class descriptor table (CDT). The classes for z/OS systems are relevant to the system on which you run Security Server (RACF).

## Installation-defined classes

You can dynamically add new class descriptor table (CDT) entries or modify or delete existing entries that you have added in the dynamic installation-defined CDT by administering resources in the CDT resource class. See [Chapter 10, “Administering the dynamic class descriptor table \(CDT\),” on page 263](#) for details. If you need to administer installation-defined entries in the static CDT (module ICHRRCDE), see *z/OS Security Server RACF System Programmer's Guide* and consult your system programmer.

When you define a new resource class, you can optionally designate that class as either a resource *group* class or a resource *member* class. For a resource group class, each user or group of users that is permitted access to that resource group is permitted access to all members of the resource group. Note that for each resource group class you create, you must also create a second class that represents the members of the group.

RACF refers to the class descriptor table (CDT) when it needs to make a class-related decision (such as, “What is the maximum length of profile names?”). With the CDT and appropriate use of RACF authorization checking services, you can extend RACF protection to any part of your system.

For more information on creating installation-defined classes, see [Chapter 10, “Administering the dynamic class descriptor table \(CDT\),” on page 263](#).

## Authority to create resource profiles

Users can create data set profiles if any of the following is true:

- The high-level qualifier of the profile matches their user ID.
- A high-level qualifier matches a group in which they have CREATE authority.
- They have the SPECIAL or group-SPECIAL attribute.

Users can create general resource profiles if either of the following is true:

- They have the CLAUTH attribute for the class.
- They have the SPECIAL attribute.

For complete descriptions of the authorizations required for any RACF command, see the description of the command in [z/OS Security Server RACF Command Language Reference](#).

If the SETROPTS GENERICOWNER option is in effect, further limitations apply. See “Restricting the creation of general resource profiles (GENERICOWNER and ENHANCEDGENERICOWNER options)” on [page 113](#).

## Authority to modify or delete resource profiles

To modify or delete a *generic* profile, you must meet at least one of the following criteria:

- Own the profile or, for data set profiles, have a user ID that matches the high-level qualifier of the profile name
- Have the SPECIAL attribute (or group-SPECIAL attribute, if applicable)

To modify a *discrete* profile, you must meet at least one of the following criteria:

- Own the profile or, for data set profiles, have a user ID that matches the high-level qualifier of the profile name
- Have the SPECIAL attribute (or group-SPECIAL attribute, if applicable)
- Have ALTER authority to the profile<sup>1</sup>

For complete descriptions of the required authorizations to any RACF command, or if adding members, see the description of the command in [z/OS Security Server RACF Command Language Reference](#).

## Owners of resource profiles

In general, when you create a RACF profile, you become the owner of the profile unless you specify otherwise. You can choose to specify either a RACF group or a RACF-defined user ID:

- If you make a *user* the owner of the RACF profile, the user can modify, list, and delete the profile, or name another user to become the owner.
- If you make a *group* the owner of a RACF profile, you extend the scope of the group (and, in some cases, the scope of its superior groups) to the RACF profile. If users have the group-SPECIAL, group-AUDITOR, or group-OPERATIONS attributes in these groups, their authority extends to the new profile. Further, if the profile is a group profile, the scope can extend to profiles owned by the group itself.

For a list of the RACF commands that owners of resource profiles can issue, see [Table 50 on page 704](#).

The concept of ownership of any kind of RACF profile (user, group, or resource) is different from other kinds of ownership:

- When a user attempts to access a protected resource, the user might be considered an "owner" of the resource, and be given the equivalent of ALTER access authority. This is true, for example, when a user opens a data set whose high-level qualifier matches the user's user ID.
- In data set profiles, you can specify a "resource owner" in the RESOWNER field. This field is used when users allocate new SMS-managed data sets protected by the profile. For more information, see [“Determining the owner of an SMS-managed data set” on page 501](#).

---

<sup>1</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,” on page 259](#).

## Setting up the global access checking table

You can use global access checking to improve the performance of RACF authorization checking for selected resources. For example, an entry in the global access checking table can allow all of the users on the system to have READ access to the SYS1.HELP data set.

The global access checking table is maintained in storage and is checked early in the RACF authorization checking sequence. If an entry in the global access checking table allows the requested access to a resource, RACF performs no further authorization checking. This can eliminate the need for I/O to the RACF database to retrieve a resource profile, which can result in substantial performance improvements.

If an entry in the global access checking table allows a requested access to a resource, no auditing is done for the request.

For information on planning and setting up the global access checking table, see [“Setting up the global access checking table” on page 194](#).

## Security classification of users and data

Security classification of users and data allows installations to impose additional access controls on sensitive resources. Each user and each resource can have a security classification in its profile. You can choose among the following:

- Security levels, security categories, or both.
- You can use security labels, which are a combination of security levels and security categories, and are easier to maintain.

For more information, see [“Multilevel security” on page 8](#) and [Chapter 5, “Classifying users and data,” on page 95](#).

## Selecting RACF options

RACF options provide flexibility in the creation and administration of your RACF security system. When implemented, RACF options can effectively enhance performance and recovery.

The SETROPTS command activates many of the RACF system-wide options. For a description of the SETROPTS options, as well as others, see [Chapter 6, “Specifying RACF options,” on page 105](#).

## Using RACF installation exits to customize RACF

---

You can tailor RACF using various *installation exits* to bypass security checking or perform additional security processing or checking. (Installation exits are perhaps more in the realm of the technical support personnel and are discussed in detail in *z/OS Security Server RACF System Programmer's Guide*. However, because you are responsible for overall security control at the installation, it is necessary for you to be aware of the use of installation exits.)

See *z/OS Security Server RACF System Programmer's Guide* for complete information on coding RACF installation exits. To obtain a report that describes the RACF installation exits on your system, use the data security monitor (DSMON).

## The RACROUTE REQUEST=VERIFY, VERIFYX, AUTH, and DEFINE exits

The RACROUTE REQUEST=VERIFY, REQUEST=AUTH, and REQUEST=DEFINE macros perform user verification, access checking, and resource definition, respectively. Preprocessing installation exits are available to tailor the parameters specified by the REQUEST=VERIFY, REQUEST=AUTH, and REQUEST=DEFINE macros or to perform any additional security checks. Postprocessing exits are available to override or modify results of the RACF processing performed by these three macros. Because several of the RACF commands use REQUEST=AUTH and REQUEST=DEFINE when performing their functions, you can use the REQUEST=AUTH and REQUEST=DEFINE exits for some command tailoring.

### The RACROUTE REQUEST=LIST exits

The RACROUTE REQUEST=LIST macro, used to build in-storage copies of general resource profiles, has two exits: the preprocessing/postprocessing exit and the selection exit. RACF calls the preprocessing/postprocessing exit before RACROUTE REQUEST=LIST processing to allow the installation to alter REQUEST=LIST processing options and after REQUEST=LIST processing to perform housekeeping. RACF calls the REQUEST=LIST selection exit to resolve conflicts between new and existing profile information.

### The RACROUTE REQUEST=FASTAUTH exits

The RACROUTE REQUEST=FASTAUTH macro uses the profiles constructed in a data space by the RACROUTE REQUEST=LIST macro, and under certain circumstances by the SETROPTS RACLIST command, to perform authorization checking. REQUEST=FASTAUTH has exits that enable you to make additional security checks, or to instruct RACROUTE REQUEST=FASTAUTH to accept or fail the request to access a resource.

The RACROUTE REQUEST=FASTAUTH macro has two preprocessing and two postprocessing exits. The invocation of these exits is determined by the circumstances involved in the invocation of the macro. These circumstances include whether the macro is invoked in cross-memory, and which operands are specified. See the RACROUTE REQUEST=FASTAUTH exit information in *z/OS Security Server RACF System Programmer's Guide* for details.

### The RACF command exits

RACF provides a command exit (IRREVX01) that is invoked before and after the execution of all RACF TSO commands except the block update command (BLKUPD), RACDCERT, RACLINK, RACMAP, RACPRIV, RACPRMCK, RVARY, and RACF operator commands, such as RESTART, TARGET, and SIGNOFF.

This exit permits the installation to perform functions that include:

- Checking additional authorization
- Modifying authorization checking when these commands are issued
- Changing the options specified on the command
- Stopping the command from executing

There are other command exits that are invoked during processing of certain commands. See *z/OS Security Server RACF System Programmer's Guide* for information on which exits are invoked for each command.

### The RACF password processing exits

The new-password exit (ICHPWX01) and the new-password-phrase exit (ICHPWX11) supplement the processing that RACF performs for new passwords, password phrases, and change interval values. After RACF has performed initial syntax checking but before a password, password phrase, or change interval is changed, these exits gain control from the PASSWORD (or PHRASE) and ALTUSER commands and from RACROUTE REQUEST=VERIFY. In addition, the ICHPWX11 exit gains control from the ADDUSER command when a password phrase is set. When a password or password phrase change fails initial syntax checking, the password processing exits are *not* invoked.

### The RACF password authentication exits

You can use the password authentication exits, ICHDEX01 and ICHDEX11, to control how RACF either encodes or masks the RACF password or OIDCARD data that is stored in the RACF database. By default, RACF encoding uses a software implementation of the data encryption standard (DES) algorithm and RACF masking uses a masking routine. You can use the password authentication exits to replace the RACF DES encoding routine with your own routine.

When the KDFAES algorithm is active, the password authentication exits have limited use. See [“Specifying the encryption method for user passwords” on page 139](#) for more details.

## Tools for the security administrator

---

RACF provides a number of tools to help you monitor and control RACF events and manage the RACF database. These tools include:

- The RACF utilities
- The block update command (BLKUPD)
- The RACF report writer
- The data security monitor
- Commands to record statistics in discrete profiles
- The RACF LIST commands
- The RACF SEARCH command

### Using RACF utilities

RACF provides several utility that can help you and the RACF system programmer manage the RACF database and extract information from it:

#### Utility

#### Description

#### **IRRMIN00**

RACF database initialization utility

#### **IRRUT400**

RACF database split/merge/extend utility

#### **IRRDBU00**

RACF database unload utility

#### **IRRUT200**

RACF database verification utility

#### **IRRUT100**

RACF cross-reference utility

#### **IRRRID00**

RACF remove ID utility

#### **IRRADU00**

RACF SMF data unload utility

### **RACF database initialization utility (IRRMIN00)**

The RACF database initialization utility, IRRMIN00, formats a data set on DASD for use as the RACF database. You can use IRRMIN00 to initialize a new database or to update an existing RACF database with a new set of RACF templates.

For information about how to run IRRMIN00, see *z/OS Security Server RACF System Programmer's Guide*.

### **RACF database split/merge/extend utility (IRRUT400)**

Using the RACF database split/merge/extend utility, IRRUT400, you can split a single RACF database into two or more data sets, merge two or more data sets of the RACF database together into one or more data sets, or copy a RACF database from one type of device to another. In the process, IRRUT400 physically reorganizes the RACF profiles and compresses the RACF database.

For information about how to run IRRUT400, see *z/OS Security Server RACF System Programmer's Guide*.

### **RACF database unload utility (IRRDBU00)**

The RACF database unload utility, IRRDBU00, gives you easy access to the information in your RACF database. You can use IRRDBU00 to unload your RACF database to a sequential data set and then use



the output in a variety of ways. For example, you can view the data set directly, sort it, merge it with other data, or use it as input to your own analysis and reporting programs. In addition, you can upload the sequential data set to a database manager such as Db2z/OS, where you can easily query the data and create reports. For information about how to run IRRDBU00 and use its output effectively, see [“Using the RACF database unload utility \(IRRDBU00\)” on page 371.](#)

### **RACF database verification utility (IRRUT200)**

You can use the RACF database verification utility, IRRUT200, to identify inconsistencies in the internal organization of a RACF database. You can also use it to make an exact, block-by-block copy of the RACF database.

For information about how to run IRRUT200, see *z/OS Security Server RACF System Programmer's Guide*.

### **RACF cross-reference utility (IRRUT100)**

The RACF cross-reference utility, IRRUT100, lists all occurrences of a specified user ID or group name that appear in a RACF database. This can help you discover the relationships between various users and groups, and learn other important information about users, groups, and the resources they control. For example, you can use the output to verify that users have the right access to resources. You can also use the output to determine the resources whose ownership must be transferred before you delete a user or group from the RACF database.

All users can run IRRUT100 for their own user IDs or any user IDs they own. To run IRRUT100 for other users' IDs, you must be defined to RACF with one of these attributes:

- AUDITOR
- ROAUDIT
- SPECIAL
- group-AUDITOR
- group-SPECIAL

IRRUT100 produces a cross-reference report that describes the occurrences of each user ID or group name that you specify. Generic profile names are followed by the letter "G" in parentheses.

For information about how to run IRRUT100 and use its output, see *z/OS Security Server RACF System Programmer's Guide*.

As an alternative to IRRUT100, you can use the output from the database unload utility (IRRDBU00). For more information, see [“Using the database unload utility output effectively” on page 375.](#)

### **RACF remove ID utility (IRRRID00)**

The RACF remove ID utility, IRRRID00, can help you keep the RACF database current. You can use this utility to remove all references to group names and user IDs that no longer exist in or are about to be removed from the RACF database. Also, you can specify a replacement ID for those IDs that will be removed.

The remove ID utility processes the output of the RACF database unload utility (IRRDBU00). You can use the remove ID utility to:

1. Find all residual references to user IDs and group names that no longer exist in the RACF database.
2. Find all references to a list of user IDs and group names that you specify in the SYSIN file.

For information about how to run IRRRID00 and use its output see [“Using the RACF remove ID \(IRRRID00\) utility” on page 391.](#)

### **RACF SMF data unload utility (IRRADU00)**

You can create a sequential file from security-related audit data using the RACF SMF data unload utility, IRRADU00. Once you create the sequential file, you can use it in a variety of ways. For example, you can



view the file directly, sort it, merge it with other data, or use it as input to your own analysis and reporting programs. In addition, you can upload the sequential file to a database manager such as Db2z/OS, where you can easily query the data and create reports.

**Tip:** You can use this utility to create reports for RACF audit records that the RACF report writer is unable to process.

For information about how to run IRRADU00, see [z/OS Security Server RACF Auditor's Guide](#).

## RACF block update command (BLKUPD)

You can repair your RACF database by directly changing its internal elements (blocks) using the RACF block update command (BLKUPD). Because this direct manipulation can result in an inconsistent and therefore unusable database structure, you must be very careful when using BLKUPD. For assistance in using BLKUPD, contact your system programmer or the IBM support center.

For more information on BLKUPD, see [z/OS Security Server RACF Diagnosis Guide](#).

## Using the RACF report writer

The RACF report writer lists information contained in RACF-generated SMF records. With the RACF report writer you can:

- Collect data about successful accesses and warnings before building resource profile access lists.
- List the contents of RACF SMF records in a format that is easy to read.
- Obtain reports that describe attempts to access a particular RACF-protected resource. These reports contain the user ID, the number and type of successful accesses, the number and type of unauthorized access attempts, and the name of the user if available in the SMF record.
- Obtain reports that describe user and group activity.
- Obtain reports that summarize system and resource use.

The output from the RACF report writer includes a header page that explains the meaning of the event and qualifier numbers that appear in the SMF record listings and summary reports. The remainder of the report comes in various forms, according to your selection. You can request a general summary, SMF record listings, and summary reports.

The RACF report writer has been stabilized at the RACF 1.9.2 level and has not been enhanced for this release. For example, it does not support audit records for z/OS UNIX events. To create reports and use the audit records for those events, use the SMF data unload utility.

See [z/OS Security Server RACF Auditor's Guide](#) for complete information on using the report writer.

## Using the data security monitor

The data security monitor (ICHDSM00, usually called DSMON) is a batch program that allows authorized users to obtain a set of reports that provide information about the current status of your installation's data security environment.

These reports help you (1) check the initial steps you took to establish system security and (2) make additional security checks periodically. If the DSMON program (ICHDSM00) is defined in the PROGRAM class (that is, it is a controlled program), the user must be authorized through its access list before the program can be run. If DSMON is not a controlled program, the user must have the AUDITOR or ROAUDIT attribute to run DSMON and produce reports. (For more information on controlled programs, see [Chapter 13, "Protecting programs,"](#) on page 301.)

For more information on the DSMON reports, see ["Using the data security monitor \(DSMON\)"](#) on page 361 or [z/OS Security Server RACF Auditor's Guide](#).

## Recording statistics in RACF profiles

In addition to placing statistical information into the various profiles when you create them, you can cause RACF to dynamically record statistics (such as the number of user accesses to a protected resource) in discrete profiles. For data sets and general resource classes, you can optionally record the following statistics:

- The number of times that a resource that is protected by a discrete profile was accessed under a specific RACF authority level (such as READ or UPDATE).

**Note:** When a RACROUTE REQUEST=AUTH is accepted at a certain level of authority (such as UPDATE), this does not necessarily mean that data is actually updated.

- The number of times that a specific user or group accessed a resource protected by a discrete profile.
- The date when a resource profile was last updated.

These statistics enable you to monitor the current operation of your computing system for administrative and control purposes. You can list the statistics and other descriptive information that is recorded in RACF profiles with various RACF commands.

Statistics are not recorded for either of the following types of profiles:

- Generic profiles
- In-storage profiles (in classes for which the SETROPTS RACLIST command or RACROUTE REQUEST=LIST has been issued)

## Listing information from RACF profiles

You can list the contents of profiles by using the LIST commands, as shown in [Table 2 on page 26](#).

Command output is in line mode unless you use ISPF panels. To capture the output of RACF commands, do the following:

- Use the TSO session manager to scroll through the output.
- Submit a batch TMP job that issues RACF commands and save the held output.

As an alternative to the LIST commands, you can obtain this information from the output of the database unload utility, IRRDBU00. See [“Using the database unload utility output effectively” on page 375](#).

*Table 2. Commands to list profile contents*

Command	Function
LISTDSD	<p>Lists the contents of data set profiles and lets you determine which generic profile applies to a particular data set.</p> <p>The listing shows:</p> <ul style="list-style-type: none"> <li>• Owner of the profile</li> <li>• UACC</li> <li>• Date the profile was created</li> <li>• Users and groups that are authorized to access the data set</li> <li>• Your highest access authority to the data set the security label (if there is one), and other information</li> <li>• Other information</li> </ul> <p>See <a href="#">z/OS Security Server RACF Command Language Reference</a> for a complete description of this listing.</p>

Table 2. Commands to list profile contents (continued)

Command	Function
LISTGRP	<p>Lists the contents of group profiles. The listing shows:</p> <ul style="list-style-type: none"> <li>• Owner of the group profile</li> <li>• Superior group name</li> <li>• Users connected to the group</li> <li>• Subgroup names</li> <li>• Other information</li> </ul> <p>See <a href="#">z/OS Security Server RACF Command Language Reference</a> for a complete description of this listing.</p>
LISTUSER	<p>Lists the contents of user profiles. The listing shows:</p> <ul style="list-style-type: none"> <li>• Owner of the profile</li> <li>• User name</li> <li>• Default group name</li> <li>• Groups that a user is connected to</li> <li>• Group authorities</li> <li>• Date the password or password phrase was last changed</li> <li>• Default security label</li> <li>• Other information</li> </ul> <p>See <a href="#">z/OS Security Server RACF Command Language Reference</a> for a complete description of this listing.</p>
RACDCERT LIST	<p>Lists digital certificate information. For each digital certificate, the listing shows:</p> <ul style="list-style-type: none"> <li>• Serial number</li> <li>• Issuer's distinguished name</li> <li>• Status information</li> <li>• Up to 256 bytes of the subject's name</li> <li>• Label</li> <li>• Validity information</li> <li>• Key information</li> <li>• Other information</li> </ul> <p>For an example of this listing, see <a href="#">Figure 47 on page 553</a>.</p>
RACDCERT LISTRING	<p>Lists key ring information. For each digital certificate in the ring, the listing shows:</p> <ul style="list-style-type: none"> <li>• Ring name</li> <li>• Label assigned to the certificate</li> <li>• Owner of the certificate (ID(<i>name</i>), CERTAUTH or SITE)</li> <li>• Usage within the ring</li> <li>• Other information</li> </ul> <p>For an example of this listing, see <a href="#">Figure 48 on page 554</a>.</p>

Table 2. Commands to list profile contents (continued)

Command	Function
RACDCERT LISTMAP	<p>Lists certificate name filter information. For each filter, the listing shows:</p> <ul style="list-style-type: none"> <li>• Label assigned to the certificate name filter</li> <li>• Trust status</li> <li>• Issuer's name filter, if applicable</li> <li>• Subject's name filter, if applicable</li> <li>• Criteria information, if applicable</li> </ul> <p>For examples of this listing, see <a href="#">Figure 54 on page 574</a> and <a href="#">Figure 61 on page 580</a>.</p>
RACLINK LIST	<p>Lists user ID associations. For each association, the listing shows:</p> <ul style="list-style-type: none"> <li>• Type of association</li> <li>• Node and user ID</li> <li>• Whether password synchronization is enabled</li> <li>• Status of the association</li> </ul> <p>For an example of this listing, see <a href="#">“Listing user ID associations” on page 411</a>.</p>
RACMAP LIST	<p>Lists distributed identity filter information. For each filter, the listing shows:</p> <ul style="list-style-type: none"> <li>• Label assigned to the distributed identity filter</li> <li>• The distributed identity's user name</li> <li>• Registry name</li> </ul> <p>For examples of this listing, see <a href="#">Chapter 28, “Distributed identity filters,” on page 671</a>.</p>
RLIST	<p>Lists the contents of profiles for general resources such as tape volumes, DASD volumes, IMS transactions, and terminals. The listing shows:</p> <ul style="list-style-type: none"> <li>• Owner of the profile</li> <li>• UACC</li> <li>• Date the profile was created</li> <li>• Users and groups that are authorized to access the resource</li> <li>• Your highest access authority to the resource</li> <li>• Security label</li> <li>• Other information</li> </ul> <p>See <a href="#">z/OS Security Server RACF Command Language Reference</a> for a complete description of this listing.</p>

## Searching for RACF profile names

You can list the names of profiles that meet certain search criteria by using the SEARCH command. This command is described in [Table 3 on page 29](#).

The output of this command is in line mode unless you use ISPF panels. You can use the TSO session manager to scroll through the output from the listing commands.

**Tip:** As an alternative to the SEARCH command, you can use the RACF database unload utility (IRRDBU00) to find additional data. The output from the database unload utility used with a relational database manager can provide you with the ability to implement additional search criteria.

Table 3. Command to search for profile names

Command	Function
SEARCH	Searches the RACF database for the names of profiles (in a particular resource class) that match the criteria you specify. For example, you can search for all TERMINAL profiles that have a security level specified. You can save the list of profile names in a data set. You can easily specify RACF commands (or other commands) to be saved with the profile names, generating a CLIST that you can run against the profiles.

The search criteria can include one or more of the following:

- Profile names that contain a specific character string
- Profiles for resources that have not been referenced for more than a specific number of days
- Profiles that contain a level equal to the level you specify
- Profiles with the WARNING indicator
- Profiles that contain a specific security level, category, or label
- Profiles to which another user has access

**Rule:** Unless you have the SPECIAL attribute, you must have at least READ access authority for each profile whose name is listed as the result of your request.

## Using the LIST and SEARCH commands effectively

### Guidelines:

- Using the SEARCH command might slow the system's performance. Therefore, avoid using the SEARCH command during busy system times.
- Investigate using the database unload utility for some of your profile searches. The database unload utility need not slow the system's performance and, in some cases, provides the same information as the SEARCH command.

### Question:

How can I tell whether (or how) a data set is protected?

### Answer:

The answer is complicated by a number of factors, including the presence of discrete and generic data set profiles, whether the data set is RACF-indicated, and the setting of such system-wide options as SETROPTS GENERIC(DATASET) and SETROPTS PROTECTALL. For more information, see [“Protecting data sets”](#) on page 147.

### Question:

How can I tell if (or how) a resource (other than a data set) is protected?

### Answer:

Use the RLIST command, omitting both the GENERIC and NOGENERIC operands:

```
RLIST classname resource-name
```

For resources that have grouping classes (such as terminals, DASD volumes, and certain IMS and CICS classes), specify the related "member class" and the RESGROUP operand on the RLIST command:

```
RLIST member-class resource-name RESGROUP
```

For example, for terminal T1:

```
RLIST TERMINAL T1 RESGROUP
```

This lists the profiles in the GTERMINL class that protect terminal T1.

This example does not work for terminals protected by a generic member in the GTERMINL class.

### Question:

How can I find the data sets that a user can access?

### Answer:

Perform the following steps:

1. Find the names of the profiles the user has access to:

```
SEARCH USER(userid) NOMASK
```

The name of a discrete profile identifies which data set it protects.

2. For each generic profile listed in Step “1” on [page 30](#), list the cataloged data sets protected by the profile (assumes that the SETROPTS CATDSNS option is in effect):

```
LISTDSD DATASET(generic-profile-name) DSNS NORACF
```

**Note:** To find out *how* a user can access a particular data set (READ, UPDATE, and so forth), analyze the profile protecting the data set to determine how RACF authorization processing would respond to an access request.

3. Find the entries in the global access checking table for the DATASET class:

```
RLIST GLOBAL DATASET
```

These entries allow all users access to data sets that match.

### Question:

How can I find the general resources that a user can access?

### Answer:

This must be done one class at a time. For each class, perform the following steps (which are similar to the steps for data sets):

1. Find the names of the profiles the user has access to:

```
SEARCH CLASS(classname) USER(userid)
```

The name of a discrete profile identifies which resource it protects.

### Tips:

- a. If the resource is in a class for which there can be resource group profiles (such as GTERMINL, GDASDVOL, and so forth), issue the SEARCH command twice, once for the member class and once for the grouping class.

For example, for terminals:

```
SEARCH CLASS(TERMINAL) USER(userid)  
SEARCH CLASS(GTERMINL) USER(userid)
```

- b. If the SEARCH command shows a profile that contains a RACF variable (indicated by one or more ampersands (&) in the name), you must list the RACFVARS profile that defines the variable. For example, if you see a profile named SAMPLE.&X.DATA, use the RLIST command to list the RACFVARS profile that defines the variable:

```
RLIST RACFVARS &X
```

2. RACF provides no direct way to determine which resources a particular general resource profile protects, as in issuing the LISTDSD command with the DSNS operand. This is because there is not generally a list, stored on the system, of the various existing resources that RACF can check. There would have to be such a list for each general resource class, and there are well over 50 resource classes (from terminals to JES input devices to tape volumes). Thus, for any particular class, an auditor or administrator would have to consult with the profile owners (or system support) to determine exactly which resources a generic profile protects.
3. Find the entries in the global access checking table for the class:

```
RLIST GLOBAL classname
```

These entries allow all users access to data sets that match.

**Question:**

How can I find the user or group profiles a user can list or alter?

**Answer:**

Enter one of the following commands.

```
SEARCH CLASS(USER) USER(userid)  
SEARCH CLASS(GROUP) USER(userid)
```

**Question:**

How can I find out the members of a RACF group?

**Answer:**

Enter the following command.

```
LISTGRP groupname
```

**Question:**

How can I find out what groups a user belongs to?

**Answer:**

Enter the following command.

```
LISTUSER userid
```

See [\*z/OS Security Server RACF Command Language Reference\*](#) for more detail on the output of these commands.





---

## Chapter 2. Organizing for RACF implementation

This topic outlines the planning that your installation should do before you install RACF.

This document describes the security administrator's tasks as they relate to RACF. A successful security program, however, goes well beyond the relationship of the security administrator to the software security program your company has chosen to protect its computerized data. This topic discusses some of the early work you and other people must do before installing RACF.

---

### Ensuring management commitment

Management's decision to install RACF is not, by itself, enough to ensure adequate security at your location. Indeed, if management were to ignore security concerns after simply selecting *any* software protection package, the eventual result would most likely be the failure of the security undertaking.

To be successful, a security implementation requires a management that is involved with questions of security policy and procedures, the resources to be allocated to the security function, and the accountability of users of the computer system. Without such management support, the security procedures will fall into disuse and become more of an administrative chore than a viable protection scheme. (In fact, such a situation could breed a false sense of security that could lead to serious exposures.)

You should work with management to prepare a clear, inclusive statement of security policy. This statement should reflect:

- Corporate security policy
- Physical protection considerations
- Installation data processing security requirements
- User department security requirements
- Auditing requirements
- A statement of policy concerning outside users of the system
- The security attitudes that will be expected from all users of the system

The resultant security policy helps to ensure that a security implementation team can prepare a RACF implementation plan that is both realistic and consistent with the installation's security policy.

---

### Selecting the security implementation team

To ensure a smooth implementation of RACF, careful planning is required, starting with your selection of an implementation team.

The implementation team should include the viewpoints of all of the user types (security and group administrators, auditor, technical support personnel, operations, and end user). In addition to knowing their own areas, the implementation team representatives should be familiar with, or have access to people who are familiar with, the following areas:

- RACF
- Privacy legislation
- The installation's organization
- Installation standards
- Major application areas

As security administrator, you should lead the implementation team. For best results, you should keep the team as small as possible. You should ensure that the results of the team's work are reviewed and fully supported by management.

## Responsibilities of the implementation team

Some of the responsibilities that might be assigned to the implementation team are:

- Defining RACF security objectives
- Deciding what to protect and how to report attempted violations
- Establishing resource ownership structures
- Developing the RACF implementation plan and installing RACF
- Educating all users of the RACF-protected system

Table 4 on page 34 describes the responsibilities of typical implementation team members.

Table 4. Participants of the security implementation team

User type	Responsibility
Security Administrator	As security administrator, you have overall responsibility for RACF implementation. It is your job to ensure that the work of the implementation team is consistent with good security practice and in line with the security policy established earlier. In addition, you or your delegate administrators should be responsible for educating the installation users about how RACF will be implemented. (That is, will there be a grace period before the new security procedures take effect? How will the implementation of RACF affect the day-to-day responsibilities of each user?)
Technical Support Person	The technical support person is normally a system programmer who installs RACF and maintains the RACF database. This person has overall responsibility for the programming aspects of system protection and provides technical input on the feasibility of implementing various aspects of the implementation plan. In addition, the technical support person writes, installs, and tests RACF exit routines, if they are required. If you will have RACF installed on more than one system in your installation, the implementation team should include a technical support person for each system on which you are using RACF. For more information, see <i>z/OS Security Server RACF System Programmer's Guide</i> .
Auditor	The auditor provides guidance on good auditing practice as it relates to data security and user access. This person implements the necessary RACF logging and reporting options to provide an effective audit of security measures. For more information on the auditor's duties, see <i>z/OS Security Server RACF Auditor's Guide</i> .
User Representative	The user representative should be a prospective group administrator who represents a major application area, perhaps a user support services or liaison function.
Other Users	Other users might be considered as members of the implementation team if appropriate. For example, other users who are involved with security include CICS, TSO, and database administrators and JES, MVS, and PSF system programmers.

The rest of this topic discusses some of the major responsibilities of the security implementation team.

## Defining security objectives and preparing the implementation plan

Working from the statement of security policy as a base, the implementation team prepares an *implementation plan*. This plan should answer the question "How do we get there from here?" Experience indicates that an evolutionary implementation of security, rather than a revolutionary one, is the most successful way to bring about adequate security measures in the quickest time possible.

The implementation team needs to set priorities about the data, applications, and users that must be secured. The implementation team should plan to phase in the security controls over a period of time to give users time to adjust to them.

The implementation plan should identify the major RACF events, when each must be completed, who is responsible for each event, and interdependencies among events. In addition, the plan should take into account any other significant activity planned for the same time period that could affect the implementation (for example, new systems, hardware, and applications). At an early stage the team

should also define a pilot group for whom the protection of business data, jobs, and users will be completed before undertaking the protection of other business data. The pilot group provides a means of obtaining RACF experience before extending protection to the rest of the installation.

## Deciding what to protect

---

Every installation has varying amounts of confidential data and varying degrees of confidentiality. For example, a development laboratory might be primarily concerned with the confidentiality of new products, whereas a bank or an insurance agency would be concerned with the confidentiality of its customers' records. Generally speaking, though, all data falls into one of the following categories:

1. Very sensitive, confidential data, which requires protection from disclosure, modification, or destruction
2. Non-confidential data, which is recoverable with little inconvenience if destroyed
3. Data that falls between these two extremes, which should be protected from inadvertent or deliberate modification or destruction

Most data falls into the last category.

Obviously, the data in the first category *must* be protected. What should also be considered is how to protect the data that *ought* to be protected in a simple yet effective manner, in a way that is transparent to the user of this data. The implementation team does a *risk evaluation* of the installation's data to determine which data needs what level of protection.

The task of protecting large quantities of data can take on significant proportions unless you can acquire this protection automatically. In the case of RACF, protecting data is quite simple and, after the controls are in place, practically free from administrative overhead.

## Protecting existing data

To protect data that already exists on your system before RACF is installed, you must create RACF profiles. You can use either discrete or generic profiles. However, using generic profiles can reduce the administrative effort of this task, because one generic profile can protect many resources. For example,

- You can protect existing data sets by using the ADDSD command. You should consider creating at least one profile for each high-level qualifier (user ID or group name) on your system. You can specify profile names with the format:

```
'high-level-qualifier.*'
```

If enhanced generic naming is in effect, use:

```
'high-level-qualifier.**'
```

You must determine the appropriate UACC, access lists, and other information (such as security classification, if used) for each profile.

For resources that have unique security requirements, you must create discrete profiles.

- You can also protect existing general resources (such as tape volumes or terminals) by using the RDEFINE command. If several resources in the same class have the same access requirements, you can use one profile to protect them. Not only does this save space, but it also saves administrative time.

If the names of the resources contain some identical characters, you can usually create generic profiles whose names contain the **\***, **\*\***, or **%** character to protect the resources.

For certain classes, such as terminals, DASD volumes, and others, you can create resource grouping profiles to protect resources whose names do not lend themselves to the use of the **\***, **\*\***, or **%** character.

For any general resource class, you can define a "RACF variable" that can be used in the profile names in general resource classes. For more information about how to select the type of profile to protect a

resource, see [“Choosing among generic profiles, resource group profiles, and RACFVARS profiles” on page 188.](#)

You must determine the appropriate UACC, access lists, and other information (such as security classification, if used) for each profile.

For resources that have unique security requirements, you must create discrete profiles.

## Protecting new data

RACF provides several ways to protect new data:

- **Generic Profiles:** Use of generic profiles can decrease the amount of administrative effort because you can use a single generic profile to protect a large number of existing resources that have a similar naming structure. Generic data set profiles protect existing data sets even if they are not RACF-indicated. (See [“Choosing between discrete and generic data set profiles” on page 152](#) and [“Protection through generic profiles” on page 151.](#))
- **Automatically-created discrete profiles:** RACF automatically protects new data sets by creating a discrete profile for each data set when the user creating them has the ADSP attribute or has specified the PROTECT=YES operand on the JCL DD statement that creates the data set. This automatic definition of discrete data set profiles occurs when the resource manager issues RACROUTE REQUEST=DEFINE.

### Note:

1. ADSP and PROTECT=YES always cause the creation of a discrete profile, which is desirable for data sets that have unique access-authorization requirements. If your data sets do not have unique access-authorization requirements, consider using generic profiles.
  2. By themselves, ADSP and PROTECT=YES allow only the creator of the data set to access the protected data. One way to allow other users access to the protected data set is to use the PERMIT command to place them (or groups of which they are members) on the access list of the profile with the desired access authority. Also, if the data set being created is a group data set, and the user creating it has the GRPACC attribute in that group, all members of the group are given UPDATE access authority to the group data set.
- **Automatic profile modeling:** One way you can allow other users to access protected data is by using *automatic profile modeling*. When you use automatic profile modeling, the profile that protects a new user or group data set automatically has an access list copied from the model profile. Therefore, users defined in the access list of the model can access the newly created user or group data set. Automatic modeling is thus valuable for establishing the initial access list for newly created generic data set profiles. You can use automatic profile modeling for profiles that are created by the user's ADSP attribute, the PROTECT=YES operand of the JCL DD statement, or the ADDSD command.

## Profile modeling

Profile modeling enables RACF or an installation exit routine to copy information (such as the access list, owner, and logging options) from an existing (model) profile when defining a new profile. (The copied profile is not necessarily identical to the model profile. For a description of the differences, see [“Possible changes to copied profiles when modeling occurs” on page 37.](#)) This copying greatly reduces the effort needed to create new profiles. Some examples of using profile modeling are:

- A user can copy information from an existing profile into a new profile by using the FROM operand (and related operands) on the ADDSD or RDEFINE commands. RACF uses the specified profile as a model when creating the new profile. However, profile segment information (CICS, DFP, DLFDATA, LANGUAGE, OPERPARM, SESSION, TSO, WORKATTR, and so on) is not copied to the new profile.
- A user can copy the access list from an existing profile into another existing profile using the FROM operand (and related operands) on the PERMIT command.
- For data sets, an installation can use automatic profile modeling. A user with the SPECIAL attribute can specify MODEL(USER), MODEL(GROUP), or MODEL(GDG) on the SETROPTS command. These operands specify that RACF is to use a model data set profile for selected users, groups, or GDG data sets.

If the SETROPTS MODEL options are in effect, the MODEL operands of the ADDUSER, ADDGROUP, ALTUSER, and ALTGROUP commands specify the data set profile that is to be used as a model from which to copy information into new data set profiles.

For more information on this topic, see [“Automatic profile modeling for data sets”](#) on page 158.

- If the preceding methods are not sufficient, an installation can also use a REQUEST=DEFINE exit routine to supply either the name of a model profile or the profile itself.

## Possible changes to copied profiles when modeling occurs

When a profile is copied during profile modeling, the new profile could differ from the model in the following ways:

- RACF places the user creating the new profile on the access list with ALTER access authority or, if the user is already on the access list, changes the user's access authority to ALTER. This is true only if ADDCREATOR is in effect, or if you are creating a discrete DATASET or TAPEVOL profile with RACROUTE REQUEST=DEFINE. Otherwise, the user creating the new profile is not placed on the access list or, if the user is already on the access list, the user's authority is not changed when the access list is copied to the new profile.

If the profile being added is for a group data set and the user has the GRPACC attribute for that group, RACF places the group on the access list with UPDATE access authority or, if the group is already on the access list, changes the group's access authority to UPDATE.

**Note:** These access list changes do not occur if the data set profile is created only because the user has the OPERATIONS attribute.

- If the model profile contains members (specified with the ADDMEM operand), the members are not copied into the new profile.
- If the SETROPTS MLS option is in effect, the security label, if specified in the model profile, is *not* copied. Instead, the user's current security label is used. For more information on security labels, see [“Understanding security labels”](#) on page 98.

**Note:** When the SETROPTS MLS option is in effect, if the SETROPTS MLSTABLE option is also in effect and the user has the SPECIAL attribute, the security label specified in the model profile is copied to the new profile. For more information on security labels, see [“Understanding security labels”](#) on page 98.

- For TAPEVOL profiles, TVTOC information is not copied to the new profile.
- Even if SETROPTS NOADDCREATOR is set, the model profile access list is copied exactly. Therefore, if the creator's user ID appeared in the model's access list, the authority is copied to the new profile exactly.
- Entries in the conditional access list of the model profile are copied to the conditional access list of the new profile *only* when the condition is valid for the class of the new profile.
  - WHEN(SYSID) is valid only for the PROGRAM class. SYSID entries are copied *only* when the new profile is a PROGRAM class profile.
  - WHEN(PROGRAM) is valid only for data sets and the SERVAUTH class. PROGRAM entries are copied *only* when the new profile is a data set profile or a SERVAUTH class profile.
  - WHEN(CRITERIA) is valid only for general resource classes. CRITERIA entries are *not* copied when the new profile is a data set profile.

## Allowing a warning period

In addition to deciding what to protect, the implementation team must consider how to phase in the new security controls with minimum disruption of current work patterns. You should consider:

- Auditing all accesses allowed by a resource profile
  - Specify GLOBALAUDIT(ALL) for the resource profile.
- Auditing all protected resources in a class

- Enter the SETROPTS LOGOPTIONS command.

These commands cause SMF logging to occur for all accesses. If the profiles allow all access, the SMF records indicate what users or jobs need access to the protected resources.

RACF also provides the option of issuing a warning message to users instead of failing a request to access a resource. You can control which resources are protected in this manner by specifying the WARNING operand on the ADDSD, RDEFINE, ALTDSD, or RALTER command. When a resource check is performed, if the check fails and WARNING has been specified, RACF issues a warning message to the user, logs the access, and allows the user to access the resource.

**Note:**

1. The warning message facility applies to in-storage profiles created by the SETROPTS RACLIST command. It might or might not apply to in-storage profiles created by the RACROUTE REQUEST=LIST macro, depending on the options chosen by the resource manager that issues the macro.
2. The warning message is issued for existing data sets, not for new data sets. For example, if you try to create a new data set that has the same name as a generic profile that does not give you ALTER access, the create fails if this profile does not have WARNING set on. If, however, WARNING is on in the generic profile, the data set allocation proceeds but no warning message is issued.

## Establishing ownership structures

---

RACF provides enough flexibility so that in most cases your RACF ownership structures can correspond to your existing installation management and organizational structures. However, this flexibility does not mean that some realignment of the organizational structures might not be advantageous from the security standpoint.

In any event, you should subdivide the ownership structures to minimize both occasions when data needs to be passed between groups, and occasions when exceptional access controls are required. If you define groups so that all users in a group share common access requirements, your administrative task of authorizing users is greatly simplified.

## Selecting user IDs and group names

In your installation it might be enough for you to isolate development work from production. On the other hand, it might be more practical for you to define many individual users and groups. In either case, you should take a look at what already exists and modify RACF to adapt to the current environment. For example, do any or all of the system users already have user IDs? If so, perhaps you can make use of them. For example, every data set name has its owner's user ID as its high-level qualifier by default.

**Batch Users:** Batch users might not already have user IDs. Here, you might consider assigning user IDs based on personnel number or, if appropriate, group name. If it is not clear what to use as a user ID, start by considering group names. Again, examine what already exists:

1. Is there an existing organizational structure that has groups with suitable abbreviations? Can the existing structure be used as is, or modified to suit?
2. What conventions already exist in job statements? It is common for the first few characters of the job names to be meaningful in terms of an application name, project, department, or some other such functional group. Could these be used as group names, or even a user ID? Are there any other fields in the job statement (for example, the account number or programmer name) that could be used? That is, could you determine from a job statement to whom or to what functional group the job belongs? (Note: The ability to derive a user ID or group from existing job statement information can be a significant migration aid. It could help you avoid the administrative effort of adding the USER= operands to existing job statements.)
3. Look at data set names to determine the local naming conventions for data sets. Can you determine to which functional group a data set belongs by looking at the name? Can you say "This is an IMS database," or "This data set belongs to the payroll group"?

It is likely that several naming conventions already exist. RACF options enable you to handle most existing variations.

Whatever you choose, consider carefully the longer term security objectives. Adding new groups and users to an existing structure presents few administrative problems. Even deleting users and groups can be done without much difficulty. However, a major reassignment of user IDs and group names, although possible, is best avoided by careful initial selection.

## Establishing your RACF group structure

You should map your groups to your organization's structure and arrange them hierarchically so that each group is a subgroup of some other group. The group SYS1 is predefined as the highest group in the hierarchy. You should document the resulting group structure as part of the implementation plan. You might want to develop a set of guidelines for your delegated security and group administrators to identify the general categories of resources and users, and the relationships between them.

For groups that might become large, and for which a quick listing of members is not needed, you might want to consider defining the groups using the UNIVERSAL operand of the ADDGROUP command. See [“Defining large groups with the UNIVERSAL attribute” on page 88](#).

[Figure 4 on page 40](#) shows relationships that can exist between users and groups.



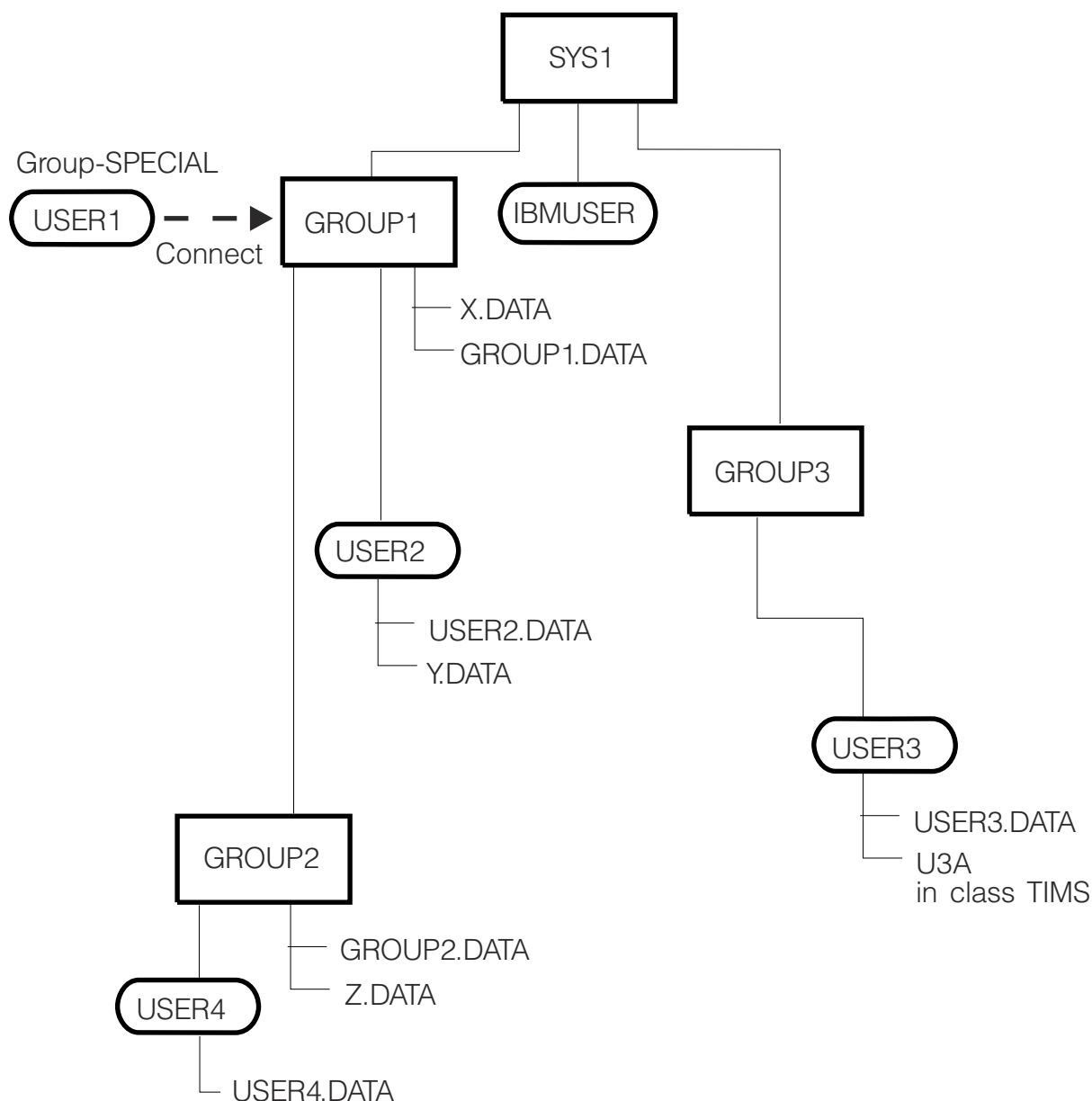


Figure 4. User and group relationships

In [Figure 4](#) on page 40:

- The users' default groups are their owning groups.
- Groups X and Y exist and are owned by GROUP1.
- Group Z exists and is owned by GROUP2.
- The highest level group, SYS1, owns subgroups GROUP1 and GROUP3 and the user IBMUSER.
- GROUP1 owns subgroup GROUP2 and the users USER1 and USER2.
- USER1 is connected to GROUP1 with group-SPECIAL authority. This gives USER1 (who is a RACF administrator) control over GROUP1's resources and resources in GROUP1's scope, but not over GROUP3's resources.

**Note:** If you run with list-of-groups checking inactive (that is, with the SETROPTS NOGRPLIST option in effect), the scope of USER1's group-SPECIAL attribute is limited to his default group or the group he specified when logging on, and the groups below that group in the hierarchy. For more information on list-of-groups checking, see [“Activating list-of-groups checking \(GRPLIST option\)”](#) on page 111.



## Educating the system users

Part of your job is to tell the system users what they need to know to work without disruption when RACF is installed.

The amount of detailed information that each user needs to know about RACF depends on the RACF functions that you authorize the person to use. Here are some examples of information that various system users typically require:

**All System Users:** All users who are defined to RACF must know:

- How to identify themselves to the system with their user ID and password or password phrase, and how to change their password or password phrase. They should also be aware of the significance of their password or password phrase to system security.
- If list-of-groups processing is *not* in effect, how to log on to a group other than their default group.

**Note:** Users can use the LISTUSER command to find out the groups to which they belong.

- If security labels are used on your system, how to log on with a security label other than their default security label. For more information, see [“Understanding security labels” on page 98](#).

**Note:** To find out what security labels they can use, users can enter:

```
SEARCH CLASS(SECLABEL)
```

- If you want them to be able to reduce their change intervals (for passwords and password phrases), how to use the PASSWORD (or PHRASE) command.
- How to use the LISTUSER command to list their own profile information.

**Users of RRSF functions:** RRSF users need to understand RRSF network concepts and know RRSF node names. Depending on your security plan, some RRSF users might also need to know how to:

- Direct commands
- Synchronize passwords
- Establish and approve user ID associations using the RACLINK command.

**Users who RACF-protect general resources:** Depending on your security plan, users might work with profiles in the TAPEVOL, JESSPOOL, or other general resource classes. These users must know:

- How to define and modify profiles in the general resource class, including whether generic profiles are allowed in the class
- What user IDs and group names they can use when giving access to the profiles
- The meaning of the access authorities (such as NONE, READ, and WRITE) in the general resource class
- What your installation's security policy is towards specific security enhancements like security levels, categories, and security labels

In addition to the education needed for administrators who are using generic profiles, even more education is required on generic profiles for those who are switching to enhanced generic naming (that is, from the SETROPTS NOEGN to the SETROPTS EGN option).

For more information, see [“Defining profiles for general resources” on page 185](#) and the topics of this document that describe how to use the class.

**Technical support personnel:** Users who install the RACF component of the Security Server must be familiar with migration planning considerations and the steps that are required to install or reinstall RACF. For complete RACF information, see all of the following z/OS documents:

- [z/OS Upgrade Workflow](#)
- [z/OS Planning for Installation](#)
- [z/OS Release Upgrade Reference Summary](#).

Users who maintain the RACF database must be familiar with the RACF utilities, which are described in *z/OS Security Server RACF System Programmer's Guide*.

**Group administrators:** Group administrators either have one of the group authorities, have a group attribute (such as group-SPECIAL), or own group resources. These users need to use the information in this document and *z/OS Security Server RACF Command Language Reference*.

**RACF auditors:** Users with the AUDITOR or ROAUDIT attribute should see *z/OS Security Server RACF Auditor's Guide* for information on using RACF for auditing.

Note that if ISPF and TSO/E are installed, the user can use the RACF ISPF panels to perform most of the same functions as the RACF commands. Using the RACF ISPF panels frees users from the need to know the details of command syntax. (The ISPF panels cannot be used to activate or deactivate mixed-case passwords.)

**Note:** You can ask a user with the AUDITOR attribute to issue the SETROPTS command with the CMDVIOL operand. This causes RACF to log all of the RACF command violations that it detects. The auditor can then use the RACF report writer to produce a printed audit trail of command violations. From the report, you can determine how many command violations are occurring and which users are causing the violations. A significant number of command violations, especially when RACF is first installed, might mean users need more education. The report can also help you to identify any specific users who are persistently trying to alter profiles without the proper authority.

*z/OS Security Server RACF Command Language Reference* contains detailed information on the RACF commands used.

**Programmers writing unauthorized applications:** Programmers writing unauthorized applications can use the RACROUTE macro to request many security-related services, including controlling access to protected resources (RACROUTE REQUEST=AUTH).

**Note:** Your installation can create installation-defined resource classes. If your installation creates profiles in those classes, an application can issue a RACROUTE REQUEST=AUTH to check if a user has sufficient authority to complete a user action. How much authority is needed for any particular user action is defined by the way in which the application invokes the RACROUTE REQUEST=AUTH macro. For more information on creating installation-defined classes, see *z/OS Security Server RACF System Programmer's Guide*.

**Programmers writing authorized applications:** Programmers writing authorized applications (that is, APF-authorized programs) can use the RACROUTE macro to request security-related services, including:

- Identifying and verifying users (RACROUTE REQUEST=VERIFY)
- Replacing or retrieving fields in RACF profiles (RACROUTE REQUEST=EXTRACT)

For more information on using the RACROUTE macro, see *z/OS Security Server RACROUTE Macro Reference*.

## Summary

---

As an overall strategy in organizing for RACF implementation, the implementation team should strive for a policy of security by evolution, rather than revolution. Wherever transparency can be used, it should be. In some cases, you must actively solicit management support.

You should examine organizational structures to establish the most efficient profile ownership structures, educate users with the level of information they need to perform their assigned functions, and prepare guidelines for the various administrators.

Finally, you and the implementation team should prepare an implementation plan to guide the work of the team. [Table 5 on page 43](#) provides a checklist for the implementation team to use while preparing the implementation plan. Note that this checklist represents only a starting point; it is not meant to be exhaustive.

Table 5. Checklist for implementation team activities

Item	Comments
<b>Objectives</b>	<ul style="list-style-type: none"> <li>• What are the installation's security objectives?</li> <li>• Over what time frame are they to be achieved?</li> <li>• Is the position of management clear on all objectives?</li> <li>• Is the statement of security policy clear and complete for all objectives?</li> </ul>
<b>Protection</b>	<ul style="list-style-type: none"> <li>• What resource classes are to be protected?</li> <li>• What resources within these classes are to be protected?</li> <li>• Can protection be phased in?</li> <li>• Which resources must be protected, and when?</li> </ul>
<b>Naming conventions</b>	<ul style="list-style-type: none"> <li>• What installation data set or general resource naming conventions already exist?</li> <li>• Are changes necessary?</li> <li>• Does implementing RACF provide an opportunity to enforce naming conventions?</li> <li>• If so, can they be enforced across the entire installation or only over a subset of the installation?</li> <li>• Immediately or eventually?</li> </ul>
<b>Organization</b>	<ul style="list-style-type: none"> <li>• Can the definition of RACF groups (and their associated users) be mapped to the existing organizational structure?</li> <li>• What changes to the organizational structure, if any, are necessary?</li> <li>• How is RACF to be controlled and administered?</li> <li>• Which functions are to be retained centrally?</li> <li>• Which functions are to be delegated, wholly or in part?</li> <li>• Which users should have what RACF attributes?</li> </ul>
<b>User and group names</b>	<ul style="list-style-type: none"> <li>• What names are to be established for groups and user IDs?</li> <li>• Which groups and users are to be defined to RACF?</li> <li>• Which user verification technique is to be used?</li> </ul>
<b>Transparency</b>	<ul style="list-style-type: none"> <li>• Try to make RACF transparent to your users wherever possible.</li> <li>• Which resources can be protected by generic profiles?</li> <li>• Which resources require discrete profiles?</li> <li>• Which users and groups should be placed in the access lists, and with what access authorities?</li> <li>• What deviations from strict user accountability are to be allowed, and for how long?</li> </ul>
<b>RACF tailoring</b>	<ul style="list-style-type: none"> <li>• Which RACF exits are to be used, if any, and under what conditions?</li> </ul>
<b>Authorizations</b>	<ul style="list-style-type: none"> <li>• What authorizations are required for the program properties table (PPT), APF libraries, and similar items?</li> </ul>
<b>Recovery</b>	<ul style="list-style-type: none"> <li>• What recovery procedures must be established?</li> </ul>
<b>Violation procedures</b>	<ul style="list-style-type: none"> <li>• What security procedures for logging, reporting, and auditing must be established?</li> </ul>
<b>Subsystems</b>	<ul style="list-style-type: none"> <li>• What are the security requirements for IMS, CICS, and other subsystems?</li> </ul>
<b>Storage Management Subsystem (SMS)</b>	<ul style="list-style-type: none"> <li>• Is your data managed by SMS?</li> <li>• If it is, what is required for your SMS constructs, application IDs, and data set owners?</li> </ul>
<b>Test plan</b>	<ul style="list-style-type: none"> <li>• What is the plan for testing the RACF implementation?</li> </ul>

Table 5. Checklist for implementation team activities (continued)

Item	Comments
<b>Education</b>	<ul style="list-style-type: none"><li>• What is the plan for preparing user documentation and other educational material?</li><li>• Should there be a newsletter for most users and more detailed education for group administrators?</li></ul>
<b>Install RACF</b>	<ul style="list-style-type: none"><li>• What RACF options are to be used?</li><li>• What is the plan for installing RACF?</li></ul>
<b>Monitor</b>	<ul style="list-style-type: none"><li>• After beginning to define groups, users, generic profiles, and data for a pilot group, how will progress against your implementation plan be monitored?</li><li>• What procedures will be established to ensure that future applications receive the appropriate security considerations?</li></ul>

---

## Chapter 3. Defining users

As a general objective, all users should be defined to RACF. Users who are not defined to RACF can use the system virtually unimpeded, unless, of course, they attempt to access data to which they are unauthorized.

The users you must initially define are those you have selected for the pilot project and the central core of personnel who maintain and operate the system itself. Other users can then be defined as determined by convenience and the priority of their security needs.

You should consider defining the following users to RACF:

- Interactive users of CICS, IMS, TSO/E, NetView, or other products that support logging on at a terminal.
- z/OS UNIX users. You use RACF commands to define users to z/OS UNIX. The z/OS UNIX attributes are kept in the OMVS segment of the user's profile and can be specified in addition to any existing attributes. The new attributes extend the user's capabilities to include the use of z/OS UNIX functions. In order to use z/OS UNIX services, a user must have z/OS UNIX attributes defined, such as an z/OS UNIX user identifier (UID) in his or her user profile and a z/OS UNIX group identifier (GID) in the group profile of his or her current connect group (the user's default group or the one specified on the TSO LOGON screen or job card). For more information, see [Chapter 21, "RACF and z/OS UNIX," on page 513](#).
- Users who submit batch jobs without first logging on to a terminal (such as through a physical card reader).
- MVS or JES system operators. You should work with your MVS or JES system programmer to determine which MVS and JES system operators should be defined to RACF. For more information, see ["Defining and grouping operators" on page 454](#).
- Started procedures.
- Node names in an NJE network.
- RJP or RJE remote workstations or nodes.
- Console IDs if LOGON(AUTO) is specified in the CONSOLxx member of SYS1.PARMLIB. For more information, see [z/OS MVS Initialization and Tuning Reference](#).

There are some advantages in defining all users to RACF:

- Defining all users provides for better administrative control over who is using the system. This in turn can reduce misuse of system resources.
- Attempted violations by undefined users are difficult to investigate, because they do not have user IDs that are associated with real persons or processes.

Whether all users are eventually defined to RACF is your decision. You might deem individual accountability for a certain segment of the user population unnecessary in some cases. Note that this can reduce your ability to determine exactly who took security-relevant actions.

---

### User profiles

When you define a user to RACF, you create a user profile in the RACF database. A user profile consists of a base segment and, optionally, any of the following segments: CICS, CSDATA, DCE, DFP, KERB, LANGUAGE, LNOTES, NDS, NETVIEW, OMVS, OPERPARM, OVM, PROXY, TSO, and WORKATTR.

Each segment of a user profile consists of fields. When you define a user's profile (using the ADDUSER command) or change a user's profile (using the ALTUSER command), you can specify the information contained in each field of each segment of the profile.

To define or change information in a non-base segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking.

To list the contents of a user profile or the contents of individual segments of the user profile, use the LISTUSER command.

To display the information in a non-base segment of a user profile, including your own, you must have the SPECIAL, AUDITOR, or ROAUDIT attribute or at least READ authority to the segment through field-level access checking.

**Guideline:** Use field-level access control to let users view, and optionally modify, some or all of the information in the non-base segments of their user profiles.

For more information, see [“Field-level access checking” on page 200](#), [“Controlling access to the DFP segment” on page 501](#), and [“Field-level access checking for TSO” on page 510](#).

When you use the RACDCERT command to add a certificate definition and associate it with a specified RACF-defined user ID, information about the definition is added to the user profile. To see the certificate definitions, enter:

```
RACDCERT LIST
```

To issue this command, you must have one of the following authorities:

- The SPECIAL attribute
- Sufficient authority to resource IRR.DIGTCERT.LIST in the FACILITY class:
  - READ access to IRR.DIGTCERT.LIST to list this information for yourself
  - UPDATE access to IRR.DIGTCERT.LIST to list this information for others

When you use the RACDCERT command to add a certificate name filter and associate it with a specified RACF-defined user ID, information about the definition is added to the user profile. To see the certificate name filter definitions, enter:

```
RACDCERT LISTMAP
```

To issue this command, you must have one of the following authorities:

- The SPECIAL attribute
- Sufficient authority to resource IRR.DIGTCERT.LISTMAP in the FACILITY class:
  - READ access to IRR.DIGTCERT.LISTMAP to list this information for yourself
  - UPDATE access to IRR.DIGTCERT.LISTMAP to list this information for others

When you use the RACDCERT command to add a certificate key ring and associate it with a specified RACF-defined user ID, information about the definition is added to the user profile. To see the ring definitions, enter:

```
RACDCERT LISTRING
```

To issue this command, you must have one of the following authorities:

- The SPECIAL attribute
- Sufficient authority to resource IRR.DIGTCERT.LISTRING in the FACILITY class:
  - READ access to IRR.DIGTCERT.LISTRING to list this information for yourself
  - UPDATE access to IRR.DIGTCERT.LISTRING to list this information for others

When you use the RACLINK command to establish a user ID association, information about the association is added to the user profile. To see the user ID associations, enter:

```
RACLINK LIST
```

For more information on how to use the ADDUSER, ALTUSER, LISTUSER, RACDCERT, and RACLINK commands, see [z/OS Security Server RACF Command Language Reference](#).

## The base segment in user profiles

The base segment of a user profile contains basic information that is needed to define a user to RACF. You can specify the following information in the base segment:

### **USERID**

User's identification

### **NAME**

User's name

### **OWNER**

Owner of the user's profile

### **DFLTGRP**

User's default group

### **AUTHORITY**

User's authority in the default group

### **PASSWORD**

User's password

### **NOPASSWORD**

Indicates that the user cannot enter the system using a password and when the user also has the NOPHRASE and NOIDCARD attributes, gives the user the PROTECTED attribute

### **PHRASE**

User's password phrase

### **NOPHRASE**

Indicates that the user cannot enter the system using a password phrase and when the user also has the NOPASSWORD and NOIDCARD attributes, gives the user the PROTECTED attribute

### **REVOKE**

Date on which RACF prevents the user from having access to the system

### **RESUME**

Date on which RACF lets the user have access to the system again

### **NEVERCONTAIN**

User ID is exempt from containment.

### **UACC**

Default universal access authority for resources that the user defines

### **WHEN**

Days of the week and hours of the day during which the user has access to the system

### **ADDCATEGORY**

User's installation-defined security category

### **SECLEVEL**

User's installation-defined security level

### **CLAUTH**

Classes in which the user can define profiles

### **SPECIAL**

Gives the user the system-wide SPECIAL attribute

### **AUDITOR**

Gives the user the system-wide AUDITOR attribute

### **OPERATIONS**

Gives the user the system-wide OPERATIONS attribute

### **DATA**

Installation-defined data

**ADSP**

Indicates that all permanent data sets the user creates are to be RACF-protected with discrete profiles

**GRPACC**

Indicates that other group members can have access to any group data set the user protects with a data set profile

**MODEL**

Name of the data set model profile to be used when creating new data set profiles, either generic or discrete

**OIDCARD**

Indicates that the user must supply an operation ID card when logging on to the system

**RESTRICTED**

Indicates that global access checking, the ID(\*) entry on the access list, and the UACC will not be used to allow this user access to a protected resource.

To prevent a restricted user from gaining access to a z/OS UNIX file system resource unless specifically authorized, see [“Controlling access to file system resources for restricted users” on page 531](#).

**ROAUDIT**

Gives the user the system-wide ROAUDIT attribute

**SECLABEL**

User's default security label

**CERTNAME**

The names of the profiles in the DIGTCERT class that are associated with this RACF user ID

**CERTLABL**

The certificate labels for the profiles in the DIGTCERT class that are associated with this RACF user ID

**CERTPUBK**

The public key associated with a public key certificate. This is the BER-encoded public key as specified in the certificate.

**CERTSJDN**

The subject name of the entity to whom the certificate is issued. This is the BER-encoded format of the subject's distinguished name as contained in the certificate.

**Note:** You can only add or delete the data in the CERTNAME, CERTLABL, CERTPUBK and CERTSJDN fields by using the RACDCERT command. The ADDUSER or ALTUSER commands have no effect on these fields.

**NMAPNAME**

The names of the profiles in the DIGTNMAP class containing certificate name filters that are associated with this RACF user ID

**NMAPLABL**

The labels for the certificate name filters that are associated with this RACF user ID

See [z/OS Security Server RACF Command Language Reference](#) for information about the authorization required to create, change, or view information in the base segment.

## The CICS segment in user profiles

You can specify information for CICS terminal operators in RACF user profiles. This information is used when CICS terminal operators sign on to CICS.

For CICS and RACF planning information, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).

An installation can set the default characteristics (including authorities) for CICS terminal operators by defining CICS segments in the user profiles of these users. To do this, issue the ADDUSER or ALTUSER command with the CICS operand. You can specify the following information:



**OPCLASS**

Classes assigned to this operator to which basic mapping support (BMS) messages are to be routed

**OPIDENT**

Identification of the operator for use by BMS

**OPPRTY**

Priority of the operator

**RSLKEY**

Resource security level (RSL) keys assigned to this user

**TIMEOUT**

Time that the operator is allowed to be idle before being signed off

**TSLKEY**

Transaction security level (TSL) keys assigned to this user

**XRFSOFF**

Indicates whether the operator is to be signed off by CICS when an XRF takeover occurs

To define or change information in the CICS segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the CICS segment of a user profile, you must have the SPECIAL attribute or at least READ authority to the segment through field-level access checking. For more information, see [“Field-level access checking”](#) on page 200.

## The CSDATA segment in user profiles

You can define a CSDATA segment for user profiles. The CSDATA segment contains installation-defined data related to custom fields that your installation has defined. For details about defining custom fields, see [Chapter 26, “Defining and using custom fields,”](#) on page 637.

To define or change information in the CSDATA segment of a user profile, you must have the SPECIAL attribute or at least UPDATE authority to the segment by way of field-level access control. For details, see [“Authorizing users to update data in a custom field”](#) on page 645. To display information in the CSDATA segment of a user profile, you must have the SPECIAL attribute or at least READ authority to the segment by way of field-level access control.

## The DCE segment in user profiles

The DCE segment, defined to the RACF user profile, associates a DCE principal with the RACF user profile.

You can specify the following attributes:

**DCENAME**

User's DCE principal name

**UUID**

User's DCE principal universal unique identifier

**HOMECELL**

Home cell for this DCE user

**HOMEUUID**

Home cell universal unique identifier

**AUTOLOGIN**

Single signon processing (YES or NO)

As RACF administrator, you need to work with the DCE administrator to define RACF profiles to use these features correctly.

## The DFP segment in user profiles

You can define a DFP segment for user profiles. The DFP segment contains default values that DFP uses to determine data management and DASD storage characteristics for user data sets.

You can specify the following information in this segment:

**DATAAPPL**

User's DFP data application identifier

**DATACLAS**

User's default data class for attributes used during allocation of all new data sets

**MGMTCLAS**

User's default management class for attributes used in managing a data set after it is allocated

**STORCLAS**

User's default storage class for logical storage attributes

To define or change information in the DFP segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the DFP segment of a user profile, you must have the SPECIAL attribute, the AUDITOR or ROAUDIT attribute, or at least READ authority to the segment through field-level access checking. For more information, see [“Controlling access to the DFP segment” on page 501](#).

## The KERB segment in user profiles

You can define a KERB segment for user profiles. The KERB segment contains the information about a z/OS Network Authentication Service user, such as the local principal name that will be mapped to this RACF user ID.

You can specify the following information in this segment:

**ENCRYPT**

User's key encryption types

**KERBNAME**

User's local principal name

**MAXTKTLFE**

User's maximum ticket life

To define or change information in the KERB segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the KERB segment of a user profile, you must have the SPECIAL attribute, the AUDITOR or ROAUDIT attribute, or at least READ authority to the segment through field-level access checking. For more information, see [z/OS Integrated Security Services Network Authentication Service Administration](#).

## The LANGUAGE segment in user profiles

You can specify a user's preferred national languages in the LANGUAGE segment of the user's user profile. The languages stored in the user profiles are used by TSO, CICS, and any other applications that use the RACROUTE REQUEST=EXTRACT macro to determine a user's preferred national languages. For specific information on how an application checks for a user's preferred national languages, see the documentation for that product.

**Note:** In general, you should also specify an installation default using the LANGUAGE operand of the SETROPTS command.

If individual users prefer languages other than the installation defaults, use the LANGUAGE operand of the ADDUSER or ALTUSER command to assign the preferred languages. On the LANGUAGE operand, you can specify the following information:

**PRIMARY**

User's preferred national language, if different from the installation default

**SECONDARY**

User's alternate national language, if different from the installation default

To define or change information in the LANGUAGE segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the LANGUAGE segment of a user profile, you must have the SPECIAL attribute or at least READ authority to the segment through field-level access checking. For more information, see [“Field-level access checking” on page 200](#).

## The LNOTES segment in user profiles

You can define an LNOTES segment for user profiles. The LNOTES segment contains the Lotus Notes for z/OS short name that will be mapped to this RACF user ID.

You can specify the following information in this segment:

### **SNAME**

User's short name for use with Lotus Notes for z/OS

To define or change information in the LNOTES segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the LNOTES segment of a user profile, you must have the SPECIAL attribute, the AUDITOR or ROAUDIT attribute, or at least READ authority to the segment through field-level access checking. For more information, see [“RACF support for NDS and Lotus Notes for z/OS” on page 251](#).

## The NDS segment in user profiles

You can define an NDS segment for user profiles. The NDS segment contains the Novell Directory Services for OS/390 user name that will be mapped to this RACF user ID.

You can specify the following information in this segment:

### **UNAME**

User's user name for use with Novell Directory Services for OS/390

To define or change information in the NDS segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the NDS segment of a user profile, you must have the SPECIAL, AUDITOR, or ROAUDIT attributes, or at least READ authority to the segment through field-level access checking. For more information, see [“RACF support for NDS and Lotus Notes for z/OS” on page 251](#).

## The NETVIEW segment in user profiles

When you define a new NetView operator or change NETVIEW attributes for an existing operator, you can specify the following information in the NETVIEW segment of the user's profile:

### **CONSNAME**

MCS console identifier

### **CTL**

Specifies GLOBAL, GENERAL, or SPECIFIC control

### **DOMAINS**

Domain identifier

### **IC**

Initial command or list of commands to be executed by NetView when this NetView operator logs on

### **MSGRECV**

Indicates whether the operator will receive unsolicited messages

### **NGMFADMN**

Indicates whether this operator can use the NetView graphic monitor facility

### **OPCLASS**

Class of the operator

## The OMVS segment in user profiles

When you define a new z/OS UNIX user or change z/OS UNIX attributes for an existing user, you can specify the following information in the OMVS segment of the user's profile:

### **ASSIZEMAX**

User's z/OS UNIX RLIMIT\_AS (maximum address space size)

### **CPUTIMEMAX**

User's z/OS UNIX RLIMIT\_CPU (maximum CPU time)

### **FILEPROCMAX**

User's z/OS UNIX maximum number of files per process

### **HOME**

User's z/OS UNIX initial directory path name

### **MEMLIMIT**

User's z/OS UNIX non-shared memory size

### **MMAPAREAMAX**

User's z/OS UNIX maximum memory map size

### **PROCUSERMAX**

User's z/OS UNIX maximum number of processes per UID

### **PROGRAM**

User's z/OS UNIX program path name, such as a default shell program

### **SHMEMMAX**

User's z/OS UNIX maximum shared memory size

### **THREADSMAX**

User's z/OS UNIX maximum number of threads per process

### **UID**

User's z/OS UNIX user identifier

To define or change information in the OMVS segment of a user profile, including one's own, you must have the SPECIAL attribute (to view or change it), the AUDITOR or ROAUDIT attribute (to view it), or sufficient authority to the OMVS segment fields through field-level access checking. Many installations allow users to view all of their OMVS information and to update selected fields, such as the home directory or default program. (Note that specifying a given path name in either of these fields does not grant users access to the path name; users still need the appropriate file system permission to access the path.)

**Guideline:** Avoid allowing users to update their UID or the resource limit fields.

To permit users to access all fields that are not protected by a more specific profile, define the USER.OMVS.\* profile in the FIELD class. For example, to permit all users to view their own OMVS information, permit &RACUID with READ access to the USER.OMVS.\* profile. To allow authorized administrators who need to change the OMVS information in others' profiles, permit them with UPDATE access. You can define more specific profiles to address special requirements. For example, you might define the USER.OMVS.HOME and USER.OMVS.PROGRAM profiles, authorizing &RACUID with UPDATE authority. You might also need to permit UPDATE access for administrators because the access list of a more specific profile will override that of a less specific profile.

For more information, see [“Defining user identifiers \(UIDs\)” on page 514](#).

## The OPERPARM segment in user profiles

Users can enter an extended MCS console session as follows:

- If the application they use executes the MCSOPER macro (an authorized macro)
- If the user issues the TSO CONSOLE command

For information on using RACF to control the users who can establish an extended MCS console session, see [z/OS MVS Planning: Operations](#).

Users who enter an extended MCS console session can act as system operators. An installation can set the default characteristics (including command authorities) for these sessions by defining OPERPARM segments in the user profiles of these users. To do this, issue the ADDUSER or ALTUSER command with the OPERPARM operand. You can specify the following information:

**ALTGRP**

Alternative console group for recovery. Ignored when each system sharing the RACF database runs z/OS Version 1 Release 8 or higher.

**AUTH**

Operator's command authority

**CMDSYS**

Name of the system to which the operator is connected for command processing

**DOM**

Indicates whether the operator should receive delete operator message requests

**HC**

Specifies whether this console is to receive all messages that are directed to hardcopy. Note that route codes specified for a console do not apply to hardcopy messages, so this console will receive all hardcopy messages regardless of route code.

**INTIDS**

Indicates whether or not a console should receive messages directed to console ID zero (the internal console)

**KEY**

Indicates a data retrieval key used to search for operator consoles using the DISPLAY CONSOLES command

**LEVEL**

Message level that the operator receives

**LOGCMDRESP**

Indicates whether command responses received by the operator are to be recorded on the hardcopy log

**MFORM**

Format in which messages are displayed

**MIGID**

Indicates whether the operator is to receive a migration console ID. Ignored when each system sharing the RACF database runs z/OS Version 1 Release 8 or higher.

**MONITOR**

Events that the operator can monitor

**MSCOPE**

Name of the system from which the operator receives unsolicited messages

**ROUTECD**

Routing codes that the operator receives

**STORAGE**

Maximum amount of virtual storage in megabytes for message queuing

**UD**

Indicates whether the operator should receive messages that are considered undeliverable. Ignored when each system sharing the RACF database runs z/OS Version 1 Release 8 or higher.

**UNKNIDS**

Indicates whether a console is to receive messages directed to unknown console IDs

To define or change information in the OPERPARM segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the OPERPARM segment of a user profile, you must have the SPECIAL attribute or at least READ authority to the segment through field-level access checking. For more information, see [“Field-level access checking”](#) on page 200.

## The PROXY segment in user profiles

The PROXY segment is intended for use with user IDs assigned to application servers. It is not intended for use with user IDs assigned to users, such as TSO or CICS users. Therefore, it is not described here. For information about the PROXY segment, see the PROXY parameter of the ALTUSER command in [z/OS Security Server RACF Command Language Reference](#).

## The TSO segment in user profiles

When you define a new TSO user or change TSO attributes for an existing user, you can specify the following information in the TSO segment of a user's profile:

### **ACCTNUM**

User's default account number

### **COMMAND**

Command to be run during TSO/E logon

### **JOBCLASS**

Default value for user's job class

### **MSGCLASS**

Default value for the user's message class

### **HOLDCLASS**

Default value for the user's hold class

### **SYSOUTCLASS**

Destination ID for the user's SYSOUT data sets

### **PROC**

User's default logon procedure

### **MAXSIZE**

User's maximum region size

### **SIZE**

User's default region size

### **SECLABEL**

Security label specified when the user previously logged on to TSO

### **UNIT**

Default device used for allocations

### **USERDATA**

Optional user data

If a user logs on to TSO and you have defined a TSO segment in the user's profile, TSO checks the user's authority to use certain TSO resources such as account numbers and logon procedures. If the user is authorized to use a resource such as an account number, TSO continues building a session for the user. Otherwise, TSO prompts the user for a valid account number.

If a user logs on to TSO and you have not defined a TSO segment for that user, TSO checks the SYS1.UADS data set for the information it needs to build a session. If TSO does not find an entry for the user in SYS1.UADS, the user is denied access to the system.

You can move TSO user attribute information from SYS1.UADS to the RACF database. (SYS1.UADS contains an entry for each TSO user that describes the attributes that regulate the user's access to the system.) When you move this TSO information into the RACF database, it is stored in the TSO segment of the user's profile. When a user logs on to TSO, it uses the information contained in the TSO segment to build a session for the user.

Moving the TSO user information to the RACF database eliminates the need to maintain an entry in SYS1.UADS for each TSO user. However, you *must* maintain entries in SYS1.UADS for certain users, such as IBMUSER and system programmers. For example, if you need to deactivate RACF to perform maintenance on the RACF database, users authorized to perform this maintenance must be able to log

on to the system. When RACF is inactive, TSO checks entries in SYS1.UADS to authorize access to the system.

**Note:**

1. You can use the RACONVRT EXEC to help convert SYS1.UADS entries to RACF user profiles. See [z/OS TSO/E Customization](#) for more information.
2. If you are defining TSO segments in user profiles, you must activate the following TSO general resource classes: TSOPROC and ACCTNUM. For more information, see [“Protecting TSO resources” on page 507](#).
3. **Guideline:** Use field-level access control to protect fields within the TSO segment of user profiles. Otherwise, any user can list and change the information contained in this segment. For more information, see [“Field-level access checking” on page 200](#).
4. A TSO user can use the TSO/E logon panel to specify or override certain information in the TSO segment of his or her user profile. For example, a user can change an account number, or specify an account number if one has not been specified, using the TSO/E logon panel. RACF checks the user's authorization to the ACCTNUM profile that protects the specified account number. If the user is authorized to use the specified account number, TSO stores the account number in the TSO segment of the user's profile and uses it as a default value the next time the user logs on to TSO. Otherwise, RACF denies access to the account number.

If users attempt to change their user profiles when logging on, the logon is allowed but the TSO segment is not updated in either of the following cases:

- The RACF database is *locked*.
- The system is enabled for sysplex communication and RACF is in read-only mode.

See [z/OS TSO/E User's Guide](#) for a description of the information that a user can specify on the TSO/E logon panel.

5. A TSO installation can write a TSO logon pre-prompt exit to bypass checking SYS1.UADS for user attribute information. See [z/OS TSO/E Customization](#) for more information.

## The WORKATTR segment in user profiles

You can specify work attribute (WORKATTR) segments in user profiles. WORKATTR segments include information such as e-mail, SYSOUT, and account information for the users.

An installation can set the default characteristics (including authorities) for these users by defining WORKATTR segments in the user profiles of these users. To do this, issue the ADDUSER or ALTUSER command with the WORKATTR operand.

You can specify the following information in the WORKATTR segment of a user's profile:

**WANAME**

User name on SYSOUT

**WABLDG**

Building on SYSOUT

**WADEPT**

Department on SYSOUT

**WAROOM**

Room on SYSOUT

**WAADDR1**

SYSOUT address line 1

**WAADDR2**

SYSOUT address line 2

**WAADDR3**

SYSOUT address line 3

### WAADDR4

SYSOUT address line 4

### WAACCN

Account number

### WAEMAIL

E-mail address

To define or change information in the WORKATTR segment of a user profile, including your own, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access checking. To display information in the WORKATTR segment of a user profile, you must have the SPECIAL attribute or at least READ authority to the segment through field-level access checking. For more information, see [“Field-level access checking” on page 200](#).

## User naming conventions

---

The rules for naming users, like those for naming groups, are simple:

- A RACF user ID must be 1 - 8 characters in length, and can consist of any combination of uppercase A - Z, 0 - 9, # (X'7B'), \$ (X'5B'), or @ (X'7C').

**Note:** TSO and MVS also require that the first character of user IDs be uppercase A - Z, # (X'7B'), \$ (X'5B'), or @ (X'7C').

- The #, \$, and @ characters might be displayed differently on terminals outside the United States; therefore, use the characters with the hexadecimal equivalents shown above.
- No two user IDs can be the same. No user ID can be the same as a group name.

For information about user naming conventions for z/OS UNIX user identifiers (UIDs), see [“Defining user identifiers \(UIDs\)” on page 514](#).

## Suggestions for defining user IDs

---

Basically, there are no requirements for establishing a specific type of user ID. That is, in some installations, you might form user IDs by adding a numerical suffix to a group name (for example, ADMIN01 or MKT06). In other cases, you might use first names (for example, PETER and PAUL could be defined and connected to the RESEARCH group. In this case, if PETER subsequently leaves the RESEARCH group to join the TEST group, he need not change his user ID.)

The concept of user IDs based on group names appears practical because a quick glance at the user ID reveals the group. However, this concept might not prove so practical a few years later if many of the current users have changed groups. In addition, how does such a user handle their user data sets (data sets with the user ID as the high-level qualifier) after the user ID is changed? In the long run, user IDs based on something like user names or personnel numbers do not have this problem and offer the greatest long-term flexibility.

For suggestions related to z/OS UNIX, see [“Defining user identifiers \(UIDs\)” on page 514](#).

## Migrating existing user IDs to RACF

Where user IDs already exist in machine readable form (for example, in SYS1.UADS), a simple CLIST can provide a valuable administrative aid in migrating users to RACF.

**Note:** You can use the RACONVRT EXEC to help convert SYS1.UADS entries to RACF user profiles. For more information, see [z/OS TSO/E Customization](#).

## Creating new user IDs from scratch

Where user IDs are assigned from scratch, they can often be created in blocks, using a CLIST. For example, you could centrally create 50 user IDs, MKT01 through MKT50, and allocate them to the manager of group MKT to assign to users in the department. The default group (MKT), password, and other operands can all be preset. You should assign the REVOKE attribute to unused user IDs.



## Creating user IDs for system operators

You can create user profiles for system operators with the ADDUSER command. An operator who already has a TSO user ID, for example, could use this user ID to log on to a console without the need for creating a new user ID.

**Note:** To keep new passwords from showing up in the system log, system operators logging on to a console should change their passwords only when they log on to the system.

## Creating user IDs for RRSF users

Some users might need to use RRSF functions, including:

- Command direction
- Password synchronization
- Use of the RACLINK command to establish and approve user ID associations.

See [Chapter 17, “The RACF remote sharing facility \(RRSF\),” on page 405](#) for more information.

## Ownership of a RACF user profile

---

Each user defined to RACF has a user profile and all user profiles have another RACF user or group as the owner. The owner (or a user who is connected to the owning group and has the group-SPECIAL attribute, or someone with SPECIAL) can change, list, and delete the user's profile and also has control over the user's attributes (including the ability to prevent the user from entering the system).

For a list of the RACF commands that owners of user profiles can issue, see [Table 50 on page 704](#).

## User attributes

---

User attributes are extraordinary capabilities, limitations, or environments that can be assigned to a user either all of the time or when the user is connected to a specific group or groups. When an attribute is to apply all of the time, it is specified at the system level and is called a user attribute. When an attribute is to apply only to a specified group or groups, it is specified at the group level and is called a group-related user attribute. For example, user attributes that you specify in an ADDUSER or ALTUSER command are stored in the user's profile and are in effect regardless of the group to which the user is connected.

The user attributes are:

- SPECIAL
- AUDITOR
- ROAUDIT
- OPERATIONS
- CLAUTH
- REVOKE
- GRPACC
- ADSP
- RESTRICTED

## The SPECIAL attribute

A user who has the SPECIAL attribute can issue all RACF commands. The SPECIAL attribute gives the user full control over all of the RACF profiles in the RACF database.

The SPECIAL attribute can be delegated only by a user who has the SPECIAL attribute. It should be limited to the RACF security and group administrators. Persons having the SPECIAL attribute should be required to use operator identification cards and passwords or password phrases, and should change their passwords or password phrases often to help ensure security.

**Note:** Because any user can access an unprotected resource, users who have the SPECIAL attribute should take care to protect their own data sets, because they can contain sensitive information.

You can assign the SPECIAL attribute at the group level. When you do, the *group-SPECIAL* user has full control over all of the profiles within the scope of the group. For additional details, see [“User attributes at the group level” on page 62](#).

For a list of the RACF commands that this attribute allows users to issue, see [Table 43 on page 699](#).

## The AUDITOR attribute

A user who has the AUDITOR attribute has the authority to specify logging options on the ALTDSD, ALTUSER, RALTER, and SETROPTS commands. In addition, the auditor can list auditing information using the LISTDSD, RLIST, LISTUSER, LISTGRP, and SEARCH commands, as well as the IRRUT100 utility. The AUDITOR attribute gives the auditor control of logging to the SMF data set. Logging to SMF helps to detect changes (or attempted changes) to the RACF database and accesses (or attempted accesses) of RACF-protected resources.

The user who has the AUDITOR attribute can list all of the profile information that is available to the SPECIAL user, as well as information that is available to auditors. Note, however, that this extended listing authority does not give the auditor additional access to protected data or additional authority to change information in the RACF database.

If the DSMON program (ICHDSM00) is not defined in the PROGRAM class (it is not a controlled program), a user must have the AUDITOR or ROAUDIT attribute to run the DSMON program. (If DSMON is a controlled program, the AUDITOR attribute is not enough to run it. The user, or the user's group, must be in the access list of the DSMON profile, ICHDSM00, to run the DSMON program.)

You should assign the AUDITOR attribute only to users who are responsible for auditing RACF security controls and functions. To provide a check and balance on RACF security measures, you should give the AUDITOR attribute to security or group administrators other than those who have the SPECIAL attribute.

The AUDITOR attribute can be assigned only by a user (security or group administrator) who has the SPECIAL attribute.

**Note:** Because any user can access an unprotected resource, users who have the AUDITOR attribute should take special care to protect their own data sets, because they can contain sensitive information.

You can assign the AUDITOR attribute at the group level. When you do, the *group-AUDITOR* user's authority is limited to profiles that are within the scope of that group. For detailed information, see [“User attributes at the group level” on page 62](#).

For a list of the RACF commands that this attribute allows users to issue, see [Table 44 on page 700](#).

## The ROAUDIT attribute

A user who has the ROAUDIT attribute has the authority to list auditing information using the LISTDSD, RLIST, LISTUSER, LISTGRP, SETROPTS LIST, and SEARCH commands, as well as the IRRUT100 utility. Unlike users with the AUDITOR attribute, users with the ROAUDIT attribute are unable to specify logging options or to control logging to the SMF data set.

The user who has the ROAUDIT attribute can list all of the profile information that is available to the SPECIAL user, as well as information that is available to auditors. Note, however, that this extended listing authority does not give the auditor additional access to protected data or additional authority to change information in the RACF database.

If the DSMON program (ICHDSM00) is not defined in the PROGRAM class (it is not a controlled program), a user must have either the AUDITOR or the ROAUDIT attribute to run the DSMON program. (If DSMON is a controlled program, the ROAUDIT attribute is not enough to run it. The user, or the user's group, must be in the access list of the DSMON profile, ICHDSM00, to run the DSMON program.)

You should assign the ROAUDIT attribute only to users who are responsible for auditing RACF security controls and functions, but who are not to be responsible for establishing those security controls or functions. For example, you might give the ROAUDIT attribute to a user account created for an external

auditor to permit that person to audit the security controls and function without being able to alter those controls. To provide a check and balance on RACF security measures, you should give the ROAUDIT attribute to auditors or users other than those who have the SPECIAL or AUDITOR attribute.

The ROAUDIT attribute can be assigned only by a user (security or group administrator) who has the SPECIAL attribute.

**Note:** Because any user can access an unprotected resource, users who have the ROAUDIT attribute should take special care to protect their own data sets, because they can contain sensitive information.

For a list of the RACF commands that this attribute allows users to issue, see [Table 44 on page 700](#).

## The OPERATIONS attribute

A user who has the OPERATIONS attribute has full access authorization to all RACF-protected resources in the DATASET, DASDVOL, GDASDVOL, PSFMPL, TAPEVOL, VMBATCH, VMCMD, VMMDISK, VMNODE, and VMRDR classes, with the following exceptions:

- If users, their current connect group, or any of their connect groups (if list-of-groups checking is active) is in the access list of a resource profile, they have only the access specified in the access list. For this reason, you should plan carefully before making users who have the OPERATIONS attribute members of any group that is in the access lists of resource profiles.
- Security classification checking or security label checking can deny access.

In addition to having access authorization, an OPERATIONS user can:

- Copy, reorganize, catalog, and scratch user or group data sets.

**Note:** The OPERATIONS attribute is required if you use DFSMSdss to copy data sets that result in a DEFINE or a discrete data set profile for data sets you do not own.

- Perform input/output operations on tape volumes.
- Create or destroy labels on tape volumes through OPEN and end-of-volume operations.
- Create group data sets for groups.

An OPERATIONS user *cannot* create group data sets for groups when *both* of the following are true:

1. The user is connected to the group with less than CREATE authority
2. The user has less than ALTER access to the data set if it is protected by a generic profile

If the user has the group-OPERATIONS attribute (that is, the user is connected to a superior group with the OPERATIONS attribute), the group for which the new data set is being created must be within the scope of that superior group.

- Create user data sets. If the user has the group-OPERATIONS attribute (that is, the user is connected to a group with the OPERATIONS attribute), the high-level qualifier of the new data set must be the ID of a user who is within the scope of that group.

In addition, RACF creates a discrete profile for the user data set if the OPERATIONS user does one of the following:

- Has the automatic data set protection (ADSP) attribute
- Specifies PROTECT on the TSO ALLOCATE command that creates the data set
- Specifies PROTECT=YES or SECMODEL=*profile-name* on the JCL DD statement that creates the data set
- Define profiles for group data sets when one of the following is true:
  - The user is *not* connected to the group of the new data set. If the user has the group-OPERATIONS attribute (that is, the user is connected to a superior group with the OPERATIONS attribute), the group for which the new data set is being created must be within the scope of that superior group.
  - The user is connected to the group with at least CREATE group authority.

## Limiting the capabilities of the OPERATIONS attribute

You can limit the access to existing resources allowed by the OPERATIONS attribute in two ways:

- By placing the OPERATIONS user (or a group to which the user is connected) in the access list of sensitive resources (using the PERMIT command). The specific access authority (such as NONE or READ) takes precedence over the OPERATIONS attribute.
- By using security levels, security categories, or security labels

You can limit the ability of the OPERATIONS user to create group data sets by ensuring that both of the following are true:

1. The user is connected to the group with less than CREATE authority
2. The user has less than ALTER access to the data set if it is protected by a generic profile

**Guideline:** Because the OPERATIONS attribute can permit access to a wide range of resources, assign this attribute to a minimum number of people. Also, audit those users to whom you have assigned the OPERATIONS attribute. To do this, a user with the AUDITOR attribute must issue the following command.

```
SETROPTS OPERAUDIT
```

To reduce the number of users who have the OPERATIONS attribute at the system level (and therefore have the attribute for all resources in the system), you can assign the OPERATIONS attribute at the group level. When you do, the *group-OPERATIONS* user's authority is limited to resources within the scope of the group. For more information, see [“The scope of authority for the users with group-level attributes”](#) on page 63 and [“User attributes at the group level”](#) on page 62.

## OPERATIONS and DASDVOL authority

If a person needs to perform maintenance activities on DASD volumes, it is more efficient (for RACF processing) and better (for limiting the resources the person can access) to give the person authority to those volumes using the PERMIT command than to assign the person the OPERATIONS or group-OPERATIONS attribute. To give the person authority to those DASD volumes, define the volumes to RACF and add the person to the access list with the access authority required by the particular resource manager (such as DFSMSdss). For more information, see [“DASD volume authority”](#) on page 167.

If you use DFSMSdss, you can designate the user as a DFSMSdss storage administrator. This method has certain advantages over both OPERATIONS and DASDVOL authorization. For more information, see [“DFSMSdss storage administration”](#) on page 168.

The OPERATIONS attribute can be delegated only by a user (security or group administrator) who has the SPECIAL attribute.

For a list of the RACF commands that the OPERATIONS attribute allows users to issue, see [Table 46](#) on page 701.

## The CLAUTH (class authority) attribute

Users receive the CLAUTH attribute on a *class-by-class* basis. You cannot assign the CLAUTH attribute at the user or group level. If a user has the CLAUTH attribute in a class, or in a class that shares the same POSIT value in the class descriptor table (CDT), RACF allows the user to define profiles in that class.

The classes you can specify with CLAUTH are the USER class and any general resource class.

### Note:

1. The authority of all users to define profiles in general resource classes can be limited by issuing the SETROPTS GENERICOWNER command. For more information, see [“Restricting the creation of general resource profiles \(GENERICOWNER and ENHANCEDGENERICOWNER options\)”](#) on page 113.
2. You must activate the class for which a user has the CLAUTH attribute to enable the user to define profiles in that class.

3. A user's authority to define profiles extends to any class that has the same POSIT value in the class descriptor table (CDT). For example, if you give a user CLAUTH(TERMINAL), that user can also define profiles in class GTERMINL, because both of these classes have the same POSIT value.

For information about the POSIT values of classes in the dynamic portion of the CDT, and for general information about the CDT, see [Chapter 10, "Administering the dynamic class descriptor table \(CDT\)," on page 263](#). For the information about the POSIT values of the classes in the static CDT, see the description of the class descriptor table (CDT) in *z/OS Security Server RACF Macros and Interfaces*.

You should give the CLAUTH attribute only to those users who are responsible for defining profiles to RACF in the specified classes and in any classes with the same POSIT value.

4. A user to whom you assign the CLAUTH attribute for the USER class is authorized to define new users to RACF with the ADDUSER command, as long as the user is the owner of or has JOIN authority in the new user's default group.

The CLAUTH attribute can be delegated only by a user with the SPECIAL attribute, or by a user who has both the authority to update the user profile and the CLAUTH attribute for the class authority being delegated.

For a list of the RACF commands that the CLAUTH attribute allows users to issue, see [Table 47 on page 701](#).

## The REVOKE attribute

You can prevent a RACF user from entering the system by assigning the REVOKE attribute on the ALTUSER command. This attribute is useful when you want to prevent a user from entering the system but you cannot use the DELUSER command because the user still owns RACF resource profiles.

You can also assign the REVOKE attribute on a group level by using the CONNECT command. If the user has the REVOKE attribute for a group, the user cannot enter the system by connecting to that particular group, or access resources as a member of that group.

RACF allows you to specify a future date for a REVOKE to occur (at both the system and the group level). You can also specify a future date to remove the REVOKE attribute by using the RESUME operand on the ALTUSER command.

You can clear or delete a user's revoke date by issuing the NOREVOKE operand of the ALTUSER COMMAND.

```
ALTUSER BLIX NOREVOKE
```

Only the owner of a user's profile (or a user who has the SPECIAL attribute) can assign the REVOKE attribute.

## The GRPACC (group access) attribute

If a user has the GRPACC attribute, any group data set profiles that the user defines to RACF (through either the ADSP attribute, the PROTECT parameter on the DD statement, or the ADDSD command) are automatically made accessible to other users in the group if the user defining the profile is a member of that group. The group whose name is used as the high-level qualifier of the data set name is given UPDATE authority to the data set. Note that, if the defining user does not have the GRPACC attribute, and profile modeling is not being used, the user must use the PERMIT command to allow the group to access the group data set.

A user to whom you assign the GRPACC attribute at the *user* level has this attribute in all of the groups of which the user is a member. If a user has the GRPACC attribute at the *group* level, the attribute applies only to the group in which the user has the attribute.

You should assign the GRPACC attribute with care, especially if the RACF user to whom you are assigning the attribute is allowed to RACF-protect group data sets in several groups. This user could unintentionally authorize groups to access a group data set to which they should not have access.

Only the owner of a user's profile (or a user who has the SPECIAL attribute) can assign the GRPACC attribute.

**Tips:**

1. The use of automatic modeling (for example, the MODEL operand in user and group profiles) provides more flexibility than the GRPACC attribute.
2. You can provide more flexible coverage for all users, in some resource classes, by using appropriate &RACGPID entries in the global access checking table. For more information, see [Table 17 on page 198](#).

## The ADSP (automatic data set protection) attribute

When a user has the ADSP attribute, RACF always automatically creates a discrete profile every time the user defines a permanent DASD or tape data set. (For tape data sets, the TAPEDSN and TAPEVOL options must be active.)

You can assign ADSP at the group level using the CONNECT command. If assigned at the group level, ADSP is in effect only when that group is the user's current connect group.

If generic profile checking is active, you should consider removing the user's ADSP attribute. You can do this on a user-by-user basis with the ALTUSER command, or for an entire installation by using the NOADSP operand on the SETROPTS command.

A data set created under ADSP is accessible only to the user who created it, unless other users or groups are added to the access list (such as through the PERMIT command, the GRPACC user attribute, or modeling), or if global access checking allows the access.

Only the owner of a user's profile (or a user who has the SPECIAL attribute) has control over the ADSP attribute.



**Attention:** A DASD data set is defined to RACF at allocation. If the data set disposition is changed at deallocation (through dynamic deallocation), the change is *not* reflected in the RACF database. For example, if the data set disposition is DELETE at allocation and KEEP at deallocation, the data set is not automatically RACF-protected. However, RACF performs generic profile checking if you have activated this option for the DATASET class by specifying GENERIC(DATASET) on the SETROPTS command.

## The RESTRICTED attribute

You can prevent RACF users from gaining access to protected resources they are not specifically authorized to access by assigning the RESTRICTED attribute on the ADDUSER or ALTUSER command. See [“Defining restricted user IDs” on page 75](#) for more information.

Only the owner of a user's profile (or a user who has the SPECIAL attribute) can assign the RESTRICTED attribute.

## User attributes at the group level

---

You can specify the SPECIAL, AUDITOR, and OPERATIONS user attributes at the group level by using the CONNECT command. When you specify these attributes at the group level, they are identified as group-SPECIAL, group-AUDITOR, and group-OPERATIONS to distinguish them from attributes at the system level.

Group attributes are indicated in the description of the user-to-group connection in the user profile. Unless list-of-group checking is active, group attributes are in effect for the user only when the user is connected to the group during a batch job or terminal session.

If list-of-groups checking is active, then, regardless of which group the user is logged on to (the current connect group), RACF recognizes the user's group-related attributes in the user's other connect groups. (It is as though the user was logged on to each group at the same time.) For more information on list-of-groups checking, see [“Activating list-of-groups checking \(GRPLIST option\)” on page 111](#).



When you initially define a new user, the user's connection to the default group does not indicate any group-related attributes. You can then use the CONNECT command to define the user's group attributes within the default group.

## The scope of authority for the users with group-level attributes

The authority of the group-SPECIAL, group-AUDITOR, and group-OPERATIONS users is limited to the *resources* that are within the scope of the group. For details about users with group-level attributes and their scope of authority related to resources, users and resources, see [Table 6 on page 64](#), [Figure 5 on page 65](#), and [Figure 6 on page 66](#).

Resources, not users or other groups, that are within the scope of the group include the following.

- Resources owned by the group (for example, GROUP1.DATA owned by GROUP1)
- Resources that are owned by users who are owned by the group (for example, USER2.DATA owned by USER2 who is owned by GROUP1)
- Resources that are owned by subgroups that are owned by the group (for example, GROUP2.DATA owned by GROUP2, which is owned by GROUP1)
- Resources that are owned by subgroups that are owned by subgroups, owned by the group, and so on (for example, GROUPZ.DATA owned by GROUPZ, which is owned by GROUP2, which in turn is owned by GROUP1).

Note that the scope of the group does *not* extend to the following resources:

- Resources that are owned by groups that are owned by users who are owned by the group (for example, GROUPY.DATA owned by GROUPY which is owned by USER2 who is owned by GROUP1)
- Resources owned by users who are, in turn, owned by users who are owned by the group (for example, USER6.DATA owned by USER6 who is, in turn, owned by USER5 who is owned by GROUP2)

By establishing the group structure so that subgroups are owned by their superior groups, the authority of the group-SPECIAL, group-OPERATIONS, and group-AUDITOR user can be made to percolate down through the group tree structure as far as the security administrator desires. When a user's attribute percolates down from a group to which the user is connected with the group attribute, the user's authority in the subgroups is the same as if the user was connected directly to the subgroups with the group attribute.

**Note:** The data security monitor (DSMON) produces a group tree report that lists, for each requested group, all of its subgroups, all of the subgroups' subgroups, and so on. This report can be very useful in checking to which subgroups the authority of the group-SPECIAL, group-OPERATIONS, or group-AUDITOR applies. For more information on the group tree report, see [z/OS Security Server RACF Auditor's Guide](#).

The limits of the security administrator, group administrator, auditor, and operations personnel authority at the group level are described in [Table 6 on page 64](#). (Of course, these users continue to have whatever authorities they possess from other sources, such as ownership, that are not covered by their group-level authorities.)

Table 6. Scope of authority for user attributes at the group level

Resource	Attribute, user, and authority
Data sets	<p><b>Group-SPECIAL attribute:</b> A user with the group-SPECIAL attribute has full authority to work with:</p> <ul style="list-style-type: none"> <li>• Data set profiles that are owned by the group</li> <li>• Data set profiles that have a high-level qualifier that is the same as the group identifier</li> <li>• Data set profiles that are owned by users or groups that are owned by the group</li> <li>• Data set profiles that have a high-level qualifier that is a user or group identifier owned by the group</li> </ul> <p>The group-SPECIAL user can also define data set profiles with a high-level qualifier that is the group identifier or a user or group identifier owned by the group.</p> <p><b>Group-AUDITOR and group-OPERATIONS attributes:</b> A user with the group-AUDITOR or group-OPERATIONS attribute can perform all of the functions of an auditor or operator, but is limited to the same subset of data sets as the user with the group-SPECIAL attribute.</p>
General resources	<p><b>Group-SPECIAL attribute:</b> A user who has the group-SPECIAL attribute has full authority to work with:</p> <ul style="list-style-type: none"> <li>• Resource profiles that are owned by that group</li> <li>• Resource profiles belonging to users or groups that are owned by the group</li> </ul> <p>To create new resources, the user must have the CLAUTH attribute in the applicable class.</p> <p><b>Group-AUDITOR and group-OPERATIONS attributes:</b> A user who has the AUDITOR or OPERATIONS attribute can perform all of the functions of an auditor or operator, but is limited to the same above subset of resources as the user with the group-SPECIAL attribute.</p>
Users	<p><b>Group-SPECIAL attribute:</b> A user with the group-SPECIAL attribute has full authority to work with:</p> <ul style="list-style-type: none"> <li>• User profiles that are owned by the group</li> <li>• User profiles that are owned by a subgroup that is owned by the group, by a subgroup that is owned by a subgroup that is owned by the group, and so on</li> </ul> <p>The group-SPECIAL user must have the CLAUTH attribute in a class in order to give the CLAUTH attribute to another user in that class. The group-SPECIAL user cannot give a user the SPECIAL, AUDITOR, or OPERATIONS attribute at a system level, but can assign these attributes at the group level. To create new users, the group-SPECIAL user must have the CLAUTH attribute in the USER class.</p> <p><b>Group-AUDITOR attribute:</b> A user who has the group-AUDITOR attribute can perform all of the functions of an auditor, but is limited to the same subset of users as the user with the group-SPECIAL attribute.</p>
Groups	<p><b>Group-SPECIAL attribute:</b> A user who has the group-SPECIAL attribute has authority over that group, over subgroups owned by that group, and so on. The group-SPECIAL user can connect any user to, or remove any user from, any group that is included in this authority.</p>

The following two figures show the scope of authority of a group-SPECIAL user. [Figure 5 on page 65](#) shows a typical authority structure containing three major groups: Groups 1, 2, and 3.

[Figure 6 on page 66](#) shows the addition of a new element: a new user, USER1, is connected to Group 1. The resultant authority USER1 receives as a group-SPECIAL user is highlighted (the non-shaded area) in part 2 of this figure.

USER1 has authority to the profiles in the non-shaded area for the reasons summarized in [Table 6 on page 64](#). USER1 does *not* have authority to any of the resources in the shaded area for the following reasons:

- GROUP1 does not own IBMUSER, GROUP3, USER3, or USER4.



- GROUP1 does not own GROUPY.
- Neither GROUP1 nor GROUP2 own USER6.
- USER3.DATA is not owned by a user who is owned by GROUP1.
- USER4.DATA is not owned by a user who is owned by GROUP1. USER1 cannot display the profile information for this data set with LISTDSD, even if USER2, for example, is in its access list. (However, if USER1 runs the IRRUT100 utility, RACF informs USER1 that USER2 is in the access list of USER4.DATA.)
- U4A is not a general resource that is owned by a user who is owned by GROUP1.

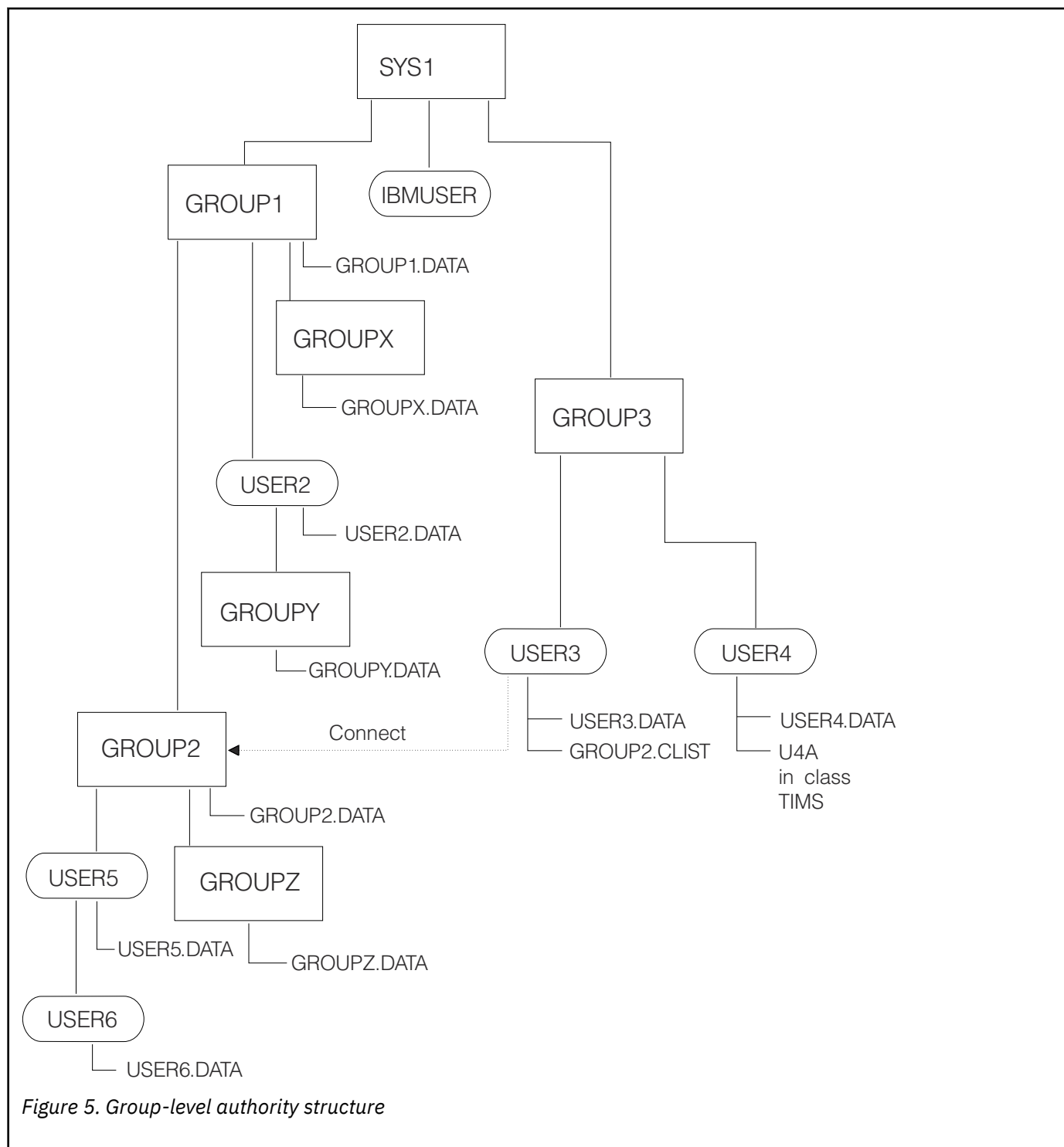


Figure 5. Group-level authority structure

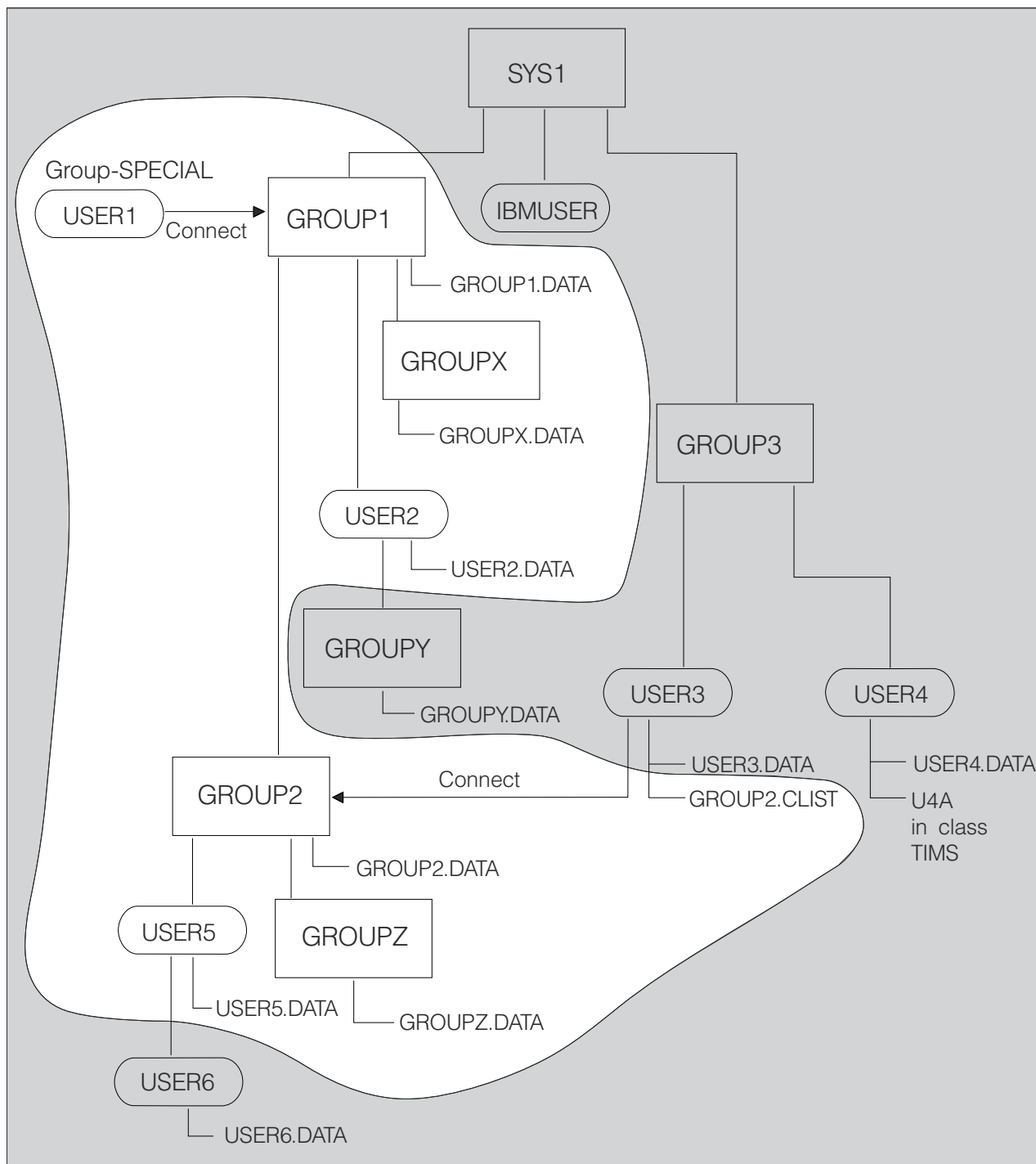


Figure 6. Scope of authority for a group-SPECIAL user

## Suggestions for assigning user attributes

When defining users to RACF with the ADDUSER command, or when modifying user attributes with the ALTUSER command, RACF security and group administrators should assign:

- SPECIAL, AUDITOR, ROAUDIT, and OPERATIONS attributes to only those users responsible for administering RACF on a system-wide basis
- CLAUTH attributes to only those users who are responsible for defining other users and general resources

- RESTRICTED attribute to only those user IDs that should not gain access to protected resources they are not specifically authorized to access

**Note:** You cannot assign the ADSP attribute to a user who allocates space for data sets that do not meet the RACF or installation naming conventions.

## Verifying user attributes

The data security monitor (DSMON) generates reports that describe the current status of the data security environment at your installation. Two of these reports, the selected user attribute report and the selected user attribute summary report, are useful for verifying the attributes that you have assigned.

The selected user attribute report lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, ROAUDIT, or REVOKE attributes and specifies whether they possess these attributes on a system-wide (user) or group level. You can use this report to verify that only those users whom you want authorized to perform certain functions have been assigned the corresponding attribute.

The selected user attribute summary report shows the number of installation-defined users and totals for users with the SPECIAL, OPERATIONS, AUDITOR, ROAUDIT, and REVOKE attributes, at both the system and group level. You can use this report to verify that the number of users with each of these attributes, on either a system or group level, is the number that your installation wants.

## Default universal access authority (UACC)

Each user who is connected to a group is assigned a default universal access authority (UACC) of NONE, READ, UPDATE, CONTROL, or ALTER. You can specify this default UACC on the ADDUSER, ALTUSER, or CONNECT command. If you do not specify a value for UACC, RACF uses NONE as a user's default universal access authority.

RACF uses this default UACC for all new resources that a user defines while connected to the specified default group as follows:

- When a user issues the ADDSD command to define a new data set profile and does not specify a value for the UACC operand, RACF uses the default UACC as the UACC for the profile unless profile modelling is used.
- When a user issues the RDEFINE command to define a new general resource profile and does not specify a value for the UACC operand, RACF uses the default UACC as the UACC for the profile unless a value for UACC is specified in the class descriptor table (CDT).

For more information on using the UACC operand on the ADDUSER, ALTUSER, and CONNECT commands, see [z/OS Security Server RACF Command Language Reference](#).

## Assigning security categories, levels, and labels to users

You can assign security categories and security levels to users and sensitive resources. You can also assign security labels, which are a combination of security levels and security categories, and are more easily maintained, to users and sensitive resources. User profiles and resource profiles that have been assigned a security label need not be changed if the definition of a security label is changed.

A security category is an installation-defined name corresponding to department or area within an organization that has similar security requirements. A security level is an installation-defined name that is associated with a number in the range 1 - 254.

If security levels and categories are being used (the SECDATA class is active), and security labels are *not* being used (the SECLABEL class is not active), RACF takes the following steps when a user requests access to a resource that has a security category or a security level associated with it:

1. If the resource has a SECLEVEL, RACF compares the security level of the user with the security level of the resource. If the resource has a higher security level than the user, RACF denies the request.

For a terminal session, the security level that RACF uses for the user is the lower of the user's SECLEVEL and the terminal's SECLEVEL. Thus, if the terminal has a SECLEVEL of 50 and the user has a

SECLEVEL of 100, the user cannot access, through that terminal, any data that has a SECLEVEL of over 50. RACF then proceeds to the category check.

If the resource does not have a SECLEVEL, RACF proceeds to the category check in Step 2.

2. RACF compares the list of security categories in the user's profile with the security categories in the resource profile. If the user's security level is high enough to access the resource, RACF compares the list of security categories in the user's profile with the list of security categories in the resource's profile. If RACF finds any security category in the resource profile that is not in the user's profile, RACF denies the request. If RACF does not deny the request, RACF continues with authorization processing. If there are no categories in the resource profile, RACF continues with authorization processing.

If your installation has activated the SECLABEL class, and a user requests access to a resource that has a security label associated with it, RACF *ignores* any security level or security categories that are specified in the resource profile. Instead, RACF performs security label authorization checking, which involves the security levels and categories that are used to define the security labels of the resource and the user.

You can use security labels as a simple replacement for security levels and categories, with the same access authority requirements, or you can use SETROPTS options such as MLACTIVE and MLS to set up a more rigorous security environment. For more information on how the SETROPTS options change the effects of security labels, see [“Security label authorization checking” on page 733](#). For more information on security classification of users and data, see [Chapter 5, “Classifying users and data,” on page 95](#).

## Passwords and password phrases

When a user logs on to a z/OS system, the user must supply an authentication factor to identify the user. In RACF, that authenticator can be either a password or a password phrase. A password is a traditional one to eight character alphanumeric value. A password phrase is a character string that consists of mixed-case letters, numbers, and special characters including blanks. Password phrases have security advantages over passwords as they are long enough to withstand most hacking attempts and are unlikely to be written down because they are easy to remember. A user can be assigned a password, a password phrase, both, or neither.

When a user profile is created, there is no default value assigned to either the password or the password phrase. The user is a protected user and will not be able to log on. This is acceptable and preferred as the situation for a started task or other functional user ID. Likewise, a protected user can be created by removing the password and password phrase from a user who has them. See [“Defining protected user IDs” on page 74](#) for more information.

The choice of the type of authenticator to assign depends on your policy and on which applications are used by the user. If the user authenticates to z/OS with an application that does not support password phrases, then the user must be assigned a password. If the user logs on to applications that support password phrases, then the user might be assigned a password phrase.

A user cannot assign the initial password or password phrase. The password or password phrase needs to be assigned by an authorized user. When one is assigned, the user can change that value at any time, but cannot remove it. When a user assigns an initial value, be sure that it is difficult to guess. By default, the user is forced to change this initial value the first time it is used.

To assign a password, use the PASSWORD operand of the ALTUSER command:

```
ALTUSER GLENN PASSWORD(g1GgiTty)
```

To remove a password:

```
ALTUSER GLENN NOPASSWORD
```

To assign a password phrase, use the PHRASE operand of the ALTUSER command:

```
ALTUSER STEWIE PHRASE('I shall rule the world!')
```

To remove a password phrase:

```
ALTUSER STEWIE NOPHRASE
```

These sample commands assign a value that must be changed by the user when the user first logs on, ensuring that from that point on, the user is the only one who knows the password. The ALTUSER command has a NOEXPIRED option that assigns a password or password phrase that does not need to be changed when the user logs on. This is intended for use by trusted applications that set RACF passwords on behalf of the user (for example, a password synchronization application). It should not be used by administrators because the principle of user accountability rests on the idea that a user is the only one who knows their own password. See [z/OS Security Server RACF Command Language Reference](#).

**Note:**

1. By default, passwords are one-way encrypted in the RACF database. However, RACF can be configured to envelope passwords so they can be recoverable in clear text by trusted applications such as a password synchronization application. See [Chapter 25, “Password and password phrase enveloping,”](#) on page 623 for more information.
2. A PassTicket is a one-time-use password substitute that can be used to authenticate a user. A PassTicket is not entered by a user, but automatically generated by an application to authenticate a user through a protocol that expects a user ID and password. See [Chapter 11, “Using PassTickets,”](#) on page 281 for more information.
3. A user can be required to authenticate with multiple authentication factors instead of a password or password phrase. In this case, the contents of the string that is entered by the user when logging on is not defined by RACF, but by IBM Multi-Factor Authentication for z/OS. This is transparent to the application to which the user is logging on where it appears to be either a password or password phrase, depending on its length, and is passed to the appropriate authentication API. RACF passes it to IBM MFA when appropriate for authentication. See [“Multi-Factor Authentication for z/OS”](#) on page 70 for more information.

The following topics describe various considerations and options for passwords and password phrases:

- [“Establishing password syntax rules \(PASSWORD option\)”](#) on page 108
- [“Allowing mixed-case passwords \(PASSWORD option\)”](#) on page 106
- [“Allowing special characters in passwords \(PASSWORD option\)”](#) on page 106
- [“Assigning password phrases”](#) on page 69
- [“Setting the maximum and minimum change interval \(PASSWORD option\)”](#) on page 108
- [“Extending password and user ID processing \(PASSWORD option\)”](#) on page 109
- [“Specifying the encryption method for user passwords”](#) on page 139.

## Assigning password phrases

You can issue the PHRASE operand of the ADDUSER or ALTUSER command to assign a password phrase for a user. This enables the user to authenticate using a password phrase instead of a password when using an application that supports password phrases.

```
ALTUSER ARUNDATI PHRASE('g0d0fsm@11things')
```

A password phrase is a character string consisting of mixed-case letters, numbers, and special characters including blanks. Password phrases have security advantages over passwords in that they are long enough to withstand most hacking attempts yet are unlikely to be written down because they are so easy to remember.

Unless you specify NOEXPIRED with the ALTUSER command when you set a password phrase, it is set as expired, requiring the user to change it on initial use.

RACF enforces a basic set of syntax rules to establish strength in password phrases. These syntax rules apply to all password phrases and you cannot alter or avoid them. However, you can add password phrase syntax rules to impose additional restrictions when your installation tailors the new-password-phrase exit

(ICHPWX11). IBM provides a sample exit routine that allows your installation to add syntax rules coded in REXX.

When the new-password-phrase exit (ICHPWX11) is present and allows it, the password phrase can be 9 - 100 characters. When ICHPWX11 is not present, the password phrase must be 14 - 100 characters. Contact your system programmer to find out if your installation uses the new-password-phrase exit (ICHPWX11). See [z/OS Security Server RACF System Programmer's Guide](#) for programming details.

### Syntax rules for password phrases:

- Maximum length: 100 characters
- Minimum length:
  - 9 characters, when the encryption algorithm is KDFAES or ICHPWX11 is present and allows the new value
  - 14 characters, when ICHPWX11 is not present and the encryption algorithm is not KDFAES
- 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
- If a single quotation mark is intended to be part of the password phrase, you must use two single quotation marks together for each single quotation mark.

If the new-password-phrase exit (ICHPWX11) is present, it can reject the specified password phrase. RACF rejects password phrases shorter than 14 characters unless ICHPWX11 is present and allows the new value.

If the specified password phrase is accepted, it is made the user's current password phrase and, when SETROPTS PASSWORD(HISTORY) is in effect, it is added to the user's password phrase history.

See [z/OS Security Server RACF Command Language Reference](#) for details about using the PHRASE operand of the ADDUSER and ALTUSER commands.

## Multi-Factor Authentication for z/OS

---

Today, the most common way for users to access z/OS systems is by the use of passwords or password phrases. Due to the simplicity of passwords, they can present a relatively simple point of attack for exploitation. In order for systems that rely on passwords to be secure, they must enforce password controls and provide user education. Some of the common problems with a simple password are that users tend to: choose common passwords, write down their passwords, or unintentionally install malware that can key log passwords.

A more secure option is for systems to require multiple authentication factors to verify the user's identity.

### What is Multi-Factor Authentication?

A multi-factor authentication system requires that multiple authentication factors be presented during login to verify a user's identity. Each authentication factor must be from a separate category of credential types:

- Something you know: A password or security question
- Something you have: An ID badge or cryptographic token device
- Something you are: Fingerprint or other biometric data

By requiring multiple authentication factors, a user's account can not be compromised if one of their factors is discovered.

## IBM MFA on z/OS

IBM Multi-Factor Authentication for z/OS, RACF provides support for authenticating with multiple authentication factors. RACF users can be configured to require authentication through IBM MFA. For these select users, RACF calls IBM MFA to help in making the authentication decision during logon processing.

## Configuring RACF for IBM MFA

There are a number of steps to be completed in order to begin using IBM Multi-Factor Authentication for z/OS with RACF. IBM MFA should be installed as described in *IBM Z Multi-Function Authentication Installation and Customization*, which is available at the [IBM Z Multi-Factor Authentication documentation \(www.ibm.com/docs/en/zma\)](http://www.ibm.com/docs/en/zma) website. Then, perform the following steps to configure RACF for MFA:

### 1. Define the factor to RACF:

An IBM MFA factor is defined by creating an MFADEF class profile with the name `FACTOR.factor-name`. Supported authentication factors are listed in the IBM Multi-Factor Authentication for z/OS product documentation. Note that a single factor name may enforce multiple authentication factors during logon.

For example, to define the RSA SecurID factor supported by IBM MFA:

```
RDEFINE MFADEF FACTOR.AZFSIDP1
```

### 2. Assign the factor to users:

MFA factor data can be added to a RACF user ID with the MFA keyword of the ALTUSER command. The factor must be defined in the MFADEF class before this step can be completed. The sub-keywords of MFA are:

#### **FACTOR/DELFACOR**

Use the FACTOR keyword to identify the name of the factor that is being added or modified.

Use the DELFACTOR keyword to delete a factor from a user profile.

#### **ACTIVE/NOACTIVE**

Use the ACTIVE keyword to activate a factor for use during logon.

Use the NOACTIVE keyword to disable a factor and revert to password checking.

#### **TAGS/DELTAGS/NOTAGS**

Use the TAGS keyword to assign configuration data that is specific to the factor. The data is specified in *name:value* format. The IBM Multi-Factor Authentication for z/OS product documentation contains information on supported tags. IBM MFA is called to validate the data. The MFA started task must be available when assigning tags, or the ALTUSER command fails.

Use the DELTAGS keyword to delete specific tags.

Use the NOTAGS keyword to delete all tags for the specified factor.

#### **PWFALLBACK/NOPWFALLBACK**

Use the PWFALLBACK keyword to allow the user to logon with a RACF password or password phrase whenever the ability to perform multi-factor authentication is not available (for example, the MFA started task is down). PWFALLBACK is not factor-specific.

Use NOPWFALLBACK to require the user to always authenticate using MFA.

#### **ADDPOLICY/DELPOLICY**

Use the ADDPOLICY keyword to add the user's list of MFA authentication policies where *policy-name* is the name of the MFA policy profile defined in the MFADEF class.

Use the DELPOLICY keyword to delete the specified policies from the user's list of MFA policies.

#### **NOMFA**

Use the NOMFA keyword to remove all MFA data from a user's profile.



See the [z/OS Security Server RACF Command Language Reference](#) for more information on the MFA keywords.

### Example:

To require a user to authenticate with RSA SecurID, but allow the user to logon with their RACF password when MFA is unavailable:

```
ALTUSER SLJAXON MFA(FACTOR(AZFSIDP1) ACTIVE PWFALLBACK  
TAGS(SIDUSERID:SamLJ))
```

### 3. Activate MFA checking:

When setup is complete, activate the MFADEF class.

```
SETOPTS CLASSACT(MFADEF)
```

When this is completed, RACF will call IBM Multi-Factor Authentication for z/OS to perform user authentication for any user who has an active MFA factor.

MFA checking can be disabled for all users by deactivating the MDADEF class:

```
SETOPTS NOCLASSACT(MFADEF)
```

## MFA considerations for the RACF password and password phrase

MFA information cannot be assigned to a PROTECTED user, and thus an MFA user must have a password or password phrase.

When the user is assigned the NOPWFALLBACK attribute, the password/phrase cannot be used to logon. In this case, consider assigning the user a long, random password phrase.

When the user is assigned the PWFALLBACK option, the user needs to maintain the password as usual. However, the password will not be able to be changed during logon unless MFA is unavailable, the user's password is expired, and the application prompts the user to enter a new password. The user's password can be changed using the PASSWORD or ALTUSER command.

## MFA application bypass

In some cases it may be desirable to bypass select z/OS applications from MFA processing. IBM MFA provides controls to allow the RACF administrator to name applications for which MFA authentication will not be enforced. The MFA bypass controls allow for different bypass policy for different RACF users.

For more details on the MFA application bypass feature, refer to *IBM Z Multi-Function Authentication Installation and Customization* and *IBM Z Multi-Function Authentication User's Guide* in the [IBM Z Multi-Factor Authentication documentation \(www.ibm.com/docs/en/zma\)](http://www.ibm.com/docs/en/zma) website.

## MFA policy

Installations can create MFA policies to define a set of rules that users must follow when authenticating with IBM MFA. The policy attributes are defined in the MFPOLICY segment of profiles in the MFADEF class. These policies can be associated with individual users with the ALTUSER ADDPOLICY keyword.

For information about on MFA policies, refer to *IBM Z Multi-Function Authentication Installation and Customization* and *IBM Z Multi-Function Authentication User's Guide* in the [IBM Z Multi-Factor Authentication documentation \(www.ibm.com/docs/en/zma\)](http://www.ibm.com/docs/en/zma) website.

## MFA compound In-Band

MFA users may be provisioned so that they are required to authenticate with both a token device code and their RACF authenticator (password or password phrase). Users configured for compound in-band may enter their token code and RACF authenticator concatenated together in the password phrase field of an application with a separator character between the two values. When the RACF authenticator is



expired it can be changed by passing the new value into the new password or new password phrase field of the application by itself without the token code portion.

For more information on MFA compound in-band support, refer to *IBM Z Multi-Function Authentication Installation and Customization* and *IBM Z Multi-Function Authentication User's Guide* in the [IBM Z Multi-Factor Authentication documentation \(www.ibm.com/docs/en/zma\)](http://www.ibm.com/docs/en/zma) website.

## Limiting when a user can access the system

Installations can limit a user's ability to log on by limiting:

- The user's ability to log on to the system to certain days of the week, and certain hours within each day
- The use of individual terminals (in the TERMINAL class only) to certain days of the week, and certain hours within each day

To limit the times during which a user can enter the system, use the WHEN operand on the ADDUSER and ALTUSER commands. For example, to specify that USER12 can enter the system only on weekdays between the hours of 7:00 a.m. and 5:00 p.m., enter:

```
ADDUSER USER12 WHEN(DAYS(WEEKDAYS) TIME(0700:1700))
```

Similarly, to control when users can access the system from a specific terminal, specify the WHEN operand on the RDEFINE and RALTER commands for the appropriate profile. For example, to specify that terminal TRM07C can be used at any time during the week, but not at all during the weekend, enter:

```
RDEFINE TERMINAL TRM07C WHEN(DAYS(WEEKDAYS))
```

Note that on the RDEFINE command, TIME(ANYTIME) is the default.

The WHEN operand on these commands (for both users and terminals) allows you to specify individual days and specific times within these days.

RACF also provides support for installations that have terminals in different time zones by associating with each terminal its location relative to the local time where the processor complex on which RACF is executing is located.

**Note:** RACF does not provide any specific support for daylight saving time. If the installation changes the value of the local time (as given by the TIME macro) to accommodate daylight saving time, RACF automatically adjusts its time calculations accordingly. However, if any terminals are located in an area that does not follow the same time adjustment, you must adjust the terminal information.

For more information on the WHEN operand, see the command descriptions in [z/OS Security Server RACF Command Language Reference](#).

## Time-of-day and day-of-week checking for users and terminals

The time and day-of-week checking for users and terminals applies when users log on to terminals from products such as TSO/E, IMS, CICS, and NetView (beginning with Release 2), but does not apply to batch jobs or started tasks.

User verification processing includes the following time/day-of-week checks:

1. The user's authority to use the specific terminal. If the profile protecting the terminal does not have any time or day-of-week information, the user can log on. If there is time and day-of-week information, RACF calculates the time-of-day at the location of the terminal from which the user is logging on (if that time is different from the time-of-day at the location of the processor complex), and checks whether the terminal can be used at this time on this day of the week.
2. The user's authority to access the system. If the user's profile does not have any time or day-of-week information, the user can log on. If there is time and day-of-week information, RACF calculates the time-of-day at the location of the terminal from which the user is logging on (if that time is different from the time-of-day at the location of the processor complex), and checks whether the user can enter the system.

## Defining protected user IDs

You can define a protected user ID by assigning the NOPASSWORD, NOPHRASE, and NOOIDCARD attributes through the ADDUSER or ALTUSER command. Protected user IDs are protected from being used to logon to the system and from being revoked through inactivity or unsuccessful attempts to access the system using incorrect passwords and password phrases. However, they can be revoked using the ALTUSER (*userid*) REVOKE command. If revoked, protected user IDs can be activated using the ALTUSER (*userid*) RESUME command.

A protected user ID cannot be used to enter the system by any method that uses, a supplied password, such as TSO logon, CICS signon, PassTicket authentication, z/OS UNIX `rlogin`, batch job submission when a password is specified using the PASSWORD parameter of the JOB statement, or by supplying a password phrase. Before assigning the PROTECTED attribute to a user ID, you should ensure that the user ID will not be used in any situation where specification of a password, PassTicket, or password phrase is required.

Applications can authenticate a protected user ID with an Identity Token (IDT) when the covering IDTDATA profile indicates PROTALLOWED(YES). See z/OS Security Server RACF Command Language Reference and z/OS Security Server RACROUTE Macro Reference for more details on the RACF IDT support.

You might want to assign protected user IDs to z/OS UNIX, and to the UNIX daemons, started procedures, applications, servers or subsystems associated with z/OS UNIX, to minimize their exposure to inadvertent or malicious misuse or revocation. Surrogate-submitted batch jobs can use protected user IDs. See [“Using protected user IDs for batch jobs” on page 458](#) for more information. Protected users can be associated with started procedures defined in the STARTED class (preferred method) or in the started procedures table (ICHRIN03). For more information, see [“Assigning RACF user IDs to started procedures” on page 141](#).

The following example shows the ALTUSER command used to assign the PROTECTED attribute to an existing user ID.

```
ALTUSER SERVER8 NOPASSWORD NOPHRASE
```

A protected user ID will have the PROTECTED attribute displayed in the output of the LISTUSER command.

To remove the PROTECTED attribute from an existing user ID, use the ALTUSER command to assign a password:

```
ALTUSER SERVER8 PASSWORD(password)
```

## Restrictions for using protected user IDs with z/VM systems

If you share the RACF database with a z/VM system and use protected user IDs, the following restrictions apply for releases prior to z/VM Version 6 Release 2:

- Protected user IDs can be used to attempt logon from the z/VM system. This might result in protected user IDs being revoked through malicious or inadvertent incorrect logon attempts from the z/VM system.
- Do not administer protected user IDs from the z/VM system by issuing ADDUSER or ALTUSER command with the PASSWORD/NOPASSWORD or PHRASE/NOPHRASE operands. If you issue the ALTUSER command with these operands from the z/VM system, a protected user ID might lose its PROTECTED attribute.
- When you issue a LISTUSER command from the z/VM system for a protected user ID, the output does not indicate the PROTECTED attribute.

**Note:** As of z/VM 7.3, sharing of the RACF database between z/OS and z/VM is no longer supported. The RACF publications for z/OS V2R5 are updated to remove references to database sharing with RACF/VM. For more information, see [“As of z/VM 7.3, sharing of the RACF database between z/OS and z/VM is not allowed” on page 10](#).

## Defining restricted user IDs

---

You can define a restricted user ID by assigning the RESTRICTED attribute through the ADDUSER or ALTUSER command. Restricted user IDs cannot be used to access protected resources they are not specifically authorized to access. Access authorization for restricted user IDs bypasses global access checking. In addition, the UACC of a resource and an ID(\*) entry on the access list are not used to enable a restricted user ID to gain access.

The RESTRICTED attribute does not prevent users from gaining access to z/OS UNIX file system resources unless you take certain steps. See [“Controlling access to file system resources for restricted users”](#) on page 531 for information about preventing restricted users from gaining access to file system resources they are not explicitly authorized to access.

The RESTRICTED attribute can be added to shared user IDs, such as PUBLIC and ANONYMOS, that are assigned by application servers that allow users to enter the system without identifying themselves. Without the RESTRICTED attribute, users that are assigned shared user IDs can gain access to any resource that has an ID(\*) entry in the access list, UACC, or global entry that allows access.

The following example shows the ALTUSER command used to assign the RESTRICTED attribute to an existing shared user ID.

```
ALTUSER ANONYMOS RESTRICTED
```

A restricted user ID has the RESTRICTED attribute displayed in the output of the LISTUSER command.

## Using restricted user IDs for digital certificate users

Users who identify themselves by supplying a digital certificate that is not registered to RACF are eligible for certificate name filtering. These users can be assigned a RACF user ID on your system if there is an applicable name filter in effect. See [“Certificate name filtering”](#) on page 571 for more information.

To prevent users who gain access through certificate name filtering from accessing protected resources they are not specifically authorized to access, you should assign restricted user IDs to each user ID associated with a certificate name filter.

## Using restricted user IDs for distributed identity users

Users who identify themselves on distributed application servers before initiating certain transactions on z/OS subsystems might be assigned a RACF user ID on your system if there is an applicable distributed identity filter in effect. See [Chapter 28, “Distributed identity filters,”](#) on page 671 for more information.

To prevent users who are assigned a RACF user ID by a distributed identity filter from accessing protected resources they are not specifically authorized to access, assign a restricted user ID to each user ID associated with a distributed identity filter.

## Summary of steps for defining users

---

This summary presents the steps required by RACF and related IBM licensed programs to define users to RACF. Your installation might require additional steps, depending on your security policy and the products you have installed.

1. Prepare to create the user profile as follows:
  - Decide which default connect group to assign to the user. If a group profile does not yet exist for the group, create the group using the procedure described in [“Summary of steps for defining a RACF group”](#) on page 93.
  - Decide which user ID to assign to the user.
  - Decide which user or group is to be the owner of the user profile. (If the owner is a user, give him or her the information needed to manage the new profile.)

- Decide if the user should be allowed to use a password to access the system and if so, choose the user's initial password. Specify a non-trivial value.
  - Decide if the user should be allowed to use a password phrase to access the system and if so, choose the user's initial password phrase.
  - Determine if the user's access to the system should be limited to certain days of the week, hours of the day, or both.
  - Decide which user attributes (such as SPECIAL or AUDITOR) the user should have, and whether the user attributes should be limited to the scope of a group (group-SPECIAL or group-AUDITOR).
  - If security labels are used, decide which security label to assign to the user.
  - Decide whether the user can establish user ID associations to enable password synchronization and command direction between user IDs. See [Chapter 17, “The RACF remote sharing facility \(RRSF\),” on page 405](#) for more information.
  - If DFSMSdss is in use, work with the storage administrator to do the following:
    - Determine the initial values in the user's DFP segment.
    - Determine which DFP resources the user should have access to.
  - Determine which primary and secondary languages the user should have (if they should be different from the installation defaults set by the SETROPTS command).
2. If you want to authorize the user to establish an extended MCS console session, work with the system operations planner to determine the initial values in the user's OPERPARM segment. For more information, see [“The OPERPARM segment in user profiles” on page 52](#) and [z/OS MVS Planning: Operations](#).
  3. If the user is a CICS user, work with the CICS administrator to do the following:
    - Determine the initial values in the user's CICS segment.
    - Determine which primary and secondary languages the user should have (if they should be different from the CICS-specified installation defaults).

**Note:** CICS does not check the installation defaults set by the SETROPTS command.

    - Determine the CICS resources to which the user should have access.
    - For more specific CICS and RACF security information, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).
  4. Work with the APPC administrator to do the following:
    - Determine the initial values in the user's **WORKATTR** segment.
    - Determine which APPC/MVS resources the user should have access to.
  5. Create the user profile. You can use any of the following methods:
    - Issuing the ADDUSER command.
    - Enrolling the user through the TSO/E Information Center Facility (ICF) panels.

For more information about administering the Information Center Facility, see [z/OS TSO/E Administration](#).

Here is an example of using the ADDUSER command to create a user profile. Suppose you want to create a user profile for user Steve H., a member of Department A. You want to assign the following values:

- STEVEH for the user ID
- DEPTA for the default connect group
- DEPTA for the owner of the STEVEH user profile
- R3I5VQX for the initial password
- Steve H. for the user's name

Steve H. does not require any of the user profile segments except TSO. The TSO segment values that you want to set to start with are 123456 for the account number and PROC01 for the logon procedure.

To create a user profile with these values, enter:

```
ADDUSER STEVEH DFLTGRP(DEPTA) OWNER(DEPTA) NAME('Steve H.')
        PASSWORD(R315VQX) TSO(ACCTNUM(123456) PROC(PROC01))
```

6. Create a *top* generic profile for the user in the DATASET class using the ADDSD command.

For example, if the user's user ID is STEVEH, enter:

```
ADDSD 'STEVEH.**' UACC(NONE)
```

7. If users at your installation manage their own resource profiles, give them the information they need. For example, they might need to use portions of [z/OS Security Server RACF Command Language Reference](#).
8. If the user is to define general resource profiles, (as, for example, an administrator might), give the user the CLAUTH attribute in the appropriate classes and the information needed for working with those profiles, for example, the JESSPOOL class.

**Note:** If the SETROPTS GENERICOWNER option is in effect, you must create a *top* profile for the user in the JESSPOOL class, make the user the owner of the profile, and give the user CLAUTH(JESSPOOL). For more information, see [“Letting users create their own JESSPOOL profiles”](#) on page 486 and [“Defining profiles for SYSIN and SYSOUT data sets”](#) on page 484.

9. If needed, give the user access to RACF-protected resources. This can be done using one or both of the following:
  - Connect the user to groups that have the same access requirements as this user, using the CONNECT command.

For example, to allow user STEVEH to have access to his department's resources (that is, to resources belonging to group DEPTA), enter:

```
CONNECT STEVEH GROUP(DEPTA) OWNER(DEPTA)
```

By default, the command gives USE authority to STEVEH.

- If the user requires specific access to RACF-protected resources (beyond that permitted by connecting the user to groups), give the user the access required, using the PERMIT command.

Consider the following:

- If the user is a TSO user, remember the necessary TSO resources (such as TSOPROC).
- If data sets are managed by SMS, remember the MGMTCLAS and STORCLAS classes.

For example, to give user STEVEH permission to use a customized TSO logon procedure called CUSTPROC (whose profile in the TSOPROC general resource class has already been defined with a universal access of NONE), enter:

```
PERMIT CUSTPROC CLASS(TSOPROC) ID(STEVEH) ACCESS(READ)
```

## Summary of steps for deleting users

This summary presents the steps required by RACF and related IBM licensed programs to delete users from RACF. Your installation might require additional steps, depending on your security policy and the products you have installed.

1. To prevent the user from entering the system, revoke the user ID:

```
ALTUSER userid REVOKE
```

2. If the user is already logged on to the system, or has a job running on the system, ask the system operator to examine any logons (or jobs) for the user and cancel those that should not be allowed to continue.
3. Use the RACLINK LIST command to see if the user has any user ID associations defined. If so, use the RACLINK UNDEFINE command to delete them. You cannot delete a user ID that has any associations defined. See [Chapter 17, “The RACF remote sharing facility \(RRSF\),” on page 405](#) for more information.
4. Use the RACMAP LIST command to see if the user has any distributed identity filters defined. If so, use the RACMAP DELMAP command to delete them. You cannot delete a user ID that has any distributed identity filters defined. See [Chapter 28, “Distributed identity filters,” on page 671](#) for more information.
5. Find all of the data sets associated with this user (that is, data sets for which the user's user ID is the high-level qualifier of the data set name) and perform the following steps:
  - a. Delete or rename (with a new high-level qualifier) the user's user data sets. If you rename or delete a data set that is protected by a discrete profile, the discrete profile is also renamed or deleted.

**Tip:** You can do this using the DATA SET LIST utility of ISPF.

- b. Identify all of the remaining (generic) data set profiles, create new ones modeled on them if needed, and then delete the remaining profiles.

**Important:** Make sure that you do not delete an old profile unless it is no longer needed.

**Tips:**

- i) You can use the following SEARCH command to identify the user's data set profiles:

```
SEARCH MASK(userid.) CLIST('LISTDSD DA(' ' ) ALL')
```

As specified, the CLIST operand generates a CLIST that you can run to list all of the information in the data set profiles. This can help you assess whether to use the profiles as models.

- ii) You can use the FROM operand on the ADDSD command to create new profiles modeled on the old profiles.

If the user has profiles in other classes (such as the JESSPOOL, JESJOBS, and NODES classes) that might have the user's user ID in their profile names, use the FILTER operand on the SEARCH command. For example:

```
SEARCH CLASS(classname) FILTER(**.userid.**)  
CLIST('RDELETE classname')
```

6. To research the following steps, use the IRRRID00 utility to list the occurrences of the user ID in the RACF database. For information, see [“Using the RACF remove ID \(IRRRID00\) utility” on page 391](#).
7. If the user is the owner of group data set profiles (the user's user ID was specified on the OWNER operand on the ADDSD or ALTDSD command for the group data set profile), decide which user or group is to be the new owner of the group data set profiles.
 

**Tip:** If the user is the owner of any group data set profiles, specify the new owner on the OWNER operand of the REMOVE command.
8. If the user is a TSO user and has a SYS1.UADS entry, work with the TSO administrator to delete the entry.
9. If the user is a CICS user and has an entry in the CICS signon table, work with the CICS administrator to delete the entry.
10. Remove the user from any access lists in which the user's user ID is specified.

**Tip:** To do this, use the DELETE operand on the PERMIT command.

For example, suppose user ELVIS has update permission to a set of data sets defined in the PROJA.\*\* profile. To remove ELVIS from the profile's access list, enter:

```
PERMIT 'PROJA.**' ID(ELVIS) DELETE
```

11. If the user owns any RACF profiles, change the OWNER field of the profile.

**Tip:** To do this, use the appropriate command for changing profiles, such as ALTUSER or RALTER.

12. After all occurrences of the user ID are deleted from the RACF database, use the DELUSER command to delete the user profile.

For example, to delete the profile for user ELVIS, enter: DELUSER ELVIS

## General considerations for user ID delegation

This topic discusses things to consider for delegating administrative tasks to other users.

- In general, centralize first, delegate later.
- Consider the trade-offs:
  - Should one user handle all of the administration workload?
  - Should many users all be learning RACF simultaneously?
- RACF groups (not users) should own resource profiles.
- Authorize groups rather than users to resource profiles.
- Delegate power (group-SPECIAL) with care.
- Have *standby* SPECIAL and OPERATIONS user IDs for emergency situations.

**Guideline:** Carefully limit who has knowledge of the passwords for *standby* user IDs, and change those passwords when personnel changes occur.

- After control has been given, it is difficult to take it away again.
- Group-SPECIAL is the most powerful authority a user can have at the group level.
  - Group-SPECIAL enables the user to use more commands.
  - Group-SPECIAL also percolates to other groups, as far as the scope of the group allows.

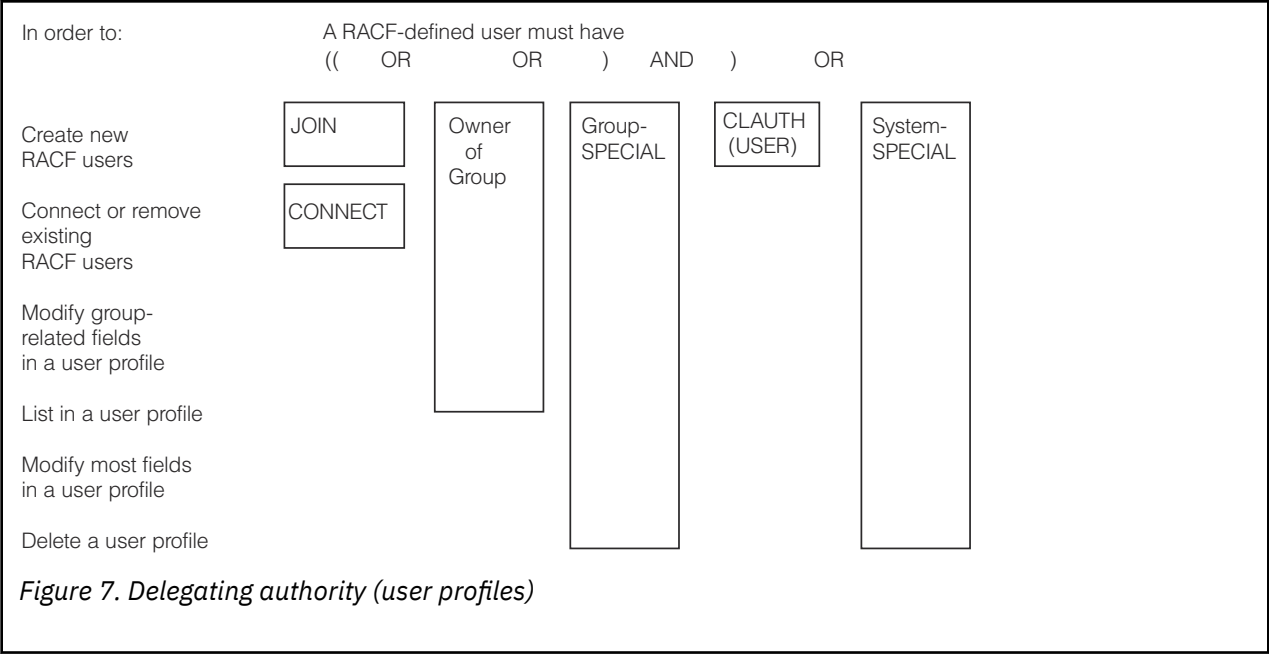
Choose the best option for your installation.

- For authority over a *single* group of users based on protection objectives, use JOIN and CLAUTH(USER).
- For authority over one or more groups of users based on protection objectives and scope of the group, use group-SPECIAL and CLAUTH(USER).

**Note:** The group-SPECIAL attribute allows password resetting for user IDs within the group whereas JOIN does not.

Figure 7 on page 80 shows delegating authority in another way.





A user with the SPECIAL attribute has full authority over all users and groups. By contrast, a user without the SPECIAL attribute might require a combination of authorities to complete the same tasks with limited scope.

For example, to create a new RACF user, the creating user without the SPECIAL attribute must have at least one of the following *and* have the CLAUTH(USER) attribute:

- JOIN group authority in the new user's default group
- or*
- Be the owner of the new user's default group
- or*
- Have group-SPECIAL in the new user's default group
- or*
- Have SPECIAL

For detailed information about the authorities required for the following administrative tasks related to user ID delegation, see the "Authorization required" topic for the associated RACF command in [z/OS Security Server RACF Command Language Reference](#).

User ID delegation tasks	Associated RACF command
Create a new RACF user	ADDUSER
Connect or remove an existing RACF user	CONNECT or REMOVE
Reset passwords or modify fields in a user profile	ALTUSER
List user profile information	LISTUSER
Delete a user	DELUSER

For details about the group-SPECIAL attribute, see [“User attributes at the group level” on page 62](#) and [“The SPECIAL or group-SPECIAL attribute” on page 699](#).

For details about delegating administrative tasks to help desk personnel, see [Chapter 27, “Authorizing help desk functions,” on page 655](#).



## RACF user ID containment

When you revoke a user ID, you prevent the user from initiating a new session or submitting new work. However, any sessions that are already started by the user remain active, which allows the user to continue accessing system resources based on the user's privileges. Work that is in progress under that user ID, such as active TSO/E sessions, submitted jobs, and server subtasks, continues to function.

In the case of a suspected malicious user, your security administrator might encounter a situation in which it is necessary to immediately deny all access to a user, including active sessions, without knowing what processes are currently operational for the user.

With user ID containment, RACF provides the ability for the administrator to *contain* a user ID, stopping the user from accessing system resources from any active sessions. With a single command, the administrator can immediately revoke the privileges of the user to protect critical resources on the system. When a user is contained, all SAF calls made on the user's behalf fail. The user is effectively quarantined.

RACF provides the user ID containment function as an extension of its user ID revocation processing. This functionality is available in z/OS 3.2 and z/OS 3.1 when you install the PTF for APAR OA67286. On a z/OS 3.1 system, you must also install the PTF for SAF APAR OA67288.

Initiating and managing RACF user containment is performed by authorized personnel using RACF commands. When a user ID is contained, RACF assigns the CONTAINED attribute to the user's profile in the RACF database and maintains the user ID in an in-storage list called the containment list.

After a user ID is contained, your security team can diagnose the event that caused the containment and then determine which subsequent remediation actions should be taken. For example, after evaluating a potential threat, you might choose to delete a contained user ID from the system (and possibly add it back later). Or you might determine that it is safe to remove the user ID from containment and continue using it. Further, it is possible to mark certain user IDs as exempt from containment (through the NEVERCONTAIN attribute), and later change that setting, if needed.

### How does it work

Most operations related to user ID containment are performed by using the ALTUSER command with the appropriate operands: CONTAIN, NOCONTAIN, NEVERCONTAIN, and ALLOWCONTAIN. Other functions involve the use of the ADDUSER, LISTUSER, and SETROPTS commands. For descriptions of these commands, including syntax and authorization requirements, see [z/OS Security Server RACF Command Language Reference](#).

The following summary describes how these commands are used in a typical containment scenario:

- The ALTUSER CONTAIN command is used to contain a user ID. This command indicates that the specified user ID is to be immediately denied access to system resources, even from currently active sessions. The CONTAIN operand behaves the same as the REVOKE operand without a date, and, in addition, causes all future access requests to fail in active sessions. It adds the CONTAINED and REVOKED attributes to the user ID and places the user ID in the containment list.

The CONTAIN operand cannot be used with the RESUME operand and if a resume date exists in the affected profile, the date is removed.

The following example shows the ALTUSER command used to assign the user SOMEUSER to the containment list.

```
ALTUSER SOMEUSER CONTAIN
```

- The ALTUSER NOCONTAIN command is used to remove a user ID from containment, which allows the user's access requests in active sessions to complete. However, NOCONTAIN by itself does not resume the user ID. To successfully resume a user ID under these circumstances, you must also specify the RESUME keyword, on the same ALTUSER command or on separate ALTUSER commands.

- A user can be made exempt from containment by being assigned the NEVERCONTAIN attribute through ALTUSER. If a user is contained when this attribute is set, the user ID remains contained. A separate ALTUSER command can be entered with the NOCONTAIN operand to remove containment of the user.
- To remove the NEVERCONTAIN attribute from a user ID, use the ALTUSER ALLOWCONTAIN command.
- The DELUSER command deletes a user from the RACF database, but does not remove the user ID from the containment list. If you want to remove a user ID from the containment list, use an ALTUSER command with the NOCONTAIN keyword before deletion. If the user is deleted while contained and needs to be added again, use an ADDUSER command with the NOCONTAIN keyword.
- The ADDUSER command checks for containment before adding the user. If the user being added has the CONTAINED attribute, the ADDUSER command fails, unless it also includes the NOCONTAIN keyword to remove the user ID from containment.
- The LISTUSER command displays the containment status of a specific user. A contained user ID has the CONTAINED attribute displayed in the output of the LISTUSER command.

**Tip:** If you ever find that a particular user ID cannot be resumed, the cause might be that the user ID is contained. Here, you can use the LISTUSER command to determine whether the user ID is contained and take the necessary action to resume it, such as entering the ALTUSER command with NOCONTAIN and RESUME specified.

- The SETROPTS command can be used to display the list of user IDs that have been contained since the last IPL. User IDs that were contained prior to IPL cannot have active sessions; they were revoked when contained and cannot be resumed until the CONTAINED attribute is removed from the associated user profile.

## How containment status is communicated in the system

When a user ID is contained, SAF calls made on the user's behalf fail with a unique reason code, console message, and an SMF 80 type 80 record (event code 92) to log the user containment event.

RACF communicates changes to the containment status of a user ID through ENF signal 71. Active listener exits can detect such changes by checking for the containment-related qualifiers in signal 71. For more information, see [ENF2: RACF parameter list for ENF event code 71 listen exits in z/OS Security Server RACF Data Areas](#).

The R\_Admin callable service returns information about user ID containment. For more information, see [R\\_admin \(IRRSEQ00\): RACF administration API in z/OS Security Server RACF Callable Services](#).

A new event type and security event record extension is defined for SMF record type 80. For more information, see [The format of the unloaded SMF type 80 data in z/OS Security Server RACF Macros and Interfaces](#).

The "Selected user attribute summary report" of the data security monitor (DSMON) is updated to list attributes related to user ID containment. For more information, see [The data security monitor \(DSMON\) in z/OS Security Server RACF Auditor's Guide](#).

## Usage considerations

Observe the following usage considerations:

- Containment is intended to be a short-term state for a user ID. It is an immediate response to protect the system from a perceived threat. After your security team has conducted an analysis of the event that prompted the containment, take the appropriate actions to resolve the issue, for example, by deleting or reinstating the user ID, depending on your findings.
- It is recommended that you carefully monitor the containment of user IDs. Having a list of contained user IDs can sometimes affect system performance, depending on your workload mix.
- Containing a user ID for a server type address space makes the associated server function unavailable, and thus can have negative effects on your system.
- When you assign the NEVERCONTAIN attribute to a user ID, you exempt the user ID from containment. Before using this option, ensure that the user ID is trusted and unlikely to be compromised. Consider

using the NEVERCONTAIN attribute for user IDs that are assigned to servers and critical system services to avoid the impact of incorrectly assigning them the CONTAIN attribute.

## Requirements

The RACF address space must be active for user IDs to be added to or removed from containment. After a user ID is placed in the containment list, the user's access is denied regardless of whether the RACF address space is active.

## Restrictions

Up to 100 user IDs can be assigned to the containment list, which is saved for the current IPL. After IPL, the list is empty. User IDs contained prior to the IPL remained contained across IPLs (their associated profiles retain the CONTAINED attribute), but the user IDs are no longer included in the active list. Attempts to add more than 100 users to the containment list will fail with the users being revoked, but still able to continue active sessions with normal access until completion.

If your installation reaches the 100 user ID limit, it is recommended that your staff review the list of contained user IDs by using the SETROPTS command. Determine why so many user IDs are included and act to remediate the cause. Update any user IDs that should not be contained by setting them to NOCONTAIN, which will free up additional entries in the user containment list. Then enter ALTUSER with CONTAIN for the original, failing user ID to add it to the list.

## Coexistence consideration for using a shared RACF database

Systems with the user ID containment support will automatically honor user ID containment functions. Systems at an earlier level ignore the user containment settings in the RACF database.

Coexistence support is required on z/OS 2.5 to ensure that containment settings are maintained when a shared database is used. Coexistence for z/OS 2.5 is provided when you install the PTF for APAR OA67786.



---

## Chapter 4. Defining groups

The group structure of RACF can be mapped to the organizational structure that exists at your installation. That is, RACF conforms naturally to a tree structure of groups, where each group except SYS1, which is predefined as the highest group, has a superior, or owning group. Groups can correspond directly to business entities such as divisions, departments, and projects. Users can be connected to one or more groups.

---

### Types of groups

When you define a group, you should consider the basic purpose of the group. Is it an administrative group, a holding group, a data control group, a functional group, or a user group?

When setting up RACF groups, keep in mind that the maximum number of users that you can connect to any one group is approximately 5900. See the topic on determining storage requirements for profiles in *z/OS Security Server RACF Macros and Interfaces* for information about how to determine the exact maximum number.

For groups that might become large, and for which a quick listing of members is not needed, you might want to consider defining the groups using the UNIVERSAL operand of the ADDGROUP command. Universal groups might be appropriate for holding groups and other types of groups. See [“Defining large groups with the UNIVERSAL attribute”](#) on page 88.

### Administrative groups

You can create a group simply as an administrative convenience. For example, you might create a group to represent an organizational entity, such as a region or a division.

With RACF delegation, you can create this kind of group for each group administrator. Operating from such groups, the group administrators can then define other groups needed by their local users.

### Holding groups

A popular technique that retains user definition centrally, yet allows the effective use of group administrators, is to establish a *holding group*. You define all users centrally and initially connect them to a group named HOLD with the minimum of authorities. HOLD does not appear in any access lists, and therefore has no real significance to the user.

Group administrators, to whom you give CONNECT (but not JOIN) authority, can connect the appropriate users to the groups under their control and change the users' default group name as appropriate. This technique allows the installation to assign correct account numbers and control other installation considerations while allowing flexibility in the grouping of the user population.

**Note:** A group cannot contain more than approximately 5900 users. Therefore, if you have more than this number of users, you cannot assign them to a single holding group. Also, you should be aware that extremely large groups can have performance implications for the RACF database. For more information, see [z/OS Security Server RACF System Programmer's Guide](#).

### Data control groups

You can create a group to act as a control point for the protection of data. For example, by using the group SYS1, you can determine which users are permitted to protect the SYS1 data sets. Only users with CREATE authority or higher in this group can protect system data sets. At your location, you or your delegate might consider defining one such group for every high-level qualifier representing data that is to be protected.

For more information, see [“Protecting group data sets”](#) on page 149.

## Functional groups

A group can represent a functional area of the installation for the purpose of data sharing. For example, a financial analyst might need to access a variety of resources across many groups, such as accounting, payroll, marketing, and others. Of course, the owners of each resource could permit the financial analyst to access their resources by placing the analyst's user ID on an access list. But if a new financial analyst takes over the job, it is then necessary to add the new user ID to each RACF profile. Likewise, the RACF profiles must be updated when the analyst no longer has a need to access the data. This arrangement involves a great deal of unnecessary activity by the resource owners.

Instead, you can create a group that represents the financial analyst function and permits access to the data defined to the group. Access to the entire range of data can then be managed by controlling the user population in the defined group. For those cases involving *one-time* access, owners of the needed data would simply PERMIT access by the defined group. Where appropriate, the group name could be included in profile access lists to ensure automatic availability of needed data to the financial analyst group. New financial analysts could be connected to the group, as required, to gain access to the entire range of data. Likewise, analysts could be removed from the group whenever necessary. By controlling the user population of such a functional group, resource profile changes on a day-to-day basis become unnecessary.

## User groups

You can define a group to serve as an anchor point for users who otherwise have no common access requirements. For example, engineers and scientists, as well as other problem-solving users, might have no need to access application-related data in the system. Their only interest might be in their own personal data. You can place this set of users in a single group that has no access to other data.

Also, you can define groups based on access level. For example, if PAY . DATA is a RACF-defined data set, two groups could be defined, PAYREAD and PAYUPDTE, both of which would appear in the PAY . DATA access list, but with READ and UPDATE access, respectively. Any users requiring access would be connected, as appropriate, by the group administrator.

## Group profiles

---

When you define a group to RACF, you create a group profile in the RACF database. A group profile consists of *segments*: a base segment and optionally, CSDATA, DFP, OMVS, OVM, and TME segments.

Each segment of a group profile consists of *fields*. When you define a group's profile (using the ADDGROUP command) or change a group's profile (using the ALTGROUP command), you can specify the information contained in each field of each segment. To define or change information in a non-base segment of a group profile, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access control.

You can list the contents of an entire group profile or the contents of individual segments of the group profile using the LISTGRP command. To display information in a non-base segment of a group profile, you must have the SPECIAL, AUDITOR or ROAUDIT attribute or at least READ authority to the segment through field-level access control.

For more information, see [“Field-level access checking”](#) on page 200 and [“Controlling access to the DFP segment”](#) on page 501.

For information on how to use the ADDGROUP, ALTGROUP, and LISTGRP commands, see [z/OS Security Server RACF Command Language Reference](#).

## The Base segment in group profiles

The base segment of a group profile contains basic information that is needed to define a group to RACF. You can specify the following information in the base segment:

***groupname***

Name of the group

**OWNER**

Owner of the group's profile

**SUPGROUP**

Name of the group's superior group

**TERMUACC or NOTERMUACC**

For RACF-protected terminals: Indicates whether to allow access based on the UACC of the terminal profile

**DATA**

Installation-defined data

**MODEL**

Name of the profile used as a model for new group data set profiles, either generic or discrete

**UNIVERSAL**

Universal group

See [z/OS Security Server RACF Command Language Reference](#) for information about the authorization required to create, change, or view information in the base segment.

## The CSDATA segment in group profiles

You can define a CSDATA segment for group profiles. The CSDATA segment contains installation-defined data related to custom fields that your installation has defined. For details about defining custom fields, see [Chapter 26, “Defining and using custom fields,”](#) on page 637.

To define or change information in the CSDATA segment of a group profile, you must have the SPECIAL attribute or at least UPDATE authority to the segment by way of field-level access control. For details, see [“Authorizing users to update data in a custom field”](#) on page 645. To display information in the CSDATA segment of a group profile, you must have the SPECIAL attribute or at least READ authority to the segment by way of field-level access control.

## The DFP segment in group profiles

You can define a DFP segment for group profiles. The DFP segment contains default values that DFP uses to determine data management and DASD storage characteristics for SMS-managed data sets. You can specify the following information in this segment:

**DATAAPPL**

Group's DFP data application identifier

**DATACLAS**

Group's default data class for attributes used during allocation of all new data sets

**MGMTCLAS**

Group's default management class for attributes used in managing a data set after it is allocated

**STORCLAS**

Group's default storage class for logical storage attributes

To define or change information in the DFP segment of a group profile, you must have the SPECIAL attribute or at least UPDATE authority to the segment by way of field-level access control. To display information in the DFP segment of a group profile, you must have the SPECIAL attribute or at least READ authority to the segment by way of field-level access control. For more information, see [“Controlling access to the DFP segment”](#) on page 501.

## The OMVS segment in group profiles

You can use the OMVS segment of the group profile to specify information about the group's z/OS UNIX group. Specifically, when you define a new z/OS UNIX group or change z/OS UNIX attributes for an existing group, you can specify the following information in the group's profile:

### GID

The group's z/OS UNIX group identifier

To define or change information in the OMVS segment of a group profile, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access control.

To display information in the OMVS segment of a group profile, you must have the SPECIAL attribute or at least READ authority to the segment through field-level access control.

You can authorize users to access to the OMVS segment of a group profile using the GROUP.OMVS.\* or the GROUP.OMVS.GID profile in the FIELD class.

When a GID is assigned to a group, all users connected to this group as their default group who have an z/OS UNIX user identifier (UID) in their user profile can use z/OS UNIX functions and can access z/OS UNIX files based on the GID and UID values assigned. If a user's current connect group is not their default group, a GID must also be assigned to the current connect group.

For more information, see [“Defining group identifiers \(GIDs\)” on page 513](#).

## The TME segment in group profiles

You can define a TME segment for group profiles. You can specify the following information in this segment:

### ROLES

A list of role profiles that refer to this group

To define or change information in the TME segment of a group profile, you must have the SPECIAL attribute or at least UPDATE authority to the segment through field-level access control.

To display information in the TME segment of a group profile, you must have the SPECIAL attribute or at least READ authority to the segment through field-level access control.

The TME segment fields are intended to be updated only by the Tivoli® product, which manages updates, permissions, and cross references. Security administrators should directly update TME fields only on an exception basis.

## Defining large groups with the UNIVERSAL attribute

If you are planning to create some groups that might become large and are unlikely to be deleted, such as groups that contain all users, you might define them as universal groups using the UNIVERSAL operand of the ADDGROUP command. *Universal groups* are user groups that do not have complete membership information stored in their group profiles. The benefit is that there is no limit on the number of group members. However, you cannot list all the members using the LISTGRP command.

The UNIVERSAL attribute can be defined for a group only at group creation time. The UNIVERSAL attribute cannot be added or removed using the ALTGROUP command. The output of the LISTGRP command contains the UNIVERSAL attribute for universal groups.

**Rule:** Do not change the TERMUACC setting for a universal group after users have been connected to the group. The change is not reflected in the profiles of all of the connected users.

The group profile of a universal group contains limited group membership information. Only users who have group authorities other than USE, and those who have the group-SPECIAL, group-OPERATIONS or group-AUDITOR attribute, are added to the member list. Therefore, the output of the LISTGRP command will not provide a complete list of group members. The best way to list the members of a universal group is using the output of the database unload utility. See [“Using IRRDBU00 with universal groups” on page 372](#). For information about sample RACFICE reports available to assist you, see [“Reports based on the database unload utility \(IRRDBU00\)” on page 378](#).

If you want to delete a universal group, you should use the remove ID utility (IRRRID00) to delete the group and remove all member connections. See [“Processing universal groups” on page 403](#) for information about using IRRRID00 to delete universal groups.



## Group naming conventions

The naming conventions for groups are relatively simple:

- A group name must be 1 - 8 characters in length, chosen from the uppercase letters (A - Z), numbers (0 - 9), or # (X'7B'), \$ (X'5B'), or @ (X'7C'). It must not start with a number.

**Tip:** The #, \$, and @ characters might be displayed differently on terminals outside the United States. Therefore, use the characters with the hexadecimal equivalents shown above.

- No two groups can have the same name. No group name can be the same as a user ID.

Because two or more users might want to use the same group name (for example, ADMIN), you should adopt naming standards locally to prevent conflicts. For example, consider assigning a unique 1- or 2-character group name prefix to each group administrator. Then each group defined by a group administrator would have a name that consists of the administrator's prefix followed by whatever characters the administrator chooses to use. This prefixing ensures that two group administrators cannot use the same group name.

For information about group naming conventions for z/OS UNIX group identifiers (GIDs), see [“Defining group identifiers \(GIDs\)”](#) on page 513.

## Benefits of using RACF groups

Some of the benefits of using RACF groups include:

- Reducing the effort to maintain access lists
- Avoiding the need to refresh in-storage profiles
- Providing a form of timed PERMIT
- Minimizing the length of access lists

**Note:** For detailed information, see [“Limiting the size of your access lists”](#) on page 193.

## Reducing the effort of maintaining access lists

Using RACF groups can reduce the time you spend administering access to resources.

Instead of adding and deleting users to the access lists of several profiles, you can create a RACF group and place it on the access list instead of the user IDs. Then, you can give CONNECT group authority in that group to an appropriate person (perhaps the owner of the resource profiles). That person can then change the membership of the group (through the use of the CONNECT and REMOVE commands) instead of issuing the PERMIT command many times to change the access lists of all of the affected resource profiles.

## Avoiding the need to refresh in-storage profiles

If your installation maintains in-storage copies of resource profiles through the SETROPTS RACLIST or SETROPTS GENLIST command, changes to those profiles do not take effect on the system until a SETROPTS RACLIST REFRESH or SETROPTS GENERIC REFRESH command is issued.

For the access list of an in-storage profile that requires frequent maintenance, you might avoid refreshing the in-storage copy by adding a RACF group instead of individual user IDs to the access list. When you connect or remove a user from a RACF group, group membership takes effect at the user's next logon. Therefore, you can use the CONNECT and REMOVE commands (rather than the PERMIT command) to more quickly change the access authorities of an in-storage profile when you connect or remove users from a group already on the profile's access list.

**Note:**

1. If a user who is already logged on to the system is added to a RACF group with the CONNECT command, the user must log off and log on again before using the group authority to access resources in classes that have been RACLISTed.

2. If a user who is already logged on to the system is deleted from a RACF group with the REMOVE command, the user must log off and log on again before accessing resources in classes that have been RACLISTed without using the group authority.
3. If the user ID is associated with a started procedure, such as JES2, you must stop and restart it to use the new authority.

In addition, you can delegate the ability to maintain the membership of the RACF group to someone else because SPECIAL authority is not needed to use the CONNECT and REMOVE commands. Give CONNECT authority for the group to an appropriate person (perhaps the owner of the resource profile) and allow her to administer the access list of the affected resource profile without involving a SPECIAL user to refresh the in-storage profile.

## Providing a form of timed PERMIT

You can allow a user to access a protected resource for a limited time by taking the following steps:

1. Ensure that the only access the user has to the resource is by virtue of the fact that the user is connected to a RACF group that has the desired access to the resource. (List the appropriate resource profiles to check for the user's user ID, or other groups to which the user is connected, in the access list. Also, list the user's RACF user profile to check for the OPERATIONS or group-OPERATIONS attribute. Depending on the class of the resource, the OPERATIONS attribute might allow the user to access the resource.)
2. Connect the user to the group with a resume or revoke date. To cause the user's access to stop on a certain date, enter:

```
CONNECT userid GROUP(groupname) REVOKE(date)
```

To cause the user's access to start on a certain date, enter:

```
CONNECT userid GROUP(groupname) RESUME(date)
```



**Attention:** If the user's membership in the group allows the user to create profiles, and the user becomes the OWNER of such profiles, the user might still have access to the profiles after the revoke date.

## Group ownership and levels of group authority

The following topics discuss group ownership, group authorities, suggestions for assigning group authorities, and the group terminal option.

### Ownership of a RACF group

Each group that you define to RACF must be owned by a RACF-defined user or by its superior group. You assign ownership of a group with the ADDGROUP command when you create a new group profile or with the ALTGROUP command when you change an existing group profile. If you are the owner of a group (or if you are a connected user who has the group-SPECIAL attribute), you have the authority to:

- Define new users to RACF (provided you also have the CLAUTH attribute for the USER class)
- Connect and remove users from the group
- Delegate and change group authorities and set the default UACC for all new resources belonging to members of the group
- Modify, list, and delete the group profile
- Define, delete, and list the names of the subgroups under the group
- Specify the group terminal option

**Note:** Ownership of a group by a user does not allow that user to update the access lists of resource profiles owned by the group.

For a list of the RACF commands that group owners can issue, see [Table 50 on page 704](#).

## Group ownership of profiles

You can assign a *RACF group* as the owner of a user profile, group profile, data set profile, or general resource profile. In this way, profile ownership can remain constant, regardless of how often users change jobs in your organization.

Any user connected to the owning group who has the group-SPECIAL attribute has the authority of SPECIAL for all profiles owned by the group (see [“User attributes” on page 57](#)) and also has the ability to perform all owner functions for the group.

You can assign any group to be the owner of a profile. (A group profile must be owned by either a user or its superior group.) An owning group does not need to be a group to which a user (represented by the profile) is connected. Being able to assign any group as an owner gives you flexibility in defining an authority structure. For example, you could establish one group for the sole purpose of owning user profiles, and give a group administrator the group-SPECIAL and CLAUTH (for the USER class) attributes in that group.

## Group authorities

Each user in a group requires a level of group authority for that group. If a user is connected to several groups, the user has a level of group authority for each group. The group authorities are described in [Table 7 on page 91](#).

Table 7. Group authorities

Authority	Functions permitted	RACF commands permitted
USE	A user with the USE group authority can enter the system under control of that group, and can access resources (such as data sets, terminals, and others) to which the group is authorized.	LISTDSD, RLIST
CREATE	A user with the CREATE group authority can allocate new group data sets, RACF-protect group data sets, and control access to the profiles he or she has created. However, unless the user has access other than the CREATE group authority itself, the user cannot delete the data sets.  CREATE group authority includes the privileges of USE group authority.	ADDSD command for group data set profiles (all operands except NOSET)
CONNECT	A user with CONNECT group authority can connect users who are already defined to RACF to the group and assign USE, CREATE, or CONNECT group authority to users in the group.  CONNECT group authority includes the privileges of both the USE and CREATE group authorities.	All of the preceding, plus: <b>ALTUSER</b> GROUP, AUTHORITY, or UACC operands only <b>CONNECT</b> All operands except SPECIAL/ NOSPECIAL, OPERATIONS/NOOPERATIONS, AUDITOR/NOAUDITOR <b>LISTGRP</b> Groupname operand only <b>REMOVE</b> All operands

Table 7. Group authorities (continued)

Authority	Functions permitted	RACF commands permitted
JOIN	<p>A user with JOIN group authority can define new users and groups to RACF and assign any level of group authority to new users (including the JOIN authority). To define new users, the user with JOIN authority must also have the CLAUTH user attribute for the USER class. When a user defines a new group, it becomes a subgroup of the group in which the user has JOIN authority.</p> <p>JOIN authority includes the privileges of the USE, CREATE, and CONNECT group authorities.</p>	<p>All of the preceding, plus:</p> <p><b>ADDGROUP</b> All operands</p> <p><b>ADDUSER</b> All operands except OPERATIONS, SPECIAL, AUDITOR and ROAUDIT</p> <p><b>ALTGROUP</b> SUPGROUP operand only (to change the superior group of a group, a user must have JOIN authority in one group and be the owner of or be connected with the group-SPECIAL attribute to another group)</p> <p><b>DELGROUP</b> All operands</p> <p><b>LISTGRP</b> Groupname operand only</p>

For a list of the RACF commands that the group authorities allow users to issue, see [Table 48 on page 702](#).

## Suggestions for assigning group authorities

As a security or group administrator, you can create different types of administrative structures, according to how you assign group authorities and group ownership. Two examples of possible structures are total and partial delegation.

### Total delegation

With total delegation, one delegate (group owner) is responsible for the administration of a group, the users in the group, and the group resource profiles. In this scheme, the group owner connects to the group with JOIN authority, defines the group resource profiles, and connects other users to the group with USE authority.

### Partial delegation

With partial delegation, you can share the responsibility for the administration of a group, the users in the group, and the group resource profiles. Under this scheme, the owner of the group connects one user to the group with JOIN authority and this user connects other users to the group, giving CREATE authority to one user and USE authority to all other users. In this way, the owner of the group can monitor the group, the user with JOIN authority can monitor the users in the group, and the user with CREATE authority can create and maintain the group's data set profiles.

Another way to share administration responsibilities for the group, the users, and the group's resources is as follows: the owner of the group connects one user to the group with CREATE authority and all other users with USE authority. The owner of the group can then monitor the group and the users in the group, and the user with CREATE authority can define and control group data sets.

## The group terminal option

The group administrator (that is, the owner of a group) can specify a group terminal option for the group by using the ALTGROUP command with the NOTERMUACC operand. With this option, users of the group are authorized to log on to TSO only from those RACF-protected terminals to which they have been specifically authorized access by the PERMIT command. That is, users of the group might not be authorized to log on to TSO from terminals (either RACF-defined or otherwise) based on the universal access authority of the terminals.

## Summary of steps for defining a RACF group

This summary presents the steps required by RACF for defining a RACF group. Your installation might require additional steps, depending on your security policy.

### 1. Prepare to create the group profile as follows:

- Decide which group is to be the superior group.
- Decide the group name. (This cannot be the same as a user ID.)
- Decide who (a user or RACF group) is to be the owner of the new group. (If the group owner is a user, give him or her the information needed to manage the group.)
- Decide if the group should be a universal group.
- If your installation is using RACF to protect terminals, and the users in this group are terminal users who are to be limited to specific terminals, consider specifying the NOTERMUACC operand on the ADDGROUP command.
- If DFSMSdss is in use, work with the storage administrator to set the initial values in the group's DFP segment.

### 2. Create the group profile using the ADDGROUP command.

For example, to create a group for Department A called DEPTA whose owner and superior group is to be a group called ALLDEPT, enter:

```
ADDGROUP DEPTA OWNER(ALLDEPT) SUPGROUP(ALLDEPT)
```

### 3. Connect appropriate users to the new group.

- Most users should have USE group authority.
- A few users might need a group authority higher than USE group authority (such as CONNECT).

For example, to connect department members SUE, LIZ, and GENE to the DEPTA group and also give LIZ and SUE authority to add new users to the group, enter:

```
CONNECT (SUE LIZ) GROUP(DEPTA) OWNER(DEPTA) AUTHORITY(CONNECT)
CONNECT GENE GROUP(DEPTA) OWNER(DEPTA)
```

These commands assign ownership of each connection to group DEPTA rather than to the issuer of the CONNECT command (the default). Because GENE's authority defaults to USE, GENE can use any of the resources (for example, data sets) that belong to group DEPTA.

### 4. If the group is to own group data sets, do the following:

- Create a *top* generic profile for the group data sets in the DATASET class. For example:

```
ADDSD 'DEPTA.**' UACC(NONE)
```

- If appropriate, assign the GRPACC user attribute to a member of the group. (However, before assigning the GRPACC user attribute, see [Table 17 on page 198](#).)

### 5. If the group requires access to RACF-protected resources, give the group the required access using the PERMIT command. For example:

```
PERMIT 'RACF.PROTECT.DATA' ID(DEPTA) ACCESS(READ)
```

### 6. If the group requires access to z/OS UNIX resources, alter the profile to include an OMVS segment with an z/OS UNIX group identifier (GID). For example:

```
ALTGROUP DEPTA OMVS(GID(100))
```

## Summary of steps for deleting groups

This summary presents the steps required by RACF and related IBM licensed program products to delete groups from RACF. Your installation might require additional steps, depending on your security policy and the products you have installed.

1. Determine if the group is a universal group by using the LISTGRP command, and look for the UNIVERSAL attribute.
2. If the group is not a universal group, use the output of the LISTGRP command to list the members and remove them from the group.

You can use the REMOVE command to do this. If the user is the owner of any group data set profiles, specify the new owner on the OWNER operand of the REMOVE command. Before removing a user from the user's default connect group, you must first connect the user to a new group (CONNECT command), and then change the user's default connect group to the new group (ALTUSER command).

3. If the group is a universal group, use the remove ID utility (IRRRID00) to remove all members from the group.

The LISTGRP command might not list all members of a universal group. For information, see [“Processing universal groups” on page 403](#)

4. Find all data sets associated with this group (that is, the group name is the high-level qualifier of the data set name) and perform the following steps:
  - a. Delete or rename (with a new high-level qualifier) the group's group data sets. If you rename or delete a data set that is protected by a discrete profile, the discrete profile is also renamed or deleted.

**Tip:** You can do this using the DATA SET LIST utility of ISPF.

- b. Identify all of the remaining (generic) data set profiles, create new ones modeled on them if needed, and delete the remaining profiles.

**Important:** Make sure that you do not delete an old profile unless it is no longer needed.

**Tips:**

- i) You can use the following SEARCH command to identify the group's data set profiles:

```
SEARCH MASK(groupname.) CLIST('LISTDSD DA(' ' ) ALL')
```

As specified, the CLIST operand generates a CLIST that you can run to list all of the information in the data set profiles. This can help you assess whether to use the profiles as models.

- ii) You can use the FROM operand on the ADDSD command to create new profiles modeled on the old profiles.

5. To research the following steps, use the IRRRID00 utility to list the occurrences of the group name in the RACF database. For information, see [“Using the RACF remove ID \(IRRRID00\) utility” on page 391](#).
6. For each subgroup of the group to be deleted, change the subgroup's superior group to an existing group.

```
ALTGROUP subgroupname SUPGROUP(new-superior-groupname)
```

7. If the group is the owner of any profiles (the group's group name was specified on the OWNER operand), change the owner of the profiles to a new group or user.

**Tip:** Use the appropriate command for changing profiles, such as ALTUSER, ALTGROUP, ALTDSD, or RALTER.

8. Remove the group from any access lists in which the group's group ID is specified.

**Tip:** Use the DELETE operand on the PERMIT command.

9. After all occurrences of the group name are deleted from the RACF database, use the DELGROUP command to delete the group profile.

## Chapter 5. Classifying users and data

This topic contains an overview of using security levels, categories, and labels to classify users and data.

**Reference:** For detailed information and procedures for implementing security levels, categories, and labels to achieve a multilevel-secure environment, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

### Security classification of users and data

Security classification of users and data allows installations to impose additional access controls on sensitive resources. Each user and each resource can have a security classification in its profile. You can choose among the following:

- Security levels, security categories, or both
- You can use security labels, which are a combination of security levels and security categories, and are easier to maintain

A *security level* (SECLEVEL) is an installation-defined name that corresponds to a numerical security level (the higher the number, the higher the security level).

A *security category* (CATEGORY) is an installation-defined name that corresponds to a department or an area within an organization in which the users have similar security requirements.

A *security label* (SECLABEL) is an installation-defined name that corresponds to a security level and zero or more security categories.

This topic discusses security levels and security categories first. Security labels are discussed later (see [“Understanding security labels”](#) on page 98).

### Effect on RACF authorization checking

For RACROUTE REQUEST=AUTH access checking, security classification processing takes place after global access checking (if active), but before RACF checks the standard access list. If global access checking does not allow access to the resource, RACF does security classification processing for any resource that is protected by a profile that has security category or security level data. (For information on global access checking, see [“Setting up the global access checking table”](#) on page 194. For a complete list of the sequence of checks that RACF makes to grant or deny access to a resource, see [“Authorization checking for RACF-protected resources”](#) on page 718.)



**Attention:** Because RACF performs global access checking before many of the other kinds of access authority checks, such as security label checking or access list checking, global access checking might allow access to a resource you are otherwise protecting. To avoid a security exposure to a sensitive resource, do not create an entry in the global access checking table for a resource protected by a profile that contains a security level, security category, or security label (if the security label in the profile is SYSLOW, a global access checking table entry with an access authority of READ can be created). See [“Authorization checking for RACF-protected resources”](#) on page 718.

### Security levels and security categories

Security classification processing consists of a two-step checking process that occurs when RACF is processing an authorization request. (Note that the SECDATA class must be active, the SECLABEL class must not be active, and the protecting resource profile must have security levels or security categories.)

1. RACF compares the security level of the user with the security level of the resource. If the resource has a higher security level than the user, RACF denies the request.



For a terminal session, the security level that RACF uses for the user is the lower of the user's SECLEVEL and the terminal's SECLEVEL. Thus if the terminal has a SECLEVEL of 50 and the user has a SECLEVEL of 100, the user cannot access, through that terminal, any data that has a SECLEVEL of over 50.

2. RACF compares the list of security categories in the user's profile with the list of security categories in the resource's profile. If RACF finds any security category in the resource profile that is not in the user's profile, RACF denies the request. If RACF does not deny the request, RACF continues with authorization processing. If there are no categories in the resource profile, RACF continues with authorization processing.

## Security labels

Security label authorization checking is dependent on the concept of controlling user access to resources on the basis of three factors:

1. The sensitivity of the data that the resource contains
2. The user's authorization to access information at that level of sensitivity
3. The purpose for which the user is attempting to access the resource

The security administrator indicates the sensitivity of the data in the resource as well as the authorization of the user by assigning appropriate security labels in the resource or user profile.

Security label authorization checking involves comparing the user's security label with the security label of the resource. A user who lacks sufficient authorization is prevented from accessing information in the resource.

Three requested access levels are supported for security label authorization checking:

### Read-only

A user is attempting to read information from a resource.

#### Examples:

- TSO LISTBC command
- OPEN macro for READ

### Write-only

A user is attempting to write information to a resource (with no reading).

#### Examples:

- TSO SEND command (when the recipient of the message has a lower security classification than the sender)
- Writing a new entry in a z/OS UNIX directory

### Read-write

A user is attempting to access a resource for the purpose of both reading and writing.

#### Example:

- OPEN macro for WRITE

For detailed information, see [“Security label authorization checking” on page 733](#).

## Understanding security levels and security categories

When RACF is first installed, security classification of users and data is inactive. To use security levels and categories, activate the SECDATA class (but not the SECLABEL class).

You can choose to use one or both parts of security classification processing. To use security level checking, you must define a profile in the SECDATA general resource class with the name SECLEVEL. To use security category checking, you must define a profile in the SECDATA general resource class with the name CATEGORY. The installation names for security categories and security levels are then defined



as members of these profiles (in a manner similar to the global access table entries). You maintain the member entries by using the ADDMEM operand on the RDEFINE command and the ADDMEM and DELMEM operands on the RALTER command.

In the CATEGORY profile, the member entries are the names of the security categories. In the SECLEVEL profile, each member entry consists of a security level name followed by its associated security level number.

**Note:** You cannot define a SECLEVEL for a SECLEVEL profile in the SECDATA class. As a result, RACF does not perform security level checking when determining a user's authority to access a SECLEVEL profile. Also, if you issue the RLIST SECDATA SECLEVEL command to display a SECLEVEL profile, RACF does not display values in the SECLEVEL or CATEGORY fields of the profile.

## CATEGORY and SECLEVEL information in profiles

The RACF commands for users, data sets, and general resources allow you to define and maintain security classification information. Some examples of commands with security category and security level information follow. (For complete information on these commands, see [z/OS Security Server RACF Command Language Reference](#).) The examples assume that the SECLEVEL and CATEGORY tables shown earlier have been defined.

## Converting from LEVEL to SECLEVEL

Many installations use the LEVEL field for their own implementation of security-level checking. To convert these profiles to use SECLEVEL instead, installations can use the SEARCH command to search for profiles that have a specified value in the LEVEL field. Installations can use the SEARCH command with the CLIST option to locate all data set profiles with certain LEVEL values, and convert them to use SECLEVEL instead.



**Attention:** Before converting from the use of LEVEL to SECLEVEL, all user profiles must have the appropriate SECLEVEL values (if the SECDATA class is activated).

## Deleting UNKNOWN categories

If you delete a member from the CATEGORY profile, and that category is still specified in resource profiles, the resource profile listing (produced by the RLIST command, for example) shows an UNKNOWN category. To delete this category, enter the RALTER command with no category specified on the DELCATEGORY operand:

```
RALTER classname profile-name DELCATEGORY
```

To search for such profiles, enter the search command as follows:

```
SEARCH CLASS(classname) CATEGORY
```

## Maintaining categories in an RRSF environment

RACF assigns an internal value to each category that you define using the RACF commands. The internal value is not displayed when the CATEGORY profile is listed using the RLIST command. If you use an application program to issue a RACROUTE REQUEST=DEFINE, ICHEINTY, or RACROUTE REQUEST=EXTRACT, TYPE=REPLACE macro that specifies internal CATEGORY values, and you use RRSF to keep your RACF databases synchronized, you must ensure that each CATEGORY has the same internal value assigned to it on each of the RACF databases. Use the RACF database unload utility (IRRDBU00) to unload the databases and check the CATEGORY profiles in the SECDATA class. If the internal category values are different, you must delete the category information from all user and resource profiles, delete the CATEGORY profiles, then redefine the categories making sure that the command definitions occur in the same order on all systems. Once the CATEGORY profiles are identical on all systems, reassign the categories to users and resources. The SEARCH command with the CLIST option can be used to simplify this process.

## Understanding security labels

You can use *security labels* to associate a specific security level with a set of (zero or more) security categories. Security labels, when associated with resources, users, and jobs, provide the following advantages over security levels and security categories:

- Security labels can be assigned to data that is not necessarily protected by a resource profile. For example, spool files are assigned the security label of their creators. In many cases, data that has been assigned a security label retains that security label from the time the data is created until the data is deleted. For example, when a spool file is created by a user or job that is running under a security label, the spool file is assigned the security label of the user or job. The spool file retains that security label until the spool file itself is deleted (which can be long after the user logs off or the job ends).
- Users can log on with different security labels at different times but with the same user ID; without security labels, a user always has the same (default) security level and categories.
- Output printed for a user or job by Print Services Facility (PSF) for z/OS can have a PSF identification label related to the security label of the user or job printed on every page.
- It is easier to maintain the security classification of users and data (changing the definition of a security label affects all users and resources that have that security label; you need not make the same change for many different profiles as you would for security levels and categories).

## Comparing security labels

When authorization checks are made to determine security label authorization (for example during read-only, write-only, and read-write requests), the relationship between security labels is assessed. A relationship can occur between the security labels of two users or between a user and a resource. (For purposes of this explanation, examples will be drawn based on the relationship of the security label of a user and the security label of a resource.) The types of relationships are:

- Dominance
- Equivalence
- Disjoint

To be considered *dominant*, the user's security label must be greater than or equal to the security label of the resource. When dominance occurs, *both* of the following conditions are true:

1. The security level used to define the user's current security label is equal to or higher than the security level used to define the security label of the resource.
2. All of the categories (if any) used to define the security label of the resource are in the user's current security label.

Note that the security label of a resource can also *dominate* the security label of a user in the contrasting scenario.

To be considered *equivalent*, the user's security label must have the same definition as the security label of the resource. When equivalence occurs, *both* of the following conditions are true:

1. The security level used to define the user's current security label must be the same as the security level used to define the security label of the resource.
2. All of the categories (if any) used to define the security label of the resource are the same as the categories used to define the user's current security label.

To have equivalence, the names of the security labels do not have to be the same.

When security labels are *equivalent*, each security label can be said to *dominate* and be *dominated* by the other.

To be considered *disjoint*, the user's current security label and the resource security label must not be equivalent and neither one can dominate the other. When a disjoint occurs, *both* of the following conditions are true:

1. The set of security categories that defines the user's current security label includes only a subset, or none, of the security categories that define the security label of the resource.
2. The set of security categories that defines the security label of the resource includes only a subset, or none, of the security categories that define the user's current security label.

Note that a disjoint can occur in the relationship between the security labels of two users.

## Considerations related to security labels

There are several considerations for implementing security labels with RACF system-wide options. For details about implementing security labels, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

1. Once you activate the SECLABEL class, users who logon without a security label can no longer access resources protected by security labels. Therefore, you should assign a security label to all users before you assign a security label to a commonly accessed resource, such as SYS1.BROADCAST. Each user who does not have a default label must provide a security label at logon time or else be denied access to the system.
2. If you assign a security label to a resource profile while the SECLABEL class is active, the security label you assign to the users must allow the required users to access the resource.  
  
This also applies to the z/OS UNIX environment. When you create a z/OS File System (zFS) file system in a data set covered by a security label while the SECLABEL class is active, directories and files within that file system will be created with security labels. Only users with the appropriate security labels will be allowed to access those files and directories.
3. If your installation uses the SETROPTS MLACTIVE option, all data protected by classes that require security labels must have security labels. For more information, see [“Enforcing multilevel security \(MLACTIVE option\)”](#) on page 135.
4. If your installation uses the SETROPTS MLS(FAILURES) option, there are tighter restrictions on attempted accesses to resources, depending on the kind of access attempt and the security labels of the resource and user. For more information, see [“Security label authorization checking”](#) on page 733.
5. If your installation uses the SETROPTS MLS(FAILURES) option, the first data set written to a tape volume defines the security label of any data set that is later written to the tape. For more information, see [“Preventing the copying of data to a lower security label \(SETROPTS MLS option\)”](#) on page 134.
6. If your system includes products that do not support security labels when they invoke RACF, you should consider using the SETROPTS COMPATMODE option. See [“Activating compatibility mode for security labels \(COMPATMODE option\)”](#) on page 135.
7. If your installation uses the SETROPTS MLFSOBJ option, all files and directories must have security labels. No users (except trusted and privileged started tasks) will be able to access files and directories that do not have security labels. See [“Restricting access to z/OS UNIX files and directories \(MLFSOBJ option\)”](#) on page 137.
8. If your installation uses the SETROPTS MLIPCOBJ option, all resources related to interprocess communication must have security labels. No users (except trusted and privileged started tasks) will be able to access interprocess communication resources that do not have security labels. See [“Restricting access to interprocess communication objects \(MLIPCOBJ option\)”](#) on page 137.
9. If your installation uses the SETROPTS MLNAMES option, users cannot view the names of data sets, files, and directories that cannot be read from their current user security labels. Users are similarly restricted from seeing authorized resource names when they list catalogs and directories. This option is also called *name-hiding*. Note that if the SECLABEL class is not active while MLNAMES is active, data set names will still be hidden from users who do not have at least READ access to the data sets. See [“Using name-hiding \(MLNAMES option\)”](#) on page 138.
10. If your installation uses the SETROPTS SECLBYSYSTEM option, certain security labels can be activated on certain systems, based on the system identifiers specified in the SECLABEL class profile

for each security label. See [“Activating security labels by system image \(SECLBYSYSTEM option\)”](#) on page 138.

## How users specify current security labels

TSO users can override their default security label by specifying a value in the SECLABEL field on the logon panel or LOGON command. Batch users can also override their default security label by specifying the SECLABEL parameter on the JOB statement when a job is submitted to the system.

**Note:** When SECLBYSYSTEM is in effect, a batch job submitted with no security label executes with the security label of the JESINPUT class profile, unless the JESINPUT class security label is SYSMULTI.

When a TSO user logs on using the TSO/E logon panel, and specifies a value in the SECLABEL field on the TSO/E logon panel, TSO records the value from the SECLABEL field in the TSO segment of the RACF user profile (if the TSO segment exists). The next time the user logs on, TSO displays the value from this field in the SECLABEL field on the logon panel as a default. If the user changes the SECLABEL field while logging on, TSO modifies the SECLABEL field in the user's TSO segment with the new current security label. This new value is used as the default that is presented for the next logon.

A user can also be assigned a current security label based on their port-of-entry. For example, the security label of the port-of-entry (in this case, a terminal) overrides a TSO user's default security label if all of the following conditions are true:

1. The TSO user did not specify a security label.
2. The TERMINAL class is active.
3. The profile covering the terminal has a security label.

When you are migrating from security levels and security categories to security labels, consider setting the SECLABEL field using the ADDUSER and ALTUSER commands as follows:

```
ADDUSER userid SECLABEL(security-label)  
ALTUSER userid SECLABEL(security-label)
```

## Listing security labels

To display the security label stored in a resource profile, specify the ALL option on the LISTDSD and RLIST commands.

### Displaying the default security label for a user ID

To display a user's *default* security label (the security label stored in the profile using the SECLABEL operand on the ADDUSER or ALTUSER command), issue the LISTUSER command with the user ID specified. For example:

```
LISTUSER JONES
```

When you issue the LISTUSER command with any operand, such as the user ID, the default security label will be displayed.

### Displaying the current security label for a user ID

You cannot display another user's default security label. Only that user can display it. To display a user's current security label, ask the user to enter the LISTUSER command with no operands:

```
LISTUSER
```

When RACF displays a user's security label, RACF also displays the security level and any security categories that define it.

## Finding out which security labels a user can use

To find out which security labels a user can specify, enter:

```
SEARCH CLASS(SECLABEL) USER(userid)
```

**Note:** The SECLABEL class must be active when you execute this command.

## Searching by security labels

To search for all of the profiles that have a particular security label, enter:

```
SEARCH CLASS(classname) SECLABEL(security-label)
```

For example:

```
SEARCH CLASS(TERMINAL) SECLABEL(EAGLE)
```

This command displays all of the terminal profiles that have security label EAGLE specified.

```
SEARCH CLASS(USER) SECLABEL(EAGLE)
```

This command displays all of the user profiles in which security label EAGLE is the default security label.

### Restrictions:

1. Your results will be different if SECLBYSYSTEM is active.
2. You can search only one class at a time. If you do not specify a class, the DATASET class is searched by default.
3. To list all profiles, you must have SPECIAL, AUDITOR, or ROAUDIT authority. Otherwise, RACF lists only those profiles that you own, that have high-level qualifiers matching your user ID, or to which you have at least READ access authority.
4. If the SECLABEL class is active, RACF lists only the names of profiles that have security labels that are equal or lower level to that of the user's current security label.

## Restricting security label changes

You can restrict users who do not have the SPECIAL attribute from specifying security labels in resource profiles, or changing the definitions of security labels, by specifying the SECLABELCONTROL operand on the SETROPTS command. For more information, see “Restricting changes to security labels (SECLABELCONTROL option)” on page 133. When the SECLABELCONTROL option is not active, a user with sufficient authority to create or update a resource profile can specify the SECLABEL operand if the user has at least READ access authority to the associated security label.

## Requiring security labels

You can require that all work entering the system, including users logging on and batch jobs, have a security label assigned. For more information, see “Enforcing multilevel security (MLACTIVE option)” on page 135.

## Controlling the write-down privilege

When SETROPTS MLS is active in your environment, users are limited in their WRITE actions, such as their authority to copy data from a resource with one security label to a resource with a lower security label. If you need to allow certain users to have this authority, also called the *write-down privilege*, you can authorize them using a FACILITY class profile called IRR.WRITEDOWN.BYUSER.

**Restriction:** The authority to write down applies to actions on resources in classes defined in the CDT with neither the RVRSMAC nor EQUALMAC attribute. (Such classes are processed using normal MAC

processing.) For classes with the RVRSMAC attribute, the write-down privilege allows users to *write up*. For classes with the EQUALMAC attribute, this privilege has no effect.

Steps for controlling the write-down privilege

Perform the following steps to control and authorize users for the write-down privilege:

- 1. Define a resource called IRR.WRITEDOWN.BYUSER in the FACILITY class with UACC(NONE). To prevent all users from gaining the write-down privilege, do not permit any users or groups.

Example:

```
RDEFINE FACILITY IRR.WRITEDOWN.BYUSER UACC(NONE)
```

- 2. Identify which users require the write-down privilege and determine which level of access they require: READ or UPDATE authority. Both access authorities allow users to query, set and reset the write-down mode of their address spaces by executing a user command. The following user commands are available for this purpose:

For TSO/E users

RACF **RACPRIV** command. (See [z/OS Security Server RACF Command Language Reference](#) for syntax information.)

For z/OS UNIX users

z/OS UNIX **writedown** command. (See [z/OS UNIX System Services User's Guide](#) for syntax information.)

Use the following table to make your decision:

If the user requires the ability to enter write-down mode...	Then grant this access authority...
By default, upon entering the system, without executing the <b>RACPRIV</b> or <b>writedown</b> command	UPDATE
Only explicitly, by executing the <b>RACPRIV</b> or <b>writedown</b> command each time to set or reset the privilege	READ

- 3. Authorize users for the write-down privilege by adding those users, or one of their groups, to the access list with either READ or UPDATE authority, based on the users' requirements.

Example:

```
PERMIT IRR.WRITEDOWN.BYUSER CLASS(FACILITY) ID(users or groups) ACCESS(READ)
PERMIT IRR.WRITEDOWN.BYUSER CLASS(FACILITY) ID(users or groups) ACCESS(UPDATE)
```

- 4. Refresh the FACILITY class to activate your changes.

Example:

```
SETROPTS RACLIST(FACILITY) REFRESH
```

When SETROPTS MLS is active, refreshing the FACILITY class also causes ACEEs to be purged from the VLF.

## Planning considerations for security labels

For details about implementing security labels for a multilevel-secure environment, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

Even though a single user can use more than one security label, it might be easier to assign each person a separate user ID for each security label used.

Also, some consideration should be taken before using resource profiles that, due to their naming structure, would either cause large numbers of resources to be protected by the same security label (for example: *userid.\**) or protect resources that need to be accessed when the user is logged on at different security labels (for example, a NAMES data set).

These types of resources can be grouped into several categories and following certain procedures can minimize the impact of their use. You might consider creating data set profiles with the following naming structure:

```
userid.security-label.*
```

For example, if SMITH needs to use security labels EAGLE and THRUSH, create profiles like:

```
SMITH.EAGLE.* UACC(NONE) SECLABEL(EAGLE)
SMITH.THRUSH.* UACC(NONE) SECLABEL(THRUSH)
```

Some services create new data sets based on the user's user ID. If the SETROPTS MLS option is in effect, authorization failures occur whenever the user uses a security label that is different from the security label that was in use when the data set was created. Applications that create such data sets should consider using the REXX RACVAR function package to determine the current security label for inclusion in the names of such data sets.

- **Read-only**

In general, these resources pose no problem if they are protected by a profile with the lowest security label that is used. By being protected in this manner, they can be accessed any time the user is logged on. For example, system resources that are read by all users should be protected with the SYSLOW security label.

- **Read mostly**

These resources must also be protected by a profile at the lowest security label that is used. This allows the user to access them for read any time the user is logged on. If the resource needs to be updated (for example: new names being added to the nickname file *userid.NAMES.TEXT*), the user must log on at his or her lowest security label in order to update the file.

- **Read/write but able to be preallocated**

Data sets such as the ISPF list and log data sets fall into this category. These data sets can be allocated from within a logon CLIST with a different data set used for each security label. It should be noted, however, that due to multiple data sets being used, updates to a data set at one security label would not cause updates to occur to a data set that is used for the same purpose at a different security label (for example, an SPF profile data set).

- **Data sets written to from multiple levels**

An example of this type of data set is the SPF EDIT recovery data sets. The CLIST ISREDRTI that creates the ISPF edit recovery table sets the default data set names for the recovery data sets to:

```
'&ZUSER.&ZAPPLID&ZEROS&I.BACKUP'
```

or for user ID RALPH to

```
'RALPH.ISR0001.BACKUP'
```

RACF has provided sample exits in SYS1.SAMPLIB that show how to solve the problem (data sets written to from multiple levels) for ISPF and PDF data sets.

A data set profile would cause a security label authorization failure when an attempt was made to write to the data set while logged on with a security label that was different from the one in the profile. Applications that create data sets in this manner can be used only if each user has access to only one security label.

When RACF checks a user's authority to use a terminal or console, or the authority of outbound work to use a JES writer, RACF uses "reverse MAC" (mandatory access checking). That is, the security label of the profile protecting the writer must be equal to or greater than the security label of the user or outbound work.



## Chapter 6. Specifying RACF options

This topic describes the RACF options you can specify to control how RACF operates on your system.

### Using the SETROPTS command

RACF provides many system-wide options for controlling the way it works on your system. You specify most of these options by issuing the SETROPTS command with the appropriate operands or filling in the appropriate ISPF panels.

This topic discusses SETROPTS options that are useful to the RACF security administrator. It assumes that you have the SPECIAL attribute.

For a description of the SETROPTS options that are useful to the RACF auditor (with the AUDITOR attribute), see *z/OS Security Server RACF Auditor's Guide*.

See *z/OS Security Server RACF Command Language Reference* for a complete description of the SETROPTS command.

Guidelines for using selected SETROPTS options:

- If you are installing RACF for the first time, activate enhanced generic naming:

```
SETROPTS EGN
```

- Do not issue the SETROPTS TERMINAL(NONE) command unless you have RACF-protected enough terminals so that users can log on. SETROPTS TERMINAL(NONE) prevents users from logging on to unprotected terminals.

To recover from such a situation, submit a batch job that runs under a user ID with the SPECIAL attribute and that issues SETROPTS TERMINAL(READ).

- Some classes have a default return code of 8. If such a class is activated, but no profiles are defined, user activity that requires access in that class is prevented.

Do not activate a class with a default return code of 8, either explicitly (by name) or implicitly (by means of a shared POSIT value), unless you have defined profiles for that class.

RACF prevents you from accidentally activating all classes by misusing the SETROPTS CLASSACT(\*) operand.

If security labels have been assigned to resource profiles, do not activate the SECLABEL class by using SETROPTS CLASSACT(SECLABEL) unless you have assigned appropriate security labels to appropriate users.

To recover from such a situation, log on as a user with the SPECIAL attribute, specifying SYSHIGH as the current security label. Then either assign security labels, or issue SETROPTS NOCLASSACT(SECLABEL).

- Do not issue the following SETROPTS commands unless you have assigned appropriate security labels to all users and to the resources that they must access:
  - SETROPTS MLACTIVE(FAILURES)
  - SETROPTS MLFSOBJ(FAILURES)
  - SETROPTS MLIPCOBJ(FAILURES).

To recover from such a situation, log on as a user with the SPECIAL attribute, specifying SYSHIGH as the current security label. Then, either assign security labels, or issue one of the following SETROPTS commands, as appropriate:

- SETROPTS NOMLACTIVE
- SETROPTS MLFSOBJ(ACTIVE)
- SETROPTS MLIPCOBJ(ACTIVE).

**Restriction:** The ISPF panels do not support all options of the SETROPTS command. For example, the SETROPTS option to activate and deactivate mixed-case password support is not available through the RACF panels. For information about using the SETROPTS command to implement mixed-case passwords, see [“Allowing mixed-case passwords \(PASSWORD option\)”](#) on page 106.

## SETROPTS options for initial setup

**Guideline:** Specify the options described in this topic when RACF is first installed.

### Allowing mixed-case passwords (PASSWORD option)

If you have the SPECIAL attribute, you can allow mixed-case passwords for all users on all applications on this system and on all systems that share the RACF database. Use the SETROPTS PASSWORD(MIXEDCASE) option to allow mixed-case passwords at your installation.

```
SETROPTS PASSWORD(MIXEDCASE)
```

**Restriction:** The ISPF panels do not support the SETROPTS option to activate and deactivate mixed-case password support. For this, you must use the SETROPTS command with the PASSWORD option.

By default, NOMIXEDCASE is in effect and mixed-case passwords are not supported. If you want to allow mixed-case passwords, be sure that mixed-case content is permitted by your password syntax rules. (See [“Establishing password syntax rules \(PASSWORD option\)”](#) on page 108.) When SETROPTS PASSWORD(MIXEDCASE) is in effect, the RACF commands ALTUSER, ADDUSER, PASSWORD, and RACLINK no longer translate passwords to uppercase, nor do applications that provide mixed-case password support, such as TSO/E and z/OS UNIX.

**User considerations:** When you activate the MIXEDCASE option, users should be aware of the following considerations.

- Mixed-case passwords are more secure and harder to guess than uppercase passwords. Users are encouraged to select mixed-case passwords.
- Users with existing, uppercase passwords need *not* supply their passwords in uppercase. However, once the MIXEDCASE option is activated, any password that is set or changed to a value containing a lowercase character must thereafter be supplied exactly as it was created. In other words, the user must then supply every character of the password using exactly the same case used when the password was created.
- Users are prevented from entering new passwords that differ from their current passwords by only the case in which they are entered. For example, if a user's current password is IM4JUVE, the user cannot change it to a new password of Im4Juve.

### Migration considerations for mixed-case passwords

Carefully plan your application updates and password rule changes before activating MIXEDCASE. Once MIXEDCASE is activated, subsequently issuing the SETROPTS PASSWORD(NOMIXEDCASE) command might cause unintended results. When you reset to NOMIXEDCASE, users who have mixed-case or lowercase passwords will be unable to enter the system until you reset their passwords.

If you use a mix of applications that do and do not support mixed-case passwords, do not activate the SETROPTS PASSWORD(MIXEDCASE) option.

If you have implemented RRSF, see [“RRSF considerations for mixed-case passwords”](#) on page 426.

### Allowing special characters in passwords (PASSWORD option)

If you have the SPECIAL attribute, you can allow a set of special characters to be specified in passwords for all users on this system and on all systems that share the RACF database. Use the SETROPTS PASSWORD(SPECIALCHARS) option to allow special characters in passwords at your installation.

```
SETROPTS PASSWORD(SPECIALCHARS)
```

**Restriction:** The ISPF panels do not support the SETROPTS option to activate and deactivate special character password support. For this, you must use the SETROPTS command with the PASSWORD option. Enabling special characters allows the following characters to be specified in RACF passwords.

Hexadecimal value	Symbol (using the EBCDIC 1047 code page)
4B	.
4C	<
4E	+
4F	
50	&
5A	!
5C	*
60	-
6C	%
6D	—
6E	>
6F	?
7A	:
7E	=

By default, NOSPECIALCHARS is in effect and special characters are not supported. If you want to allow special characters, be sure that they are permitted by your password syntax rules. Syntax rules can be created to require special characters.

The new password exit (ICHPWX01) can be used to further restrict this set when you have characters that are known to present problems with applications that you use.

**User considerations:** When you activate the SPECIALCHARS option, users should be aware of the following considerations.

- Special character passwords are more secure and harder to guess than uppercase passwords. Users are encouraged to select special characters.
- Certain characters might pose problems for certain applications. Avoid using such characters when possible.
- Certain characters have different character representations in different code pages. This might present problems when logging in with a different terminal than you normally use, for example, while traveling internationally. Avoid the use of such characters, when necessary.

**RRSF considerations for special characters in passwords::** Be careful when RRSF nodes do not have the same settings in effect for the special characters option of the SETROPTS PASSWORD command. This can occur when one of the nodes is a downlevel system that does not have support for APAR OA43999 applied, or when the nodes have differing settings in effect for the SPECIALCHARS option of the SETROPTS PASSWORD command. When this is the case, message IRR1006I is issued when the RRSF connection is established between the nodes.

The following rules apply when RRSF nodes do not have the same special character setting in effect and a password with special characters is propagated to a system that does not have support for APAR OA43999 applied, or on which special characters are not enabled:

1. The propagation fails when it occurs by using automatic command direction with the ADDUSER, ALTUSER, and PASSWORD commands.

2. The propagation succeeds when it occurs by using automatic password direction (with RACROUTE REQUEST=VERIFY/X, RACROUTE REQUEST=EXTRACT,TYPE=REPLACE, or ICHEINTY).
  - The user is able to LOGON with this password.
  - The user cannot change the password using the PASSWORD command if support for APAR OA43999 is not applied, but is able to if the support is applied, even if SPECIALCHARS is not enabled.
  - The user is able to change the password during LOGON.

**Guideline:** Apply support for APAR OA43999 where necessary and enable SPECIALCHARS at the same time on all RRSF nodes.

## Establishing password syntax rules (PASSWORD option)

If you have the SPECIAL attribute, you can establish up to eight password syntax rules to verify that new passwords meet the installation standards. These rules allow you to control:

- The minimum and maximum length of passwords
- The character content of installation-selected positions in the passwords

**Restrictions:** The password syntax rules you define are not enforced when users log on with their current passwords. Therefore, changes you make to your password syntax rules will not affect users with current passwords. Your changes will take effect for current users only when they change their passwords. For new users, the changes will take effect when the new user logs on for the first time. In addition, password syntax rules are not enforced when you define a temporary password for another user using the ALTUSER PASSWORD command unless you specify the NOEXPIRED option.

You establish these rules by using the RULEn suboperand specified by the PASSWORD operand of the SETROPTS command. The following example shows how you can establish a syntax rule for new passwords for your installation.

```
SETROPTS PASSWORD(RULE1(LENGTH(8) VOWEL(1,3,5:8) NUMERIC(2,4)))
```

For more information see PASSWORD(RULEn) of the SETROPTS command in [z/OS Security Server RACF Command Language Reference](#).

The command establishes syntax rule RULE1. Syntax rule RULE1 specifies that new passwords must be 8 characters in length, must contain vowels in positions 1, 3, 5, 6, 7, and 8, and must contain numbers in positions 2 and 4. Thus, the password A2E2EAE follows the rule, and C3DMIER5 does not.

If you do not define a value for every position specified by the LENGTH value, the undefined positions can contain any combination of alphanumeric characters.

**Tip:** If the RACF ISPF panels are installed, you might find them easier to use for setting up password syntax rules.

## Setting the maximum and minimum change interval (PASSWORD option)

If you have the SPECIAL attribute, you can specify the INTERVAL, PHRASEINT and MINCHANGE suboperands of the SETROPTS PASSWORD command. The INTERVAL suboperand specifies the system default for the maximum number of days that each user's password and password phrase remain valid. The PHRASEINT suboperand specifies the system default for the number of days that each user's password phrase remains valid and overrides the INTERVAL setting for password phrases. The MINCHANGE suboperand specifies the system default for the minimum number of days that must pass between a user's password (and password phrase) changes. The following example specifies that each user's password remains valid for 60 days (as long as the system default for these users remains 60 days), specifies that each user's password phrase remain valid for 365 days and that no user can change their password or password phrase more often than every 30 days (as long as the system default for these users remains 30 days).

```
SETROPTS PASSWORD(INTERVAL(60) PHRASEINT(365) MINCHANGE(30))
```

These values become effective immediately as:

- The default password change interval for new users whom you define to RACF through the ADDUSER command.
- The upper limit for users who specify the INTERVAL operand on the PASSWORD command.
- The effective password phrase interval for users who do not have a user specific phrase interval value set in their user profile.

The initial system default is 30 days for the maximum change interval (INTERVAL) and 0 days for minimum change interval (MINCHANGE). The value MINCHANGE(0) allows users to change their passwords and password phrases more than once each day.

The initial default for the password phrase interval (PHRASEINT) is 0 which indicates that the INTERVAL keyword is used to control the password phrase interval.

When users are defined to RACF and have access to the system, they can use the INTERVAL operand of the PASSWORD command to set their own change interval to a value less than 30 or to a value less than that which you specified on the INTERVAL operand of the SETROPTS command (if you did so).

#### Restrictions:

1. When you change the SETROPTS PASSWORD(INTERVAL) value, the password interval set in each user's profile is not changed. If a user's INTERVAL value in the user's profile (as set using the PASSWORD command) is different than the SETROPTS value, RACF expires the password or password phrase at the shorter interval of the two values.
2. When you change the SETROPTS PASSWORD(PHRASEINT) value, a password phrase interval set in each user's profile is not changed. When a user does not have a PHRASEINT value set the SETROPTS PHRASEINT value is used. When the user has a PHRASEINT value set it overrides the SETROPTS PHRASEINT value.
3. Avoid setting the MINCHANGE value higher than any individual user's INTERVAL value (as set using the PASSWORD command). If you do, RACF expires the user's password or password phrase when the MINCHANGE period elapses, not when the user's INTERVAL elapses. Users cannot change their own passwords or password phrases until the MINCHANGE period elapses, even when the user's INTERVAL value defines a shorter period than the MINCHANGE value.

**User consideration:** Users who attempt to change their passwords or password phrases before the minimum change interval elapses are notified of their change failures but are *not* notified of the reason. The reason for the failure is withheld in the event of unethical user behavior, particularly by outside users or hackers who might exploit the information.

## Extending password and user ID processing (PASSWORD option)

If you have the SPECIAL attribute, you can specify the WARNING/NOWARNING, HISTORY/NOHISTORY, and REVOKE/NOREVOKE options.

Use the PASSWORD option on the SETROPTS command to provide the following functions:

- **WARNING:** The WARNING suboperand enables you to specify that RACF should issue warnings about expiring passwords and password phrases.

When you specify WARNING, RACF issues a message each time a user logs on to TSO or submits a batch job with an expiring password or password phrase, beginning the specified number of days before expiration. The following example specifies that RACF issue a warning message 5 days before a password or password phrase expires:

```
SETROPTS PASSWORD(WARNING(5))
```

if NOWARNING is in effect, RACF does not issue a warning message before a password or password phrase expires.

- **HISTORY:** The HISTORY suboperand enables you to specify the number of previous passwords and password phrases (1 - 32) that RACF saves for each user and compares with an intended new value. When RACF finds a match with a previous value, or with the current password or password phrase, RACF rejects the new intended value.

For passwords, RACF stores only previous passwords in each user's history. For password phrases, RACF saves the user's current password phrase in addition to the user's previous password phrases. Therefore, for password phrases, RACF saves one less previous value than the number you specify for history.

**Example:** If you specify 12 for your HISTORY number, RACF saves up to 12 previous passwords and up to 11 previous password phrases for each user.

```
SETROPTS PASSWORD(HISTORY(12))
```

If you increase the HISTORY number, RACF saves and compares that number of passwords and password phrases to the new intended value. However, the higher number might not immediately take effect for a particular user, depending on how many times the user has changed their password or password phrase in the past. If you reduce the HISTORY number, any previous passwords and password phrases that are stored in the user profile more than the newly specified HISTORY number are not deleted and continue to be used for comparison. For example, if you specify 12 for your HISTORY number and later reduce it to 8, RACF compares the old passwords and password phrases 9 - 12 with the new intended value.

**Guideline:** If you change the HISTORY number, use the PWCLEAN operand of the ALTUSER command to reorganize password history so that it immediately accepts the new value. For more information, see [z/OS Security Server RACF Command Language Reference](#).

NOHISTORY specifies that new passwords and password phrases are compared only to the current password or password phrase. Any prior history information in the user profile is not deleted or changed.

- **REVOKE:** The REVOKE suboperand enables you to specify how many consecutive attempts to use incorrect passwords and password phrases RACF permits before it revokes the user ID on the next attempt.

**Example:** If you specify 4 for your REVOKE number, RACF allows four consecutive attempts to use incorrect passwords or password phrases to access the system. For example, three incorrect passwords followed by one incorrect password phrase are allowed. But a fifth attempt, with either an incorrect password or incorrect password phrase, revokes the user ID.

```
SETROPTS PASSWORD(REVOKE(4))
```

After RACF revokes the user ID, you can activate the user ID with the RESUME operand of the ALTUSER command if you have the SPECIAL or group-SPECIAL attribute, or are the owner of the profile. If SETROPTS NOREVOKE is in effect, consecutive incorrect passwords and password phrases are ignored.

Protected user IDs are not revoked based on consecutive incorrect passwords and password phrases. For more information, see [“Defining protected user IDs” on page 74](#).

By default, users with the RACF SPECIAL attribute can be granted an extra logon attempt. Here, RACF issues the following write-to-operator message:

```
ICH302D REPLY Y TO ALLOW ANOTHER ATTEMPT OR N TO REVOKE USERID userid.
```

Your installation can choose to suppress the ICH302D prompt and thus cause the user ID to be revoked without allowing an extra logon attempt. To do so, you must define a discrete profile in the XFACILIT class with the following name:

```
IRR.DENY.SPECIAL.USER.ADDITIONAL.PASSWORD.ATTEMPTS.APPL.appl-name
```

In the profile, *appl-name* must match the APPL= *value* on the RACROUTE REQUEST=VERIFY request. If no appl-name is specified on the REQUEST=VERIFY request, the appl-name defaults to the same value that is used in PassTicket application name derivation. Only the profile existence is checked; no profile attributes, such as UACC or access list, are considered.

If your installation suppresses the ICH302D prompt, it is recommended that you have a process in place to resume a SPECIAL user so that a user ID is accessible for performing RACF administration tasks.

## Revoking unused user IDs (INACTIVE option)

The INACTIVE operand of the SETROPTS command causes RACF to revoke the user's right to use the system if the user ID has remained unused beyond a specified number of days. RACF revokes the user the next time the user attempts to enter the system.

The following example specifies that RACF revoke a user ID if it is unused for over 30 days:

```
SETROPTS INACTIVE(30)
```

If you issue the SETROPTS INACTIVE(30) command and a user has not done any of the following in 31 days:

- Logged on
- Submitted a job
- Changed their password or password phrase by any method
- Used an incorrect password to attempt an unsuccessful logon to a remote system in the RRSF network
- Received a directed command or output from RACF remote sharing

that user is considered revoked. However, the user is not actually revoked and the output of the LISTUSER command does not show that the user is revoked until the user next attempts to log on or submit a job. When you allow the user to start using the system again (using the RESUME operand on the ALTUSER command), RACF resets the effective date with which the period of inactivity starts.

When you define a new user ID, the user's last access date is set to the user ID's creation date. If the user ID is not used within the number of days specified by SETROPTS INACTIVE, the user ID will be revoked. When you issue the LISTUSER for a new user ID that has never been used, the last access date will be listed as UNKNOWN.

**Note:** An auditor cannot rely upon LAST-ACCESS date for the last (if ever) authentication of a user. If an authentication request (RACROUTE REQUEST=VERIFY) includes STAT=NO, there is no indication when an id was last (if ever) authenticated. This is the case with Db2z/OS.

If NOINACTIVE is in effect, RACF does not check the user ID against an unused user ID interval.

If NOINITSTATS is in effect, the INACTIVE, REVOKE, HISTORY, and WARNING options cannot be used.

## Activating list-of-groups checking (GRPLIST option)

List-of-groups authority checking supplements the normal RACF access authority checking by allowing all groups of which a user ID is a member to enter into the access list checking process. This process replaces the checking that compares the current connect group with the resource's access list, and can expand a user's ability to access resources. If list-of-groups checking is active, then regardless of which group the user is logged on to, RACF recognizes the user's group-related authorities in other connect groups. If a user is in more than one group and tries to access a resource, RACF uses the highest authority allowed by the user's list of groups and the resource's access list.

**Note:** A user's current connect group is the group entered on the logon panel or with the LOGON command. If no group is specified at logon, the user's default group is used.

For example, the user is logged on to Group B (the current connect group) and tries to access a resource. The resource's access list does not contain the user's user ID or the group name for Group B, but it does contain the group name for Group A with an associated access authority of READ. If the user is a member of Group A (and Group B) and list-of-groups checking is active, the user can access the resource, even though the user is logged on to Group B. (This example assumes that other RACF checks, such as security classification checking, are met.)

Similarly, if list-of-groups checking is active, RACF recognizes the user's group-related attributes (such as group-SPECIAL) in other connect groups, regardless of which group the user is logged on to. However, the user still has each group-related attribute only within the scope of that group in which the user is assigned the attribute. (For more information on the scope of a group, see [Chapter 4, "Defining groups,"](#) on page 85.)



For example, in Figure 6 on page 66, say USER1 is also connected to GROUP3, but without group-SPECIAL for GROUP3. If list-of-groups checking is not active and USER1 logs on to GROUP3, RACF does not recognize that USER1 has group-SPECIAL authority to GROUP1 resources.

If list-of-groups checking is active and USER1 logs on to GROUP3, USER1 has group-SPECIAL authority to GROUP1 resources. However, USER1 does not have group-SPECIAL authority to GROUP3 resources. Likewise, if list-of-groups checking is active and USER1 logs on to GROUP1, USER1 has group-SPECIAL authority to GROUP1 resources, but not GROUP3 resources.

If you have the SPECIAL attribute, you can specify list-of-groups checking by using the GRPLIST option of the SETROPTS command as shown in the following example:

```
SETROPTS GRPLIST
```

To use current connect group checking, specify the NOGRPLIST option on the SETROPTS command.

**Guideline:** Use the GRPLIST option because it eases administration and minimizes the number of times the user might have to log off and log back on to access resources.

### GRPLIST considerations for z/OS UNIX

z/OS UNIX groups are RACF groups that have an z/OS UNIX group identifier (GID) defined in the OMVS segment of the group's profile. Authority checks for access to z/OS UNIX files and directories use the GID in the user's current connect group and up to 300 supplementary groups (if SETROPTS GRPLIST is active) to make group access decisions. The limit of 300 supplementary groups is the same limit defined by the NGROUPS\_MAX variable of the POSIX standard. Authority checks for other system resources use the RACF current group and list-of-groups support.

For more information, see [“Defining group identifiers \(GIDs\)” on page 513](#).

## Setting the RVARY passwords (RVARYPW option)

If you have the SPECIAL attribute, you can specify passwords that the operator must use to respond to RVARY command requests to:

- Switch the RACF databases
- Change RACF status (ACTIVATE or DEACTIVATE)
- Change mode (DATASHARE or NODATASHARE)

**Note:** OPERCMDS resources can be used to authorize the use of RVARY without requiring a console prompt in most cases. However, RVARY passwords should be established for exceptions when in failsoft mode.

READ access to the following OPERCMDS resources enables the use of the RVARY command:

- IRR.RVARY.STATUS when the ACTIVE or INACTIVE keyword is used
- IRR.RVARY.SWITCH when the SWITCH, DATASHARE, or NODATASHARE keyword is used.

For details about the OPERCMDS resources, see [“Controlling the use of operator commands” on page 240](#)

You can specify the passwords using the RVARYPW operand on the SETROPTS command. RACF allows you to specify separate passwords for switching the databases and for changing RACF status. The following example specifies HAPPY as the switch password and RABBIT as the status password:

```
SETROPTS RVARYPW(SWITCH(HAPPY) STATUS(RABBIT))
```

Password names must conform to the following rules:

- Passwords can be up to 8 characters in length.
- Valid characters for passwords are alphabetic uppercase A - Z, numeric (0 - 9), and national (#, @, and \$).

When RACF is first initialized, the switch password and the status password are both set to YES.



## Restricting the creation of general resource profiles (GENERICOWNER and ENHANCEDGENERICOWNER options)

If you have the SPECIAL attribute, you can restrict the creation of profiles in general resource classes. To do this:

1. Issue a SETROPTS GENERICOWNER command.
2. Define a \*\* profile for the class, with yourself as owner. (This prevents users lacking special authority from being able to define profiles in the class.)
3. Define a *top* profile for each user, covering the subset of resources in the class which the user is allowed to create. Each user should be the owner of this *top* profile.

You have created an environment where the user can create only profiles that are more *specific* than the user's *top* profile. The only other users who can create profiles in the user's subset of the class are:

- A user with SPECIAL authority
- A user who has group-SPECIAL authority over a user who owns the *top* profile

For example, assume that neither JOE nor RONN have the SPECIAL or group-SPECIAL attribute.

If the GENERICOWNER option is in effect, and user RONN is the owner of a JESSPOOL profile called NODEA . RONN . \*\*, JOE cannot create profile NODEA . RONN . DATA . \*\*, even though JOE has the CLAUTH(JESSPOOL) attribute.

You can alternatively choose to make a group the owner of the *top* profile for a given subset in the class. In this case, only a user with group-SPECIAL authority for the group, or with SPECIAL authority, can create profiles in the subset.

The *top* profile must end in a single asterisk (\*), double asterisks (\*\*), or one or more percent signs (%). More specific profiles are profiles that match the less specific *top* profile name character for character, up to the ending asterisks or percent signs in the less specific name.

In a search for the less specific profile, a match is found if *all* of the following are true:

- The profile name ends in a single asterisk (\*), double asterisks (\*\*), or one or more percent signs (%).
- All characters preceding the asterisks or percent signs (\* or \*\* or %) match the corresponding characters in the resource name exactly.
- The characters matching the percent signs (%) in the less-specific profile are not an asterisk (\*) or period (.) in the resource name. The length of the profile must be the same for this case.

For example, to allow USERX to RDEFINE A . B in the JESSPOOL class, you need profile A . \* in the JESSPOOL class, which is owned by USERX.

**Note:** The GENERICOWNER operand does not affect the DATASET class. It cannot be activated for individual classes. When active, GENERICOWNER affects all general resource classes except the PROGRAM class and general resource grouping classes.

For example, when working with general resource grouping classes, assume that profile A\* exists in the TERMINAL class and is owned by a group that the user does not have group-SPECIAL authority to. If the GENERICOWNER option is in effect, it will prevent the user from defining a more specific profile in the member class (for example, by using the command RDEF TERMINAL AA\*). However, having the GENERICOWNER option in effect will *not* prevent the user from defining a profile if specified on the ADDMEM operand for the grouping class profile (such as with the command RDEF GTERMINL *profile-name* ADDMEM(AA\*)).

Because it is not possible to allow the user the ability to define profiles in the TERMINAL class and disallow the user the ability to define profiles in the GTERMINL class, it is not possible to use the GENERICOWNER option to restrict the users ability to define profiles that cover resource in the any member class, including TERMINAL.

The ENHANCEDGENERICOWNER option insures that members added to grouping class profile member lists follow the same rules as profiles defined to member classes.

For example, when working with general resource grouping classes, it assumes that profile A\* exists in the TERMINAL class and is owned by a group where the user does not have group-SPECIAL authority to. If the ENHANCEDGENERICOWNER option is in effect, it will prevent user the user from defining a more specific profile in the member class (for example, by using the command RDEF TERMINAL AA\*). It will also prevent the user from defining a profile via the memberlist of a grouping profile (such as RDEF GTERMINL profile-name ADDMEM(AA\*)).

**Note:** The *top* profile which allows a given user to define additional profiles in the class must be created in the MEMBER class in order to be effective. If the *top* profile is defined as a member of a grouping class profile, its ownership has no effect on ENHANCEDGENERICOWNER processing.

Example of how a user can define profiles in the TERMINAL class starting with 'A':

```
RDEFINE TERMINAL A* OWNER(JERRY) UACC(NONE)
```

Example of how a user has no ability to define profiles in the TERMINAL class starting with 'A':

```
RDEFINE TERMINAL XYZ ADDMEM(A.*) OWNER(JERRY) UACC(NONE)
```

The ENHANCEDGENERICOWNER option works with all classes except DATASET, RVARSMBR/RACFVARS, SECLMBR/SECLABEL, PMBR/PROGRAM, GMBR/GLOBAL, SCDMBR/SECDATA, VMBR/VMEVENT, VXMBR/VMXEVENT and NODMBR/NODES.

**Note:** Both the GENERICOWNER and ENHANCEDGENERICOWNER options only affect the ability to create new profiles, or add new members to a grouping profile. These options have no effect on permission to resources, or on the ability to alter the definitions of resource profiles.

To cancel this option, specify NOGENERICOWNER on the SETROPTS command.



**Attention:** Issuing SETROPTS GENERICOWNER or ENHANCEDGENERICOWNER can prevent users with the CLAUTH attribute in general resource classes from creating profiles as they are accustomed to. Therefore, make these users OWNER of appropriate *top* generic profiles in the class. For an example, see [“Delegating authority to profiles in the FACILITY class” on page 211.](#)

## Activating general resource classes (CLASSACT option)

If you have the SPECIAL attribute, you can specify that RACF provides access authorization checking for general resource classes. You can specify this option for selected general resource classes with the CLASSACT operand of the SETROPTS command.

The following example shows how to specify RACF access authorization checking for the TERMINAL and CONSOLE resource classes.

```
SETROPTS CLASSACT(TERMINAL CONSOLE)
```

RACF prevents you from activating all classes using the SETROPTS CLASSACT(\*) operand. **Guideline:** Do not activate all RACF classes. Activate only the classes that are important to your installation. This is because some classes have a default return code of 8. For those classes, activate them only after you define the resource profiles to allow needed access.

For information on activating protection for specific general resource classes, check the index of this document for the class name.

If you have the SPECIAL attribute, you can also specify the NOCLASSACT operand on the SETROPTS command. This operand indicates that RACF performs no access authorization checking for selected general resource classes. If you specify NOCLASSACT(\*), RACF does not perform access authorization checking for any of the classes in the class descriptor table (CDT). However, you can still define resource profiles to RACF through the RDEFINE command.

## Activating generic profile checking and generic command processing

If you have the SPECIAL attribute, you can activate or deactivate generic profile checking for a class. You can specify this option with the GENERIC and NOGENERIC operands of the SETROPTS command. The following example shows how to activate generic profile checking for the DATASET class.

```
SETROPTS GENERIC(DATASET)
```

### Guidelines:

- When possible, use generic profiles to protect multiple resources and reduce administrative effort. Consider issuing SETROPTS GENERIC(*classname*) for the classes you use, so that generic profiles are usable in those classes.
- If you already have general resource profiles defined in your database, avoid issuing the SETROPTS GENERIC(\*) command. This command activates generic profile checking for all classes except resource grouping classes and classes defined with the GENERIC(DISALLOWED) attribute. Some classes, such as DIGTCERT and DIGTRING, do not support generic profile checking. These and other classes might already have profile names that contain generic characters (\*, &, and %).
- If a general resource class already has discrete profiles with names that contain generic characters (\*, &, and %), enabling generic profile checking for the class *prevents* RACF from using those discrete profiles for authorization checking.

If you enable SETROPTS GENERIC for a class that has a discrete profile name containing generic characters, the profile will be marked UNUSABLE in RLIST and SEARCH output listings.

**Tip:** Use the RDELETE command with the NOGENERIC option to delete this profile.

- In general, once you activate generic profile checking for a class and define generic profiles, avoid deactivating it with the NOGENERIC operand. RACF will not use your previously defined generic profiles for authorization checking while NOGENERIC is in effect.

If you want to perform maintenance on the generic profiles in the RACF database, you might want to temporarily deactivate generic profile checking but allow RACF command processors to update generic profiles. You can specify this environment with the NOGENERIC and GENCMD operands of the SETROPTS command. The following example shows how to specify this environment for the DATASET class.

```
SETROPTS NOGENERIC(DATASET) GENCMD(DATASET)
```

NOGENERIC and NOGENCMD are in effect when a RACF database is first initialized using IRRMIN00.

If there is a global access checking table entry of \$RACUID.\*\*/ALTER for data sets, users can create unprotected data sets even if PROTECTALL is in effect. However, *other users* are not allowed to access those data sets.

## Activating statistics collection (STATISTICS option)

Using the SETROPTS STATISTICS option does the following:

- RACF maintains two sets of statistics in a discrete resource profile. One set counts all activity for the resource or profile. The other set counts activity for each entry in the access list. It can be difficult to compare the two sets of statistics meaningfully, unless you understand how RACF maintains the statistics. For more information, see the [“STATISTICS example” on page 115](#).
- If a specific resource has unique security concerns, protect it with a discrete profile.

To see how that resource is being accessed and how many times it is being accessed, you can initiate STATISTICS. Remember that the initiation of STATISTICS is *system-wide* for all discrete profiles within a particular resource class across your system. Depending on the number of discrete profiles in the various resource classes, turning on STATISTICS might negatively affect performance.

### STATISTICS example

To help you understand how RACF maintains statistics, consider the following:

- USER1.DATA is a data set profile.
- USER1.DATA has a universal access (UACC) of READ.
- USER2 is in the access list with READ authority.
- USER3 is in the access list with UPDATE authority.
- GROUP1 is in the access list with READ authority.
- GROUP2 is in the access list with UPDATE authority.
- USER4 belongs to both groups, GROUP1 and GROUP2.
- There is no entry for &RACUID.\* in the global access checking table.

If USER1 reads USER1.DATA, the overall READ count in the profile increases by one. No counts in the access list are changed, because access lists are not used when users process their own data.

If USER2 reads the data set, two counts are updated: the overall READ count and the count in USER2's access list entry.

If USER3 reads the data set, two counts are updated: the overall READ count and the count in USER3's access list entry (even though the entry says UPDATE). The counts in the access list merely record that access was granted by that entry. The access granted can be as specified by the entry, or a lower level, as in this example.

If list-of-groups processing is active (through SETROPTS GRPLIST) and USER4 reads the data set, RACF examines the access list to see if any of USER4's groups are in the list. If any of the groups is found, the entry with the highest authority is used. In this case, the access list entry for GROUP2 (UPDATE) increases, along with the overall READ count for the profile.

If any other user or group reads the data set, it gains access because of the universal access of READ, and the overall READ count increases. If any user with OPERATIONS authority updates the data set, the overall UPDATE count increases.

## Using options in RACROUTE REQUEST=VERIFY statistics collection

A user with the SPECIAL attribute can request RACF to record statistics during RACROUTE REQUEST=VERIFY processing. The REQUEST=VERIFY is issued when a user logs on to the system, a batch job enters the system, or when RACF does such work as a directed command, application update, or password change on behalf of a user. RACF maintains the following statistics:

- The date and time the REQUEST=VERIFY request is issued for a particular user

**Note:** An auditor cannot rely upon LAST-ACCESS date for the last (if ever) authentication of a user. If an authentication request (RACROUTE REQUEST=VERIFY) includes STAT=NO, there is no indication when an id was last (if ever) authenticated. This is the case with Db2z/OS.

- The number of REQUEST=VERIFY requests for a user to a particular group
- The date and time of the last REQUEST=VERIFY request for a user to a particular group

You must maintain statistics if you intend to use the INACTIVE, REVOKE, HISTORY, and WARNING options of SETROPTS.

If you do not intend to use a SETROPTS options that requires statistics and you do not need all of the statistics, you can issue SETROPTS NOINITSTATS to reduce the RACF database I/O associated with REQUEST=VERIFY requests. You must have the SPECIAL attribute to issue the SETROPTS NOINITSTATS command.

If NOINITSTATS is in effect, the INACTIVE, REVOKE, HISTORY, and WARNING options cannot be used.

If you have the SPECIAL attribute, you can also specify the INITSTATS operand on the SETROPTS command to indicate that you want RACF to record REQUEST=VERIFY statistics, as shown in the following example:

```
SETROPTS INITSTATS
```

INITSTATS is in effect when RACF is first initialized.

**Note:** Being enabled for sysplex communication might reduce the overhead associated with INITSTATS.

## Reducing application logon statistics

You can reduce the system impact of recording logon statistics for selected applications by recording statistics for only the first daily logon by each user, rather than for every logon by each user.

For applications that specify the APPL operand on the RACROUTE REQUEST=VERIFY request, you can specify daily logon statistics for selected applications by customizing the APPLDATA value of profiles in the APPL class. (Your installation might already use APPL class profiles to control user access to applications. See [“Protecting applications” on page 223.](#))

### Steps for specifying daily logon statistics

To specify recording of daily logon statistics for a selected application, perform the following steps:

1. Determine the name of the application. See your programmer or refer to the documentation for the application to determine the application name specified on its RACROUTE REQUEST=VERIFY requests.

2. Create or modify a profile in the APPL class for the application name and specify daily logon statistics, as follows.

- If not already defined, create a new APPL profile for this application name and specify daily logon statistics.

#### Example:

```
RDEFINE APPL applname UACC(NONE) APPLDATA('RACF-INITSTATS(DAILY)')
```

**Tip:** If similarly named applications have the same requirements, create a generic APPL profile.

#### Example:

```
RDEFINE APPL CICSREG* UACC(NONE) APPLDATA('RACF-INITSTATS(DAILY)')
```

- If an APPL class profile for this application is already defined, modify the profile to specify daily logon statistics:

```
RALTER APPL applname APPLDATA('RACF-INITSTATS(DAILY)')
```

- If the APPL class profile for this application already contains a value in the APPLDATA field, modify the existing APPLDATA value to include the RACF-INITSTATS(DAILY) text string.

#### Examples:

```
RALTER APPL applname
  APPLDATA('existing application data RACF-INITSTATS(DAILY)')
RALTER APPL applname
  APPLDATA('RACF-INITSTATS(DAILY) existing application data')
```

3. If you created a new APPL profile for the application in Step [“2” on page 117](#) and you specified UACC(NONE), authorize appropriate users and groups to access the application.

#### Example:

```
PERMIT applname CLASS(APPL) ID(userid-or-group) ACCESS(READ)
```

4. Activate your changes, as follows.

- If the APPL class is not already active, activate the APPL class.

**Guideline:** Activate RACLIST processing for the APPL class for improved performance, especially for high usage applications that issue RACROUTE REQUEST=VERIFY requests for every user.

**Example:**

```
SETROPTS CLASSACT(APPL) RACLIST(APPL)
```

- If the APPL class is already active and RACLISTed, refresh the APPL class.

**Example:**

```
SETROPTS RACLIST(APPL) REFRESH
```

---

You have now specified recording of daily logon statistics for a selected application.

### *Considerations for using daily logon statistics*

When RACF - INITSTATS(DAILY) is in effect for an application that uses the RACROUTE REQUEST=EXTRACT request to extract the LJDATE and LJTIME fields from user profiles, the LJDATE and LJTIME fields represent the last *recorded* date and time that the user entered the system using the RACROUTE REQUEST=VERIFY request. The last recorded date and time might be different from the last date and time that the user entered the system.

Similarly, when RACF - INITSTATS(DAILY) is in effect, the USBD\_LASTJOB\_DATE and USBD\_LASTJOB\_TIME fields in record type 200 (user basic data) from the output of the RACF database unload (IRRDUBU00) utility represent the last *recorded* date and time the user entered the system.

## Bypassing resource statistics collection

If you have the SPECIAL attribute, you can request that RACF bypass the recording of statistical information in discrete profiles for the DATASET class for classes defined in the class descriptor table (CDT). You specify this option with the NOSTATISTICS operand of the SETROPTS command.

The statistics you can bypass include:

- The date that the resource was last referenced
- The date that the resource was last updated (not recorded for terminals)
- The number of times that the resource was accessed for each of the following access authorities: ALTER, CONTROL, UPDATE, and READ (only READ count is recorded for terminals)
- The number of times that each user or group in the access list has accessed the resource

If you have the SPECIAL attribute, you can also specify the STATISTICS operand on the SETROPTS command and identify the classes for which you want RACF to record statistical information.

If you specify an asterisk (\*), you activate the recording of statistical information for all resource classes.

When a RACF database is first initialized using IRRMIN00, STATISTICS is in effect for the DATASET, DASDVOL, TAPEVOL, and TERMINAL classes. **Guideline:** Because statistics recording affects system performance, deactivate this option until your installation evaluates the need to use it against its potential performance impact. See [z/OS Security Server RACF System Programmer's Guide](#) for more information.

## Activating global access checking (GLOBAL option)

If you have the SPECIAL attribute, you can activate or deactivate global access checking on a class-by-class basis or for all classes. You can specify this option with the GLOBAL and NOGLOBAL operands of the SETROPTS command. The following example shows how to activate global access checking for the FACILITY class.

```
SETROPTS GLOBAL(FACILITY)
```

If you specify GLOBAL (\*), you activate global access checking for all valid classes. Valid classes you can specify are:

- The DATASET class
- The NODE grouping class
- The SECLABEL grouping class
- All other classes defined in the class descriptor table, except for the remaining grouping classes

When you use the SETROPTS command to activate (or reactivate) global access checking for a class, RACF builds (or updates) the in-storage global access checking tables. However, you can use the RDEFINE and RALTER commands to maintain profiles on the database, regardless of whether the global access checking option is active for a class.

NOGLOBAL is in effect when RACF is first initialized.

**Note:** The SETROPTS GLOBAL (*classname*) command is propagated when the system is enabled for sysplex communication.

## RACF-protecting all data sets (PROTECTALL option)

If you have the SPECIAL attribute, you can activate PROTECTALL processing by using the PROTECTALL operand of the SETROPTS command. If PROTECTALL is active, a user can create or access a data set only if the data set is RACF-protected by either a discrete or generic profile, *or* the access is allowed by global access checking. Note that if PROTECTALL is in effect, generic profile checking should also be in effect for the DATASET class. Otherwise, users can create only data sets that are protected by discrete profiles. The following examples show how to specify these options:

```
SETROPTS PROTECTALL
SETROPTS GENERIC(DATASET)
```

### Note:

1. PROTECTALL requires that you RACF-protect all data sets. This protection includes tape data sets if your installation specifies TAPEDSN on the SETROPTS command.
2. After defining, altering, or deleting a generic profile, the following command ensures that the profile is in effect during authorization checking:

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

3. Started procedures with the privileged or trusted attribute and users with the SPECIAL attribute can access a data set that has no RACF profile, even if PROTECTALL is in effect. These exceptions allow recovery if a critical profile is accidentally deleted.
4. If there is a global access checking table entry of &RACUID.\*/ALTER for data sets, users can create unprotected data sets even if PROTECTALL is in effect. However, *other users* cannot access those data sets.

PROTECTALL also has a warning option that allows the request even though the data set is not protected, but sends a warning message to the user and the MVS console. For example:

```
SETROPTS PROTECTALL(WARNING)
```

**Guideline:** Before using PROTECTALL(WARNING), perform the following actions to reduce the number of messages generated:

- Ensure that a RACF user or group profile is defined for all catalog aliases.
- Ensure that all RACF users and groups have a generic data set profile of the form:

```
'high-level-qualifier.*'
```

or, if SETROPTS EGN is in effect:

```
'high-level-qualifier.**'
```

### Note:

PROTECTALL applies to all data sets that do not have system-generated temporary names and that do not have names that begin with \*\*SYSUT. You can extend PROTECTALL to include temporary data sets with system-generated names by using the naming conventions table to modify the name that RACF uses to look like a permanent name. If your installation uses nonstandard names for temporary data sets, you must also predefine entries in the global access checking table that allow these data sets to be created and accessed.

If you have the SPECIAL attribute, you can also deactivate PROTECTALL processing by using the NOPROTECTALL operand.

NOPROTECTALL is in effect when RACF is first initialized.

## Activating JES2 RACF support

Several RACF options support JES processing. See the following topics for more information.

RACF option	See ...
SETROPTS JES(BATCHALLRACF)	<a href="#">“Forcing batch users to identify themselves to RACF” on page 454</a>
SETROPTS JES(XBMALLRACF)	<a href="#">“Support for execution batch monitor (XBM)” on page 454</a>
SETROPTS JES(EARLYVERIFY)	<a href="#">“JES user ID early verification” on page 455</a>
SETROPTS JES(NJEUSER)	<a href="#">“Understanding default user IDs” on page 480</a>
SETROPTS JES(UNDEFINEDUSER)	<a href="#">“Understanding default user IDs” on page 480</a>

## Preventing access to uncataloged data sets (CATDSNS option)

If you have the SPECIAL attribute, you can use the CATDSNS operand of the SETROPTS command to prevent users who do not have the SPECIAL attribute from gaining access to data sets that DFSMS controls. These data sets include system temporary data sets and data sets that are not cataloged. When CATDSNS is in effect, such users cannot read or write to temporary or uncataloged DASD data sets, and (unless the following restriction applies) they cannot read uncataloged tape data sets.

**Restriction:** If you use DFSMSrmm to manage your tape data sets and the TAPEAUTHF1 option is active (in the DEVSUPxx member of SYS1.PARMLIB), an uncataloged tape data set might be accessible to a user who has access to the first file on the tape volume when the first file is cataloged. (See [z/OS DFSMSrmm Implementation and Customization Guide](#).) If you use a different tape management system, refer to your product documentation.

When the CATDSNS option is in effect, uncataloged data sets that are protected, for example by a generic profile, cannot be accessed by users who are unauthorized by the generic profile, even if they are authorized to access uncataloged data sets because they have READ access to ICHUNCAT.data-set-name. Access to the ICHUNCAT.data-set-name resource does not provide access to uncataloged data sets where other RACF authorization checking denies it. The CATDSNS option provides an additional point of failure in the RACF authorization sequence, not a point of success.

For a detailed view of the sequence of RACF checking, see [“Pictorial view of RACF authorization checking” on page 723](#). The CATDSNS processing is covered in Step [“30” on page 723](#).

There are some exceptions. For example, CATDSNS processing does not fail the request:

- If the user has READ access to a resource called ICHUNCAT.data-set-name in the FACILITY class
- If the data set is protected by a discrete profile



- If the data set is created and used within the same job or TSO session. If this data set is not cataloged before job termination or TSO logoff, it is not accessible to any other job or TSO session.

To activate the CATDSNS option, enter:

```
SETROPTS CATDSNS
```

In addition, if SETROPTS MLACTIVE is in effect, RACF produces one or more type 83 SMF records whenever a data set profile is added, changed, or deleted. This record contains the list of the cataloged data sets that are affected by the change, addition, or deletion of the profile.

If the SETROPTS CATDSNS option is not in effect, the list of the affected data sets might not be complete. Therefore, using the SETROPTS CATDSNS option allows you to list and audit the data sets affected by the change in a particular profile.

Because the SETROPTS CATDSNS option prevents users without the SPECIAL attribute from accessing uncataloged data sets (except as noted above), the list of data set names provided by the DSNS operand on the LISTDSD command is identical to the list of all data sets affected by the profile change.

You can also specify CATDSNS(WARNING), which allows accesses, but sends a warning message to the user and the security administrator.

To cancel the CATDSNS option, specify NOCATDSNS on the SETROPTS command.

## Activating enhanced generic naming for the DATASET class (EGN option)

If you have the SPECIAL attribute, you can activate enhanced generic naming for the DATASET class by issuing the SETROPTS command with the EGN operand:

```
SETROPTS EGN
```

When you activate this option, RACF allows you to specify the generic character \*\* (in addition to the generic characters \* and %) when you define any of the following:

- A generic profile in the DATASET class
- An entry in the global access checking table for the DATASET class

Note that enhanced generic naming changes the meaning of the generic character \* for generic data set profiles. (SETROPTS EGN has no effect on general resource profiles.) In addition, a generic profile such as *hlq.\** defined while SETROPTS NOEGN is in effect, will appear as *hlq.\*.\** when listed after EGN is activated.

For information on specifying profile names with enhanced generic naming, see [z/OS Security Server RACF Command Language Reference](#).

To deactivate enhanced generic naming for the DATASET class, issue the SETROPTS command with the NOEGN operand.

**Guideline:** Do *not* deactivate enhanced generic naming after data set profiles have been created while enhanced generic naming was active.

NOEGN is in effect when a RACF database is first initialized using IRRMIN00.

## Controlling data set modeling (MODEL option)

The MODEL operand of the SETROPTS command allows you to automatically supplement the information that is normally placed in new data set profiles by ADSP, PROTECT=YES, or ADDSD. Modeling can be effective for user, group, and GDG data sets on an individual user ID or group name basis. You control this processing with the MODEL(USER), MODEL(GROUP), and MODEL(GDG) operands of the SETROPTS command. The following example shows how to specify this option for USER data sets:

```
SETROPTS MODEL(USER)
```

To specify this option, you must have the SPECIAL attribute.

**Note:** The FROM(*profile-name*) operand on the ADDSD command overrides any specifications from the MODEL(USER) or MODEL(GROUP) operands.

If you specify MODEL(NOGDG), MODEL(NOUSER), or MODEL(NOGROUP), RACF does not use a model data set profile for new GDG, USER, or GROUP data sets, respectively. For more information, see [“Automatic profile modeling for data sets”](#) on page 158.

If you specify NOMODEL, RACF does not use automatic model processing for GDG, group, or user data sets.

NOMODEL is in effect when a RACF database is first initialized using IRRMIN00.

## Bypassing automatic data set protection (NOADSP option)

If you have the SPECIAL attribute, you can specify that RACF ignore the ADSP attribute (if specified in user profiles). To do this, enter:

```
SETROPTS NOADSP
```

With the SETROPTS NOADSP operand in effect, RACF does not automatically create discrete data set profiles when users who have the ADSP attribute create new data sets. IBM recommends the NOADSP option because it reduces the number of data set profiles in the RACF database. Using generic data set profiles is generally more efficient.

You can reinstate normal ADSP processing with the ADSP operand.

ADSP is in effect when a RACF database is first initialized using IRRMIN00.

## Displaying and logging real data set names (REALDSN option)

If your installation is using the naming conventions table or installation exits to convert data set names, you can specify that RACF put the actual data set names used by RACROUTE REQUEST=AUTH and REQUEST=DEFINE into any SMF log record and operator messages. To do this, enter:

```
SETROPTS REALDSN
```

Putting the REALDSN option into effect ensures that log printouts and operator messages identify data sets by their real names rather than by the data set names that are created by installation exit routines to conform to RACF naming conventions. To specify this option, you must have the SPECIAL attribute.

**Note:** This option has no effect on single-qualifier data set names (unless they have been modified by the naming conventions table or an exit routine), whose real data set names continue to be the prefixed ones. For more information, see [z/OS Security Server RACF System Programmer's Guide](#).

NOREALDSN is in effect when a RACF database is first initialized using IRRMIN00.

## Protecting data sets with single-qualifier names (PREFIX option)

You can RACF-protect data sets that have names consisting of only a single qualifier (that is, single-level names). To get RACF protection for single-qualifier names, issue the SETROPTS command with the PREFIX operand to activate the facility and define a prefix. If you use the SETROPTS command with the PREFIX operand to define a prefix (high-level qualifier), RACF internally modifies single-qualifier names by adding the high-level qualifier when it processes requests for the data set. The prefix must be an existing group name and cannot be the name used as the high-level qualifier of any actual data sets or data set profiles in the system. The following example shows how to RACF-protect data sets with single-qualifier names with the prefix RAC1LVL:

```
SETROPTS PREFIX(RAC1LVL)
```



**Attention:** If you do not issue the SETROPTS command with the PREFIX operand, a system ABEND occurs if a discrete profile is created when a user tries to create a data set with a single-qualifier name.

To specify the PREFIX or NOPREFIX operands, you must have the SPECIAL attribute.

## Activating tape data set protection (TAPEDSN option)

If you have the SPECIAL attribute, you can activate tape data set protection by using the TAPEDSN operand of the SETROPTS command. When you activate tape data set protection, RACF refers to profiles in the DATASET class when verifying a user's access authority to a tape data set. The following example shows how to specify this option:

```
SETROPTS TAPEDSN
```

If you have the SPECIAL attribute, you can also deactivate tape data set protection by using the NOTAPEDSN operand on the SETROPTS command.

NOTAPEDSN is in effect when a RACF database is first initialized using IRRMIN00.

**Guideline:** If you use a tape management system, such as DFSMSrmm, do not enable TAPEDSN. For more information, see [“Using DFSMSrmm with RACF” on page 168](#).

## Activating tape volume protection (TAPEVOL option)

If you have the SPECIAL attribute, you can activate tape volume protection by using the CLASSACT(TAPEVOL) operand of the SETROPTS command. When you activate tape volume protection, RACF refers to profiles in the TAPEVOL class when verifying a user's access authority to a tape volume. If both the TAPEVOL class and TAPEDSN are active, RACF maintains profiles in both the TAPEVOL and DATASET classes. Data fields within these two profiles (data set name in the TAPEVOL profile and volume serial in a discrete data set profile) link the two profiles to each other. The following example shows how to activate tape volume protection:

```
SETROPTS CLASSACT(TAPEVOL)
```

If you have the SPECIAL attribute, you can also deactivate tape volume protection by using the NOCLASSACT(TAPEVOL) operand on the SETROPTS command.

**Guideline:** If you use a tape management system, such as DFSMSrmm, do not enable TAPEVOL. For more information, see [“Using DFSMSrmm with RACF” on page 168](#).

## Establishing a security retention period for tape data sets (RETPD option)

The RACF security retention period is the number of days that RACF protection remains in effect for a tape data set. For example, to select tape volumes to return to the scratch pool, a tape librarian can issue the SEARCH command with the EXPIRES operand. When the librarian issues this command, RACF uses the security retention period to check if RACF protection for all data sets on a tape volume has expired. If RACF protection has expired, the tape volume can be returned to the scratch pool.

If you use a tape management system, such as DFSMSrmm, you need not enable RETPD. For more information, see [“Using DFSMSrmm with RACF” on page 168](#).

If you define a tape volume with a TVTOC, RACF uses the security retention period when checking the authority to overwrite a data set on the volume with a data set of a different name. Before opening the tape data set for output, RACF ensures that the security retention periods for all of the following data sets on the volume have expired.

Users can specify a security retention period on the ADDSD and ALTDSD commands, or, for data sets covered by a discrete profile, by the use of the EXPDT/RETPD JCL operands. If a user does not specify a retention period with RACF commands or JCL, RACF selects a retention period through profile modeling, an installation exit, or a system default set with the RETPD operand on the SETROPTS command.

If you have the SPECIAL attribute, you can establish a system default number of days with the RETPD operand. With this operand, you can specify a one to five digit number in the range of 0 - 65533. To set

a default retention period for a data set that never expires, specify 99999. The following example shows how to specify a RACF security retention period of 365 days:

```
SETROPTS RETPD(365)
```

RACF uses the default security retention period for a tape data set in the following situations:

- When a user defines a data set (using ADDSD) without specifying a retention period
- When a user defines a data set (using ADDSD) or changes a data set profile (using ALTDSD) and specifies RETPD(0)
- When a user specifies RETPD=0 on the JCL statement
- When a user specifies EXPDT=*today's date* on the JCL statement
- When a user omits the RETPD and EXPDT parameters on the JCL statement

For example, if a user specifies RETPD=0 on the JCL statement and your installation has established a default retention period of 365 using SETROPTS RETPD, RACF uses 365 as the retention period for the user's data set.

The default security retention period when RACF is installed is RETPD(0), to indicate no retention period.

**Note:**

1. The RACF security retention period is independent of the data set retention period specified by the EXPDT/RETPD JCL operands. However, the two retention periods are the same initially if the data set has a discrete profile. You can modify the security retention period by using the ALTDSD command, but you cannot change the data set retention period in the tape label of tape data sets.
2. The security retention period tape data sets has meaning only when both the TAPEVOL class and TAPEDSN are active.

## Erasing scratched or released data (ERASE option)

If you have the SPECIAL attribute, you can activate erase-on-scratch processing with the ERASE operand on the SETROPTS command. If erase-on-scratch is active and you specify the ERASE option in the data set profile using the ADDSD or ALTDSD command (this sets the erase indicator), ERASE specifies that data management is to erase the contents of any deleted data sets and any scratched or released DASD extents that are part of a data set protected by that profile. When RACF runs on a system that includes data management support for erase-on-scratch, the contents of a scratched and erased data set cannot be read, unless the following restriction related to tape data sets applies.

**Restriction:** Setting the erase indicator in a data set profile might *not* cause data on tape to be erased unless your installation activates the TAPEAUTHDSN option in the DEVSUPxx member of SYS1.PARMLIB and your tape management supports the TAPEAUTHDSN setting.

When you activate the TAPEAUTHDSN option and a data set (with the erase indicator set) is opened on tape, data management passes the erase indicator setting to your tape management system, so that your tape management system can erase the protected data on tape before the tape is scratched. If your tape management system is DFSMSrmm, when you activate TAPEAUTHDSN and you set the erase indicator for a data set profile, all contents on tape are erased during volume release processing. For information about using this option with DFSMSrmm, see [z/OS DFSMSrmm Implementation and Customization Guide](#). If you use a different tape management system, refer to your product documentation.

The ERASE operand has several suboperands that allow an installation to override user specifications.

- ALL specifies that *all* data sets (including temporary data sets) are always erased, regardless of the erase indicator in the data set profile. When this option is selected, installation exit routines *cannot* prevent any data set from being erased by overriding this option.
- SECLEVEL allows you to specify a security level at which all data sets at this security level or higher are always erased, regardless of the erase indicator in the profile.
- NOSECLEVEL specifies that RACF is not to use the security level in the data set profile when it decides whether data management is to erase a scratched data set.

The following example shows how to activate erase-on-scratch processing for all data sets with a security level of CONFIDENTIAL or higher.

```
SETROPTS (ERASE) SECLEVEL(CONFIDENTIAL)
```

If you specify the ERASE operand without the ALL suboperand, erase-on-scratch processing applies only to data sets that do not have system-generated temporary names and do not have names that begin with \*\*SYSUT. You can extend erase-on-scratch to include temporary data sets with system-generated names by using the naming conventions table to modify system-generated names to look like permanent names. In this case, you need not specify ALL.

If you have the SPECIAL attribute, you can also deactivate erase-on-scratch processing by using the NOERASE operand on the SETROPTS command.

NOERASE is in effect when a RACF database is first initialized using IRRMIN00.

## Establishing national language defaults (LANGUAGE option)

If you have the SPECIAL attribute, you can specify the installation defaults for national languages on your system. You can specify a primary language and a secondary language. Applications that use the RACROUTE REQUEST=EXTRACT request to determine a user's primary and secondary languages can use the installation defaults set by SETROPTS if the user does not have his or her own language preferences.

To specify the installation default languages, enter:

```
SETROPTS LANGUAGE (PRIMARY(Language1) SECONDARY(Language2))
```

You must specify a 3-character language code unless RACF is running with the MVS message service active.

LANGUAGE(PRIMARY(ENU) (SECONDARY(ENU))), which means American English, is in effect when a RACF database is first initialized using IRRMIN00.

## SETROPTS options to activate in-storage profile processing

RACF provides processing to activate in-storage profiles. To help maximize the performance of your RACF database, you can activate one of the following SETROPTS operands for each eligible RACF general resource class.

### GENLIST

RACF copies all generic profiles into storage for the specified class.

### RACLIST

RACF copies all profiles, both discrete and generic, into storage for the specified class.

**Restriction:** You cannot activate SETROPTS GENLIST and SETROPTS RACLIST processing for the same general resource class.

There is no need to respecify your GENLIST or RACLIST options for each class following a system IPL. Each time you IPL, RACF automatically reactivates GENLIST and RACLIST processing for your requested classes and recopies the applicable profiles into storage.

### Guidelines for choosing between GENLIST and RACLIST processing for a class:

- Generally, RACLIST provides the best performance with the lowest usage of common storage.
- When a class contains numerous discrete profiles, activating GENLIST, rather than RACLIST, significantly reduces your common storage requirements. However, GENLIST processing might degrade performance, particularly for a z/OS system sharing the RACF database, because needed profiles might be unavailable in storage and require physical access to the RACF database.

For details about RACF virtual storage requirements, see [z/OS Security Server RACF System Programmer's Guide](#).

For information on in-storage profile processing with shared systems, see:

- “[SETROPTS GENLIST processing on shared systems](#)” on page 126
- “[SETROPTS RACLIST processing on shared systems](#)” on page 128.

## SETROPTS GENLIST processing

If you have the SPECIAL attribute, you can activate SETROPTS GENLIST processing. Activate this function for general resource classes that contain a small number of frequently referenced generic profiles. When you activate SETROPTS GENLIST processing, you enable the sharing of in-storage generic profiles for the classes you specify. For a list of the classes eligible for GENLIST processing, see the description of the class descriptor table (CDT) in [z/OS Security Server RACF Macros and Interfaces](#).

To activate this function, issue the SETROPTS GENLIST(*classname*), where *classname* is one of the following:

- A member class specified in the class descriptor table (CDT)
- Grouping class RACFVARS or NODES

RACF processes *classname* and all classes that share the same POSIT value on their class descriptor table (CDT) entries. The following example shows how to activate SETROPTS GENLIST processing for the TERMINAL class.

```
SETROPTS GENLIST(TERMINAL)
```

After you activate SETROPTS GENLIST processing for a general resource class, RACF copies a generic profile in that class from the RACF database into common storage the first time an authorized user requests access to a resource protected by it. The profile is retained in common storage and is available to all authorized users, thereby saving real storage because the need to retain multiple copies of the same profile (one copy for each requesting user) in common storage is eliminated. Also, because RACF does not have to retrieve the profile each time a user requires access to it, this function saves processing overhead.

Note that a general resource class must be active before you can activate SETROPTS GENLIST processing for that class. If the class is not active, issue the SETROPTS command with both the GENLIST and CLASSACT operands and specify the desired class. The following example shows how to activate the TERMINAL class and SETROPTS GENLIST processing for that class on the same command.

```
SETROPTS CLASSACT(TERMINAL) GENLIST(TERMINAL)
```

For more information on activating protection for specific general resource classes, check the index of this document for the class name.

## Deactivating SETROPTS GENLIST processing

If you have the SPECIAL attribute, you can deactivate SETROPTS GENLIST processing for general resource classes. To deactivate this function, issue the SETROPTS command with the NOGENLIST operand and the selected general resource classes.

NOGENLIST is the default and is in effect for all eligible classes that are defined in the class descriptor table (CDT) when a RACF database is first initialized using IRRMIN00.

See [z/OS Security Server RACF System Programmer's Guide](#) for more information about SETROPTS GENLIST processing.

## SETROPTS GENLIST processing on shared systems

If your installation has two or more systems sharing a RACF database, you need to issue the SETROPTS GENLIST command on only one of those systems. SETROPTS GENLIST processing is automatically propagated to all systems sharing the database.



## Refreshing profiles for SETROPTS GENLIST processing

If your installation has activated SETROPTS GENLIST processing for a particular resource class, you must refresh in-storage profiles for this processing when you make changes to one of these profiles in the database. Refreshing profiles for SETROPTS GENLIST processing ensures that the most current copy of a profile resides in common storage and is available for RACF authorization checking. To refresh profiles for this processing, issue the SETROPTS command with the GENERIC and REFRESH operands and specify the appropriate resource classes. For more information, see [“Refreshing in-storage generic profile lists \(GENERIC REFRESH option\)”](#) on page 130.

For information about SETROPTS REFRESH processing on shared systems, see [“Refreshing shared systems \(REFRESH option\)”](#) on page 131.

## SETROPTS RACLIST processing

If you have the SPECIAL attribute, you can activate SETROPTS RACLIST processing. When you activate SETROPTS RACLIST processing, you enable the sharing of both in-storage discrete and in-storage generic profiles for the classes you specify. For a list of the classes eligible for RACLIST processing, see the description of the class descriptor table (CDT) in *z/OS Security Server RACF Macros and Interfaces*.

To activate this function, issue SETROPTS RACLIST(*classname*), where *classname* is one of the following:

- A member class for which RACLIST=ALLOWED is specified in the class descriptor table (CDT)
- Grouping class RACFVARS or NODES

RACF will RACLIST *classname* and all classes that share the same POSIT value on their class descriptor table (CDT) entries. The following example shows how to activate SETROPTS RACLIST processing for the TERMINAL class.

```
SETROPTS RACLIST(TERMINAL)
```

### Note:

1. If the system is enabled for sysplex communication and a command is successful on the system on which it was issued, RACF propagates the command to the other members of the data sharing group.
2. If the command fails on any of the peer systems and the system is in data sharing mode, RACF stops processing the command and backs it out of all the member systems, including the system on which it was issued.
3. In non-data sharing mode, the command can fail on a peer system without backing out of the other systems.
4. If the system is not enabled for sysplex communication, the command does not take effect on the other systems sharing the database until you issue it on those systems or the systems are IPLed.

When you activate SETROPTS RACLIST processing for a general resource class, RACF loads both discrete and generic profiles for the class into a data space. These profiles are available to all authorized users, thereby eliminating the need for RACF to retrieve a profile each time a user requests access to a resource protected by it. As a result, when you activate this function, you reduce processing overhead.

If the RACGLIST class is active and has a profile with the same name as the RACLISTed class, RACF saves the results on the database as *classname\_nnnnn* profiles in the RACGLIST class, in addition to loading them into a data space. For example, RACF would save the RACLISTed data for the TERMINAL class as TERMINAL\_00001, TERMINAL\_00002, and so forth. For more information on RACGLIST, see [“The RACGLIST class”](#) on page 238.

If RACROUTE REQUEST=LIST,GLOBAL=YES was previously issued for the class, issuing SETROPTS RACLIST deletes the data space created by the RACROUTE request and replaces it with a new one. The SETROPTS RACLIST overrides the GLOBAL=YES RACLIST. Output from a SETR LIST command displays the class in the SETR RACLIST CLASSES = line rather than in the GLOBAL=YES RACLIST ONLY = line. For more information, see [“Using RACROUTE REQUEST=LIST,GLOBAL=YES support”](#) on page 237.

Note that a general resource class must be active before you can activate SETROPTS RACLIST processing for that class. If the class is not active, issue the SETROPTS command with both the RACLIST and CLASSACT operands and specify the desired class. The following example shows how to activate the TERMINAL class and SETROPTS RACLIST processing for that class on the same command.

```
SETROPTS CLASSACT(TERMINAL) RACLIST(TERMINAL)
```

For more information on activating protection for specific general resource classes, check the index of this document for the class name.

## Additional considerations for RACLIST

There are some special considerations regarding RACLIST processing. For information, see the following topics.

Topic	See...
Profile size for resources in classes processed using SETROPTS RACLIST or RACROUTE REQUEST=LIST	<a href="#">“Limiting the size of your access lists” on page 193</a>
SETROPTS RACLIST processing for resources in the CDT class (dynamic classes)	<a href="#">Chapter 10, “Administering the dynamic class descriptor table (CDT),” on page 263</a>

## Deactivating SETROPTS RACLIST processing

If you have the SPECIAL attribute, you can deactivate SETROPTS RACLIST processing for general resource classes. To deactivate this option, issue SETROPTS NORACLIST(*classname*). RACF processes *classname* and all classes that share the same POSIT value on their class descriptor table (CDT) entries.

You can also use SETROPTS NORACLIST to delete a data space created by a RACROUTE REQUEST=LIST,GLOBAL=YES command. Therefore, the command must operate on classes even though RACLIST=DISALLOWED is specified in the class descriptor table (CDT). If RACGLIST is active and RACGLIST *classname\_nnnnn* profiles exist, RACF deletes them and keeps the base RACGLIST profile name. For more information, see [“The RACGLIST class” on page 238](#) and [“Using RACROUTE REQUEST=LIST,GLOBAL=YES support” on page 237](#).

### Note:

1. You should issue a SETROPTS NORACLIST command for classes RACLISTed by RACROUTE REQUEST=LIST,GLOBAL=YES *only* after all classes that issued RACROUTE REQUEST=LIST,ENVIR=CREATE,GLOBAL=YES have given up their access to the RACLIST data space by issuing RACROUTE REQUEST=LIST,ENVIR=DELETE.
2. If the system is enabled for sysplex communication and a command is successful on the system on which it was issued, RACF propagates the command to the other members of the data sharing group.
3. If the command fails on any of the peer systems RACF does *not* back it out of the member systems.
4. If the system is not enabled for sysplex communication, the command does not take effect on the other systems sharing the database until you issue it on those systems or the systems are IPLed.

NORACLIST is the default and is in effect for all eligible classes that are defined in the class descriptor table (CDT) when a RACF database is first initialized using IRRMIN00.

See [z/OS Security Server RACF System Programmer's Guide](#) for more information about SETROPTS RACLIST processing.

## SETROPTS RACLIST processing on shared systems

If two or more systems that are not enabled for sysplex communication are sharing a RACF database, SETROPTS RACLIST processing applies only to the system on which you issue the SETROPTS command. With this type of sharing, you must issue the SETROPTS command on all of the systems in order to



perform RACLIST processing on all of the systems. If you do not issue the SETROPTS RACLIST command on one of the shared systems, RACLIST is performed for that system when the system re-IPLs.

On systems that are enabled for sysplex communication, issue the SETROPTS RACLIST command only once. If the command is successful, it is propagated to the other members of the data sharing group.

## Refreshing profiles for SETROPTS RACLIST processing

Any changes made to discrete or generic profiles activated for SETROPTS RACLIST processing become effective only when you issue the SETROPTS command with both the RACLIST and REFRESH operands. You can refresh these profiles if you have the SPECIAL attribute.

To refresh the profiles, issue SETROPTS RACLIST(*classname*) REFRESH. RACF refreshes *classname* and all classes that share the same POSIT value on their class descriptor table (CDT) entries.

### Guidelines:

- Whenever you delete a profile, issue the SETROPTS RACLIST REFRESH command immediately. If you don't, the profile no longer exists in the database, but the BASE, SESSION and ICSF segment information stored in the data space remains until the refresh is done. The mismatch between the database and the data space can cause unexpected results.
- If you update only the BASE, SESSION and ICSF segments, you can wait to issue the SETROPTS RACLIST REFRESH command until you want the changes to be active. If you update BASE, SESSION, or ICSF segment information, and other segments, issue the SETROPTS RACLIST REFRESH command immediately. If you do not update the BASE, SESSION, or ICSF segments, and you update other segments, you do not need to issue the SETROPTS RACLIST REFRESH command.

For some classes, selected profile data is kept in storage. Changes to these profiles might not be active until you refresh the in-storage profiles. Issuing a SETROPTS RACLIST REFRESH command after you make changes ensures that profile data is consistent. An example of this type of class is PTKTDATA.

You can also use SETROPTS RACLIST REFRESH to refresh a class RACLISTed by a RACROUTE REQUEST=LIST GLOBAL=YES command. RACF deletes the old data space and loads the discrete and generic profiles for the class into a new data space.

Issuing the SETROPTS RACLIST REFRESH command has no effect on which line of SETROPTS LIST output displays a RACLISTed class. If the class were RACLISTed solely by RACROUTE REQUEST=LIST, ENVIR=CREATE, GLOBAL=YES the class will be listed in the GLOBAL=YES RACLIST ONLY = line. Regardless of whether the class was RACLISTed by that means, if it was RACLISTed by SETR RACLIST *classname*, the class will be listed only in the SETR RACLIST CLASSES = line.

If the RACGLIST class is active and contains a profile named *classname*, RACF rebuilds or creates the RACGLIST *classname\_nnnnn* profiles to hold the new contents of the new data space. For more information, see [“The RACGLIST class” on page 238](#) and [“Using RACROUTE REQUEST=LIST,GLOBAL=YES support” on page 237](#).

Note that you must issue this command each time you want RACF to perform the refresh process. The following example shows how to activate refreshing of SETROPTS RACLIST processing for the DASDVOL and TERMINAL classes.

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

For information about SETROPTS REFRESH processing on shared systems, see [“Refreshing shared systems \(REFRESH option\)” on page 131](#).

## SETROPTS REFRESH option for special cases

RACF provides the SETROPTS REFRESH operand to allow the administrator to refresh profiles in any situation. The options described in this topic are:

- GENERIC REFRESH
- GLOBAL REFRESH

- REFRESH for shared systems

## Refreshing in-storage generic profile lists (GENERIC REFRESH option)

If you have the SPECIAL, AUDITOR, or OPERATIONS attribute, you can initiate the refreshing of in-storage generic profile lists by specifying the GENERIC and REFRESH operands on the SETROPTS command.

When you specify GENERIC and REFRESH, you also specify one or more classes for which you want RACF to refresh in-storage generic profile lists. This causes all of the in-storage generic profiles in the specified general resource class (except those in the global access checking table) to be replaced with new copies from the RACF database. Note that you must issue this command each time you want RACF to perform the refresh process.

To refresh the profiles, issue the SETROPTS GENERIC(*classname*) REFRESH command. RACF processes *classname* and all classes that share the same POSIT value on their class descriptor table (CDT) entries.

The following example shows how to refresh in-storage generic profiles for the DATASET and TERMINAL classes.

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

If you use SETROPTS GENLIST to activate shared in-storage generic profiles for a general resource class, RACF refreshes the profiles as well as the profile lists for that class when you specify the class with GENERIC and REFRESH. For more information, see [“SETROPTS options to activate in-storage profile processing” on page 125](#).

If you specify SETROPTS GENERIC(\*) REFRESH, RACF refreshes profile lists for the DATASET class and all active classes except resource grouping classes and classes defined with the GENERIC(DISALLOWED) attribute.

If you specify NOGENERIC on the SETROPTS command, RACF stops using in-storage generic profile lists but does not immediately delete them. RACF deletes the profile lists at the end of the job or TSO session, or when you again specify GENERIC. When you specify GENERIC, RACF rebuilds the profile lists.

**Note:** You must have the SPECIAL attribute to issue the SETROPTS GENERIC command by itself. However, to issue SETROPTS GENERIC (*classname*) REFRESH, you do not need the SPECIAL attribute. However, you must have the group-SPECIAL, group-AUDITOR, group-OPERATIONS, AUDITOR, or OPERATIONS attribute.

For information about SETROPTS REFRESH processing on shared systems, see [“Refreshing shared systems \(REFRESH option\)” on page 131](#).

## Refreshing global access checking lists (GLOBAL REFRESH option)

If you have the SPECIAL attribute, you can initiate the refreshing of global access checking lists by specifying the GLOBAL and REFRESH operands on the SETROPTS command. When you specify GLOBAL and REFRESH, also specify the class for which you want RACF to refresh global access checking lists. Note that you must issue this command each time you want RACF to perform the refresh process.

To refresh the profiles, issue the SETROPTS GLOBAL(*classname*) REFRESH command. RACF processes *classname* and all classes that share the same POSIT value on their class descriptor table (CDT) entries.

The following example specifies that you want RACF to refresh global access checking lists for the DATASET and TERMINAL classes.

```
SETROPTS GLOBAL(DATASET TERMINAL) REFRESH
```

If you specify GLOBAL (\*), RACF refreshes access checking lists for the DATASET class and all active classes in the class descriptor table (CDT).

If you specify NOGLOBAL, you disable global access checking for the class that you specify.

For information about SETROPTS REFRESH processing on shared systems, see [“Refreshing shared systems \(REFRESH option\)” on page 131](#).

## Refreshing shared systems (REFRESH option)

If two or more systems that are *not* enabled for sysplex communication are sharing a RACF database, SETROPTS REFRESH processing applies only to the system on which you issue the SETROPTS command. With this type of sharing, you must issue the SETROPTS command on all of the systems to perform REFRESH processing on all of the systems. If you do not issue the SETROPTS REFRESH command on one of the shared systems, REFRESH is performed for that system when the system re-IPLs.

On systems that are enabled for sysplex communication, issue the REFRESH command only once. If the command is successful, it is propagated to the other members of the data sharing group.

If the command fails on any of the peer systems and the system is in data sharing mode, RACF stops processing the command and backs it out of all the member systems, including the system on which it was issued.

In non-data sharing mode, the command can fail on a peer system without backing out of the other systems.

## SETROPTS options for special purposes

Some options are useful for special purposes. You can specify them at any time, but you might not want to specify them when RACF is first installed. These options include:

- TERMINAL
- CLASSACT for the SECDATA and SECLABEL classes
- SESSIONINTERVAL
- WHEN(PROGRAM)

## Protecting undefined terminals (TERMINAL option)

If you have the SPECIAL attribute, you can specify the universal access authority (UACC) that RACF uses when users attempt to log on to TSO from terminals that are not defined to RACF. You can specify this option with the TERMINAL operand of the SETROPTS command, as shown in the following example:

```
SETROPTS TERMINAL (READ|NONE)
```



**Attention:** Before you specify NONE, be sure that you define your terminals to RACF and give the appropriate users and groups proper authorization to use them. Otherwise, users cannot access the terminals.

When you specify READ or NONE, you establish a default UACC for all users to undefined terminals on your system. If you specify READ, all users can access all terminals on your system (if allowed to by the security classifications of the terminals). If you specify NONE, only users and groups that you authorize to use a terminal through its access list can use it. If you do not specify either READ or NONE, the default value is READ. For more detailed information, see [“Protecting terminals” on page 225](#).

For undefined terminals, READ access authority is in effect when a RACF database is first initialized using IRRMIN00.

## Activating the security classification of users and data

If you have the SPECIAL attribute, you can activate security classification of users and data by specifying CLASSACT(SECDATA) or CLASSACT(SECLABEL) on the SETROPTS command, as shown in the following examples:

```
SETROPTS CLASSACT(SECDATA)
SETROPTS CLASSACT(SECLABEL)
```

Security classification of users and data allows an installation to impose additional access controls on resources by defining security levels, security categories, and security labels for both users and resources.

**Note:** You can create profiles in the SECDATA and SECLABEL classes, and specify security classifications in resource and user profiles, without actually activating the SECDATA and SECLABEL classes. When authorization checking occurs, the security classifications are ignored by RACF. This allows you to "label" the profiles without enforcing the security classifications.

If you have the SPECIAL attribute, you can also deactivate security classification of users and data by specifying NOCLASSACT(SECDATA) or NOCLASSACT(SECLABEL), as appropriate, on the SETROPTS command.

NOCLASSACT(SECDATA) and NOCLASSACT(SECLABEL) are in effect when a RACF database is first initialized using IRRMIN00.

## Establishing the maximum VTAM session interval (SESSIONINTERVAL option)

If you have the SPECIAL attribute, you can set the maximum number of days that any session segment password in an APPCLU profile can go without being changed. If any APPCLU profile has a higher interval limit, the SETROPTS value is used instead.

To set the limit, enter:

```
SETROPTS SESSIONINTERVAL(n)
```

where *n* is the maximum number of days that can be specified and must be between 1 and 32767.

To remove the system-wide maximum (allowing any value in the profiles to take effect), enter:

```
SETROPTS NOSESSIONINTERVAL
```

## Activating program control (WHEN(PROGRAM) option)

If you have the SPECIAL attribute, you can activate program control by using the WHEN(PROGRAM) operand of the SETROPTS command. When program control is active, RACF provides access control to load modules, and program access to data sets and SERVAUTH resources. The following example shows how to specify this option:

```
SETROPTS WHEN(PROGRAM)
```

Access control to load modules allows only authorized users to load and execute specified load modules (programs). RACF uses profiles in the PROGRAM general resource class to control access to programs.

Program access to data sets allows an authorized user or group of users to access specified data sets in conjunction with the user's authority to execute a certain program. That is, some users can access specified data sets at a specified access level only while executing a certain program.

Program access to SERVAUTH class resources allows an authorized user or group of users to access certain IP addresses in conjunction with the user's authority to execute a certain program. That is, some users can access specified IP addresses at a specified access level only while executing a certain program.

If you have the SPECIAL attribute, you can also deactivate program control by using the NOWHEN(PROGRAM) operand on the SETROPTS command.

NOWHEN(PROGRAM) is in effect when a RACF database is first initialized using IRRMIN00.

### **Note:**

1. If the system is enabled for sysplex communication and a command is successful on the system on which it was issued, RACF propagates the command to the other members of the data sharing group.
2. If the command fails on any of the peer systems and the system is in data sharing mode, RACF stops processing the command and backs it out of all the member systems, including the system on which it was issued.

3. If the system is not enabled for sysplex communication, the command does not take effect on the other systems sharing the database until you issue it on those systems or the systems are IPLed.
4. In non-data sharing mode, the command can fail on a peer system without backing out of the other systems.

For more information, see [Chapter 13, “Protecting programs,” on page 301.](#)

## SETROPTS options related to security labels

Several options are related to security labels. They include:

- COMPATMODE (See [“Activating compatibility mode for security labels \(COMPATMODE option\)” on page 135.](#))
- MACTIVE (See [“Enforcing multilevel security \(MLACTIVE option\)” on page 135.](#))
- MLFSOBJ (See [“Restricting access to z/OS UNIX files and directories \(MLFSOBJ option\)” on page 137.](#))
- MLIPCOBJ (See [“Restricting access to interprocess communication objects \(MLIPCOBJ option\)” on page 137.](#))
- MLNAMES (See [“Using name-hiding \(MLNAMES option\)” on page 138.](#))
- MLQUIET (See [“Quiescing RACF activity \(MLQUIET option\)” on page 134.](#))
- MLS (See [“Preventing the copying of data to a lower security label \(SETROPTS MLS option\)” on page 134.](#))
- MLSTABLE (See [“Preventing changes to security labels \(MLSTABLE option\)” on page 133.](#))
- SECLABELCONTROL (See [“Restricting changes to security labels \(SECLABELCONTROL option\)” on page 133.](#))
- SECLBYSYSTEM (See [“Activating security labels by system image \(SECLBYSYSTEM option\)” on page 138.](#))

**Note:** All of the above SETROPTS options related to security labels, except MLNAMES, require the SECLABEL class to be active. The MLNAMES option can be used while the SECLABEL class is not active.

### Restricting changes to security labels (SECLABELCONTROL option)

If you have the SPECIAL attribute, you can prevent users who do not have the SPECIAL attribute from doing either of the following:

- Specifying or changing a security label in a resource profile
- Changing a SECLABEL profile using the RALTER command

To place this control into effect, enter:

```
SETROPTS SECLABELCONTROL
```

When the SECLABELCONTROL option is in effect, only certain users can specify the SECLABEL operand on RACF commands:

- Users with the SPECIAL attribute can specify the SECLABEL operand on any RACF command.
- Users with the group-SPECIAL attribute can specify the SECLABEL operand only on the ADDUSER and ALTUSER commands when they add a user to a group within their scope of control. Also, group-SPECIAL users must be permitted to the SECLABEL profiles with at least READ access authority.
- Users without the SPECIAL attribute cannot specify the SECLABEL operand.

To cancel this option, specify NOSECLABELCONTROL on the SETROPTS command.

### Preventing changes to security labels (MLSTABLE option)

If you have the SPECIAL attribute, and if the SECLABEL class is active, you can prevent users from changing the classification of data while the data is in use. Specifically, you can do all of the following:

- Prevent any user from changing the security label of a RACF profile
- Prevent any user from changing a SECLABEL profile using the RALTER command unless SETROPTS MLQUIET is in effect. Changes to the access list using the PERMIT command are allowed.

To do this, enter:

```
SETROPTS MLSTABLE
```

**Restriction:** This option cannot be activated when the SECLABEL class is inactive.

To cancel this option, specify NOMLSTABLE on the SETROPTS command.

**Note:** If you must change security labels while the system is in multilevel stable state, you can issue SETROPTS MLQUIET before making the changes. See [“Quiescing RACF activity \(MLQUIET option\)”](#) on page 134.

## Quiescing RACF activity (MLQUIET option)

If you have the SPECIAL attribute, and MLSTABLE is in effect, you can prevent users other than SPECIAL users, console operators, and started procedures from logging on, starting new jobs, or accessing resources. This prevents them from using the RACROUTE AUTH, DEFINE, and VERIFY requests.

To do this, enter:

```
SETROPTS MLQUIET
```

To cancel this option, specify NOMLQUIET on the SETROPTS command.

**Restriction:** If SETROPTS MLSTABLE is not in effect, the SETROPTS MLQUIET command is ignored.

**Note:** Do not specify SETROPTS MLQUIET if any system sharing the RACF database is not at the necessary software level for multilevel security support. For details, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

Use of the MLQUIET option can prevent a successful IPL of these systems. In addition, if no systems sharing the RACF database are capable of multilevel security, you might have to IPL with RACF inactive and update the database with the block update command (BLKUPD) in order to turn off the MLQUIET option.

## Preventing the copying of data to a lower security label (SETROPTS MLS option)

If you have the SPECIAL attribute, and if the SECLABEL class is active, you can prevent unauthorized users from copying data from a resource with one security label to a resource with a lower security label. This protection is also called controlling "writedown".

To do this, enter:

```
SETROPTS MLS(FAILURES)
```

### Restrictions:

- You cannot activate this option when the SECLABEL class is inactive.
- The resource you want to protect must be in a resource class that is defined in the CDT with neither the RVRSMAC nor EQUALMAC attribute. (If the class has the RVRSMAC attribute, users are prevented from *writing-up*. If the class as EQUALMAC, users are not restricted in their write actions.)

You can authorize certain users to copy data from a resource with one security label to a resource with a lower security label by defining and controlling the *writedown* privilege. (For more information, see [“Controlling the write-down privilege”](#) on page 101.)

You can specify `MLS(WARNING)`, rather than `MLS(FAILURES)`, to allow the user request, but to send a warning message to the user and the security administrator. If you do not specify the `FAILURES` option with the `SETROPTS MLS` command, then `MLS(WARNING)` will be activated.

**Restriction:** `SETROPTS MLS(WARNING)` does not apply to resources controlled by the `SETROPTS MLFSOBJ` option (z/OS UNIX files and directories) and the `SETROPTS MLIPCOBJ` option (interprocess communication objects).

To cancel the `SETROPTS MLS` option, specify `NOMLS` on the `SETROPTS` command.

## Activating compatibility mode for security labels (COMPATMODE option)

If you are using security labels on your system, and you observe security label failures for some calls at the designated security console or in audit records, the reason might be that the caller used a pre-RACF 1.9 protocol that did not, or was unable to, specify a security label.

If this was the case, and you want to have security label authorization checks succeed for those callers who are not using current protocols, you might be able to use the `COMPATMODE` option on the `SETROPTS` command to do so. Specifying `COMPATMODE` allows the caller to access the resources it needs, providing the user has access to a security label that could allow the requested access to the resource.

To establish `COMPATMODE`, enter:

```
SETROPTS COMPATMODE
```

**Restriction:** This option cannot be activated when the `SECLABEL` class is inactive.

To investigate the source of a security label failure, obtain a copy of the RACF audit records using output from the SMF data unload utility (`IRRADU00`). (See [z/OS Security Server RACF Auditor's Guide](#).) Examine the records for the call to see if the failure occurred because of insufficient security label authority. Next, examine the token information for the caller. If the caller's token is identified as being created by a pre-RACF 1.9 protocol that either did not, or was unable to, specify a security label, RACF failed the security label authorization check.

`NOCOMPATMODE` is in effect when a RACF database is first initialized using `IRRMIN00`.

## Enforcing multilevel security (MLACTIVE option)

If you have the `SPECIAL` attribute, and if the `SECLABEL` class is active, you can control whether security labels are required for certain resource classes. When `MLACTIVE` is in effect, the following requirements are enforced:

### Requirements:

- All work entering the system must be run by a RACF-defined user.
- A security label must be assigned to all work entering the system, including batch jobs and users logging on to TSO and MVS consoles, started procedures, and to any application that supports security labels when users log on.
- All user tasks running in a server's address space must have a security label that is equivalent to the security label of the address space.
- You must either assign and grant permission to a default security label for every RACF user ID, or permit user IDs to `SYSLOW`. Users without a default security label will attempt to run with `SYSLOW` when `MLACTIVE(FAILURES)` is in effect.
- A security label must be assigned to all profiles in the following classes:
  - `APPCPORT`
  - `APPCSERV`
  - `APPCTP`
  - `APPL`
  - `DATASET`

## RACF options

- DEVICES
- DIRECTRY
- DSNADM
- DSNR
- FILE
- GDSNBP and MDSNBP
- GDSNCL and MDSNCL
- GDSNDB and MDSNDB
- GDSNJR and MDSNJR
- GDSNPN and MDSNPN
- GDSNSC and MDSNSC
- GDSNSG and MDSNSG
- GDSNSM and MDSNSM
- GDSNSP and MDSNSP
- GDSNTB and MDSNTB
- GDSNTS and MDSNTS
- GDSNUF and MDSNUF
- SERVAUTH
- SERVER
- TAPEVOL
- TERMINAL
- USER
- VMDEV
- VMLAN
- VMMAC
- VMMDISK
- VMSEGMT
- WRITER

To enforce multilevel security, enter:

```
SETROPTS MLACTIVE (FAILURES)
```

**Restriction:** This option cannot be activated when the SECLABEL class is inactive.

You can also specify MLACTIVE(WARNING), which allows the users to log on or submit jobs. MLACTIVE(WARNING) sends a warning message to the user and to the security administrator when the user attempts to:

- Enter the system without a security label
- Access a resource in one of the previously mentioned classes but the resource has not been assigned a security label

If you do not specify the FAILURES option with the SETROPTS MLACTIVE command, then MLACTIVE(WARNING) will be activated.

To cancel the MLACTIVE option, specify NOMLACTIVE on the SETROPTS command.

**Attention:** Do not issue the SETROPTS MLACTIVE(FAILURES) command unless you have assigned appropriate security labels to users and to the resources they must access. To recover from such a situation, logon as a user with the SPECIAL attribute, specifying SYSHIGH as the current security label.



Then, either assign security labels or issue SETROPTS NOMLACTIVE. If you turn on MLACTIVE and do not correctly define all profiles that need SECLABELs, IPL failures, or other serious problems can occur.

#### Guidelines:

- Back up your RACF database with a database that you know you can use to IPL.
- Define new system profiles (including classes such as DATASET, TERMINAL, TAPEVOL, APPL or any other active class that has SLBLREQ=YES in the class descriptor table) and ensure they have the correct security labels.
- Turn MLACTIVE on in WARNING mode.
- Watch out for relevant warning messages.

**Data set and general resource profiles in WARNING mode:** A user or task *can* access a resource that is in WARNING mode and has no security label even when MLACTIVE(FAILURES) is in effect and the class requires security labels. The user or task receives a warning message and gains access. (A data set or general resource is in WARNING mode when you define or modify the profile that protects it and you specify the WARNING operand.)

## Restricting access to z/OS UNIX files and directories (MLFSOBJ option)

If you have the SPECIAL attribute, and if the SECLABEL class is active, you can prevent users (except trusted and privileged started tasks) from accessing z/OS UNIX file system resources, such as files and directories, that do not have security labels. While the SETROPTS MLFSOBJ option is in effect, all z/OS UNIX file system resources must have security labels.

To do this, enter:

```
SETROPTS MLFSOBJ(ACTIVE)
```

**Restriction:** This option cannot be activated when the SECLABEL class is inactive.

To cancel the MLFSOBJ option, specify MLFSOBJ(INACTIVE) on the SETROPTS command.

**Note:** Do not specify SETROPTS MLFSOBJ(ACTIVE) if any system sharing the RACF database is not at the necessary software level for multilevel security support. Use of the SETROPTS MLFSOBJ option should not cause problems on these systems, but it does not provide full protection on these systems. For details, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

## Restricting access to interprocess communication objects (MLIPCOBJ option)

If you have the SPECIAL attribute, and if the SECLABEL class is active, you can prevent users (except trusted and privileged started tasks) from accessing objects used for interprocess communication, such as semaphores, message queues, and shared memory, that do not have security labels. While the SETROPTS MLFSOBJ option is in effect, all interprocess communication objects must have security labels.

To do this, enter:

```
SETROPTS MLIPCOBJ(ACTIVE)
```

**Restriction:** This option cannot be activated when the SECLABEL class is inactive.

To cancel the MLIPCOBJ option, specify MLIPCOBJ(INACTIVE) on the SETROPTS command.

**Guideline:** Assign security labels to all users before activating MLIPCOBJ to ensure that all interprocess communication objects in progress are assigned security labels. One way to ensure this is to activate MLIPCOBJ at IPL time.

**Note:** Do not specify SETROPTS MLIPCOBJ(ACTIVE) if any system sharing the RACF database is not at the necessary software level for multilevel security support. Use of the SETROPTS MLIPCOBJ option should not cause problems on these systems, but it does not provide full protection on these systems. For details, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

## Using name-hiding (MLNAMES option)

If you have the SPECIAL attribute, and if the SECLABEL class is active, you can prevent users from viewing the names of data sets, files and directories that they cannot read based on their current security labels. This option is known as *name-hiding*.

To do this, enter:

```
SETROPTS MLNAMES
```

To cancel the MLNAMES option, specify NOMLNAMES on the SETROPTS command.

When MLNAMES is active, users who list catalogs and directories will not see the names of data sets and files they cannot currently access. If the SECLABEL class is not active while MLNAMES is active, data set names will still be hidden from users who do not have at least READ access to the data sets. For details about name-hiding, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

**Note:** Do not specify SETROPTS MLNAMES if any system sharing the RACF database is not at the necessary software level for multilevel security support. Use of the SETROPTS MLNAMES option should not cause problems on these systems, but it does not provide full protection on these systems.

## Activating security labels by system image (SECLBYSYSTEM option)

If you have the SPECIAL attribute, and if the SECLABEL class is active, you can allow activation of security labels on a system image basis. Specify the SMF ID of each selected system in the member list of profiles in the SECLABEL class to indicate that a particular security label is active on that system.

### Rules:

1. Security labels that are not active on a particular system cannot be used by users without the SPECIAL or AUDITOR attribute on that system, or listed by users without the SPECIAL, AUDITOR, or ROAUDIT attribute on that system.
2. If you define a security label with no member list, the security label is active on all systems.
3. If you specify a member list for the following security labels, it will be ignored:
  - SYSHIGH
  - SYSLOW
  - SYSNONE
  - SYSMULTI

When SECLBYSYSTEM is in effect, a batch job submitted with no security label executes with the security label of the JESINPUT class profile, unless the JESINPUT class security label is SYSMULTI.

After activating SECLBYSYSTEM, you must issue SETROPTS RACLIST(SECLABEL) REFRESH to complete the activation of security labels by system. This option cannot be activated when the SECLABEL class is inactive.

To activate this option, enter:

```
SETROPTS SECLBYSYSTEM
SETROPTS RACLIST(SECLABEL) REFRESH
```

To cancel the SECLBYSYSTEM option, specify NOSECLBYSYSTEM on the SETROPTS command. Then, issue the SETROPTS RACLIST(SECLABEL) REFRESH.

**Note:** Do not specify SETROPTS SECLBYSYSTEM if any system sharing the RACF database is not at the necessary software level for multilevel security support. Use of the SETROPTS SECLBYSYSTEM option should not cause problems on these systems, but it does not provide full protection on these systems. For details, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

## SETROPTS options for automatic control of access list authority

Use of the SETROPTS options ADDCREATOR and NOADDCREATOR allows you to specify whether the user ID of the person defining a resource profile is automatically placed on the access list for that resource with ALTER authority. The options are:

- ADDCREATOR
- NOADDCREATOR

### Automatic addition of creator's user ID to access list

The ADDCREATOR option indicates that the profile creator's user ID is placed on the profile access list with ALTER authority when any new DATASET or general resource profiles is defined using ADDSD, RDEFINE, or RACROUTE REQUEST=DEFINE.

This option is the default unless IRRMIN00 is run with PARM=NEW.

### Automatic omission of creator's user ID from access list

The NOADDCREATOR option indicates that the profile creator's user ID is not placed on the profile access list with ALTER authority under the following conditions:

- When any new DATASET or general resource profiles is defined using ADDSD, RDEFINE, or creating a generic profile through RACROUTE REQUEST=DEFINE
- When a discrete profile (other than the DATASET and TAPEVOL classes) is created through RACROUTE REQUEST=DEFINE

Even if the NOADDCREATOR option is used, in the DATASET and TAPEVOL classes created through RACROUTE REQUEST=DEFINE, the user ID of any profile creator is placed on the new profile's access list with ALTER authority.

This will occur when a user creates a permanent data set, if the user has ADSP and ADSP is active on the system, or when a user specifies PROTECT or SECMODEL on the JCL DD statement or TSO allocate command for a new permanent data set.

If IRRMIN00 is run with PARM=NEW, this option is the default.

## Specifying the encryption method for user passwords

RACF keeps user authentication data secure when it is stored in the RACF database. This authentication data can be a password, password phrase, or operator identification card (OIDCARD) data. Passwords and password phrases are referred to collectively as "passwords".

RACF supports several different methods to encrypt authentication data:

1. Masking
2. The data encryption standard (DES) algorithm.
3. An installation-defined method that is implemented that uses the ICHDEX01 exit.
4. The KDFAES (key derivation function with AES256) algorithm (for passwords).

Encoding functions that are performed by RACF are:

- Data encoding
- Data comparison

*Encoding* means that, given data in clear text and given an encryption key (which RACF constructs), the equivalent data is produced in encrypted form. RACF provides a "one-way" encoding. That is, data encrypted by RACF can only be decoded if the data is already known. For more details, see *z/OS Security Server RACF System Programmer's Guide*.

*Comparison* means that, given authentication data as entered by a user (in clear text form) and given that data as stored in the RACF database in encoded form, an indication whether they are equal or not is returned.

By default, RACF uses the DES algorithm to encrypt and compare authentication data.

The DES, masking, and installation-defined methods are collectively referred to as “legacy” methods in some contexts. When KDFAES is not enabled, the presence and function of the ICHDEX01 exit determines which of these algorithms is active.

**Guideline:** Do not run your system with an ICHDEX01 exit unless you are using it to implement your own encryption algorithm.

For more information about the ICHDEX01 password authentication exit, see *z/OS Security Server RACF System Programmer's Guide*.

The following SETROPTS command is used to enable KDFAES:

```
SETROPTS PASSWORD(ALGORITHM(KDFAES))
```

**Note:** Review the [Planning Considerations for enabling KDFAES in z/OS Security Server RACF System Programmer's Guide](#) prior to enabling KDFAES.

KDFAES is the strongest algorithm and is recommended to be used when possible. This algorithm is resilient against offline brute-force password attacks if your RACF database or one of its copies is compromised.

KDFAES is used for passwords, but not for OIDCARD data. When KDFAES is active, OIDCARD data continues to be protected by a legacy algorithm. When KDFAES is enabled, existing DES and installation-encoded passwords continue to evaluate correctly. When they are changed, they are encrypted using KDFAES. When KDFAES is active, the masking algorithm is no longer used to evaluate legacy passwords, and masked passwords must be changed by the administrator after KDFAES is enabled.

KDFAES passwords require more space in the RACF database than the legacy methods require. This changed format requires updates from some other software applications for them to keep functioning under the new algorithm. Be sure that you have all the available updates before you enable KDFAES.

If you need to disable KDFAES, the following SETROPTS command can also be used:

```
SETROPTS PASSWORD(NOALGORITHM)
```

After KDFAES is disabled, all legacy rules apply, however, KDFAES passwords continue to evaluate correctly. When they are changed, they are encrypted using the legacy algorithm in effect, which is determined by the presence and function of ICHDEX01.

## Performing a password conversion

After KDFAES is enabled, you might want to strengthen passwords immediately, rather than wait for users to change them on their normal schedule. For example, you might want to prove compliance with a security policy that requires the strongest possible algorithm. The PWCONVERT operand of the ALTUSER command can be used. PWCONVERT converts a DES password and DES password history entries to KDFAES format without requiring the password to be changed.

A sample Db2z/OS query against IRRDBU00 output is provided to report users that have a legacy format current password or phrase. See the RACDBUQR member of SYS1.SAMPLIB.

See the *z/OS Security Server RACF System Programmer's Guide* for more information about the PWCONVERT function.

### Notes:

1. Conversion assumes that the existing password and password history are in DES format. Do not perform the conversion if you have passwords that are masked or installation-encoded.
2. Conversion does not affect password phrases or phrase history.

3. If an older copy of your RACF database containing DES passwords is compromised and an attacker is able to guess a password value, a conversion to KDFAES will not protect against the attacker using that password to gain access to your system. If you suspect your database has been compromised, user passwords should be changed as soon as possible. The EXPIRED operand of the ALTUSER command can be used to force a user to change his password when he next logs on.

## Using started procedures

A *procedure* consists of a set of job control language statements that are frequently used together to achieve a certain result. Procedures usually reside in the system procedure library, SYS1.PROCLIB, which is a partitioned data set. A *started procedure* is usually started by an operator, but can be associated with a functional subsystem. For example, DFSMS is treated as a started procedure even though it does not need to be specifically started with a START command.

Only RACF-defined users and groups can be specifically authorized to access RACF-protected resources. However, started procedures have system-generated JOB statements that do not contain the USER, GROUP, or PASSWORD parameter.

To enable started procedures to access the same RACF-protected resources that users and groups access, started procedures must have RACF user IDs and group names. By assigning them RACF identities, your installation can give started procedures specific authorization to access RACF-protected resources. For example, you can allow JES to access spool data sets. To associate the names of started procedures with specific RACF group names and user IDs, you and your RACF system programmer can do one of the following:

1. Set up the STARTED class (the preferred method).
2. Create a started procedures table (ICHRIN03).

## Assigning RACF user IDs to started procedures

As with any other user ID and group name, the user ID and group name that you assign to a started procedure must be defined to RACF using the ADDUSER and ADDGROUP commands, and the user must be connected to the group. You might also need to use the PERMIT command to authorize the users or groups to get access to the required resources. See [z/OS Security Server RACF Command Language Reference](#) for descriptions of these commands.

### Protected user IDs

The user IDs that you assign to started procedures should have the PROTECTED attribute. Protected user IDs are user IDs that have the NOPASSWORD, NOPHRASE, and NOIDCARD attributes. They are defined or modified using the ADDUSER and ALTUSER commands. See [“Defining protected user IDs” on page 74](#) for more information.

Protected user IDs cannot be used to logon to the system, and are protected from being revoked through incorrect system access attempts. The following example shows a protected user ID being defined for a CICS region, and an existing user ID used by JES being given the PROTECTED attribute:

```
ADDUSER CICS03 DFLTGRP(STCGROUP) OWNER(STCADMIN) NOPASSWORD
ALTUSER JES DFLTGRP(STCGROUP) OWNER(STCADMIN) NOPASSWORD NOPHRASE
```

If you do not specify NOPASSWORD for a user ID assigned to a started procedure, you should specify a password and change the password periodically. If you do not specify a password and do not specify NOPASSWORD, RACF uses the default group name as the password. Anyone who knows this user ID and password combination can gain access to any resource that the started procedure can access.

See [“Using protected user IDs for batch jobs” on page 458](#) for more information.

**Note:** If the associated user ID is revoked for any reason, the started procedure might have problems allocating new SMS-managed data sets, submitting batch jobs, and obtaining printed output.

## Undefined user IDs

A started procedure runs as an undefined user if:

1. It is executed without associating its name with a RACF-defined user ID and group name.
2. The user or group is not defined.
3. The user is not connected to the group.

A started procedure running as an undefined user can access RACF-protected resources if the universal access authority for the resource is sufficient to allow the requested operation. However, if a started procedure requires access at a higher level than universal access, you *must* associate the started procedure with a RACF-defined user ID and group name.

## Authorizing access to resources

A started procedure can gain access to RACF-protected resources in the following ways:

1. By the user ID or group name assigned as for any other user of the system (for example, universal access, entry and access list, and OPERATIONS).
2. By having the privileged attribute, which allows the started procedure to pass all authorization checking (unless the CSA or PRIVATE operand is specified on the RACROUTE request). No installation exits are called, no SMF records are generated, and no statistics are updated. (Note that bypassing authorization checking includes bypassing the checks for security classification of users and data.)
3. By having the trusted attribute, which means the same as privileged, except that you can request an audit using the SETROPTS LOGOPTIONS command.

## Setting up the STARTED class

With the STARTED class, you do not need to change code or re-IPL the system in order to add or modify RACF identities for started procedures. You can modify the security definitions for started procedures *dynamically*, using the RDEFINE, RALTER, and RLIST commands. See [z/OS Security Server RACF Command Language Reference](#) for more information on these commands. In effect, the STARTED class provides a *dynamic* started procedures table.

To set up the STARTED class, enter these commands:

### Example:

```
SETROPTS GENERIC(STARTED)

RDEFINE STARTED JES2.* UACC(NONE)
  STDATA(USER(JES2) GROUP(STCGROUP) TRUSTED(YES))
RDEFINE STARTED ** UACC(NONE)
  STDATA(USER(=MEMBER) GROUP(STCGROUP) TRACE(YES))

SETROPTS CLASSACT(STARTED) RACLIST(STARTED)
  or, if the STARTED class is already in use:
SETROPTS RACLIST(STARTED) REFRESH
```

**Note:** Once you alter the TRUSTED attribute for the STARTED class, you must recycle the started task to pick up the change. This is because the TRUSTED attribute is part of the ACEE which is built when the address space starts up, so you have to recycle the task.

## Defining profile data

Profiles in the STARTED class include the STDATA segment, which contains fields for user ID, group name, trusted flag, privileged flag, and trace flag:

- The user ID can be a RACF user ID or the character string =MEMBER, which indicates that the member name is to be used as the user ID.
- The group name can be a RACF group name or the character string =MEMBER, which indicates that the member name is to be used as the group name.



- If tracing is specified, RACF issues operator message IRR812I during RACROUTE REQUEST=VERIFY or VERIFYX to indicate which profile is used.

This message can be used during diagnosis of security problems with started procedures, to determine which profile was used for a particular started procedure.

RACF performs partial diagnosis when creating the STDATA segment to help you define profiles that work correctly. For example, RACF verifies that a specified user ID is connected to the group name, if specified.



#### Attention:

- Be sure to specify a group name (not =MEMBER) as the GROUP value of the STDATA segment, if both of the following are true:
  1. The profile name contains generic characters (\*, %, or &).
  2. The USER value of the STDATA segment is the character string =MEMBER.

If you do not specify a group name, a new started procedure or job could be assigned on execution to a user ID that matches an existing user ID on your system. Consider defining a special group (for example, STCGROUP) for started procedures and job user IDs, and using this group name as the GROUP value of the STDATA segment.

- In addition, be careful which libraries your started procedures come from and do not let your users update them. Refer to the JES customization manuals for information on specifying procedure libraries.

## Specifying STARTED class profile names

You can start jobs as well as procedures using the START command. The START command specifies the member name to start and, optionally, the job name to use. See [z/OS MVS System Commands](#) for more information on the START command.

For started procedure (job) address spaces, resource names in the STARTED class are of the form *member.job*, where:

#### **member**

The 1 - 8 character name of a member of a partitioned data set that contains the source JCL for the task to be started. The member can be a job or a started procedure.

#### **job**

The name identifying the procedure to be started. If the START command does not specify a job name, and the member does not contain a JOB statement to supply a job name, the system uses the member name as the job name when constructing the resource name for STARTED class processing.

For system address spaces, resource names are of the form *job.job*, or (if your system programmer used the JOBSpace option of the ATTR parameter of the ASCRE macro to create the address space), *member.job*. To determine which form to use for a procedure associated with a system address space, refer to the documentation for the application or consult the programmer. For programming details about creating address spaces, see [Creating address spaces](#) in [z/OS MVS Programming: Extended Addressability Guide](#).

For each sample START command issued for a started procedure (job) address space, [Table 8 on page 143](#) shows the resource name that will be used for STARTED class processing, and a list of names for STARTED class profiles that could be defined for each STARTED class resource.

*Table 8. Sample profile names for STARTED class resources*

START command	STARTED class resource name	STARTED class profile name
S CICS	CICS.CICS	CICS.CICS
		CICS.*
		CICS.**

*Table 8. Sample profile names for STARTED class resources (continued)*

<b>START command</b>	<b>STARTED class resource name</b>	<b>STARTED class profile name</b>
S CICSP	CICSP.CICSP	CICSP.CICSP
		CICSP.*
		CICSP.**
		CICS*.**
		CICS*
S CICST	CICST.CICST	CICST.CICST
		CICST.*
		CICST.**
		CICS*.**
		CICS*
S IMS, JOBNAME=IMSPROD	IMS.IMSPROD	IMS.IMSPROD
		IMS.IMSP*
		IMS.*
		IMS.**
S IMS, JOBNAME=IMSTEST	IMS.IMSTEST	IMS.IMSTEST
		IMS.IMST*
		IMS.*
		IMS.**

## Using the started procedures table (ICHRIN03)

Your RACF system programmer can use the started procedures table (ICHRIN03) to associate the names of started procedures with specific RACF user IDs and group names.

The started procedures table can also contain a generic entry that assigns a user ID or group name to any started task that does not have a matching entry in the table. In this case, the table can specify that the started task name should be used as the user ID or group name, or it can assign a specific user ID or group name to the started task. You should ensure that the generic entry, if used, assigns a generic user ID or group name (for example, STCGROUP).

To modify the security definitions for started procedures using the started procedures table, you need to:

1. Edit the started procedures table.
2. Assemble and link-edit the updated table.
3. Re-IPL the system.

See *z/OS Security Server RACF System Programmer's Guide* for information on how to code the started procedures replaceable module, and for a complete description of the started procedures table (ICHRIN03).

## Started procedure considerations

Here are some things to consider when you use started procedures:

1. Even if your installation uses the STARTED class, you must have a started procedures table (ICHRIN03). RACF cannot be initialized if ICHRIN03 is not present. A dummy ICHRIN03 is shipped



with and installed by RACF. If you use the STARTED class, you should leave your existing ICHRIN03 in place, in case, for example, someone unintentionally deactivates the STARTED class. For more information, see *z/OS Security Server RACF System Programmer's Guide*.

2. For installations that have an existing started procedures table (ICHRIN03) and want to use the STARTED class, a sample REXX exec is provided in member ICHSPTCV in SYS1.SAMPLIB to process the output of ICHDSM00 and build RDEFINE commands to duplicate an existing started procedures table.
3. To make sure that critical system tasks (those marked TRUSTED or PRIVILEGED in ICHRIN03) start successfully, define specific STARTED profiles for them in the STARTED class.
4. **Guideline:** Assign the TRUSTED attribute when one of the following conditions applies:
  - The started procedure or address space creates or accesses a wide variety of unpredictably named data sets within your installation.
  - Insufficient authority to an accessed resource might risk an unsuccessful IPL or other system problem.

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

5. When the STARTED class is active, RACF uses it before using the started procedures table (ICHRIN03). A generic profile such as \*\* or \*. \* with a valid STDATA segment will override all the entries in ICHRIN03.
6. To make sure that RACF uses the STARTED class, you should verify that all START commands have a matching profile with an STDATA segment that assigns a user ID. To do this:
  - a. Define an appropriate generic profile that matches all possible START commands (for example, \*\* or \*. \*).
  - b. Specify =MEMBER or a user ID of limited privileges.
  - c. Specify a group name, if you have specified =MEMBER as the USER value.

This approach ensures that, for any START command, there is always a matching profile with an STDATA segment that assigns a user ID. In addition, using this approach avoids the following situations, which cause RACF to use ICHRIN03 to process the START command:

- a. There is no matching profile.
  - b. There is a matching profile, but it does not have an STDATA segment.
  - c. There is a matching profile with an STDATA segment, but no user ID is specified.
  - d. There is a matching profile with an STDATA segment, no user ID is specified, but the assigned user ID matches an existing user ID on your system.
7. When RACROUTE REQUEST=VERIFY or VERIFYX is issued with a started procedure name, RACF checks to see if the STARTED class is active. If it is active, RACF uses the STARTED class to determine the user ID, group name, trusted flag, and privileged flag to use. If the STARTED class is not active, RACF uses the started procedures table (ICHRIN03). RACF also uses the started procedures table, and issues message IRR813I or IRR814I if the STARTED class is active but one of the following occurs:
  - a. RACF cannot find a matching profile in the STARTED class.
  - b. RACF finds a matching profile but the profile does not assign a user ID.

**Note:** Once you alter the TRUSTED attribute for the STARTED class, you must recycle the started task to pick up the change. This is because the TRUSTED attribute is part of the ACEE which is built when the address space starts up, so you have to recycle the task.



---

## Chapter 7. Protecting data sets on DASD and tape

This topic contains in-depth information on protecting data sets on DASD and tape.

---

### Protecting data sets

This topic describes considerations related to using RACF to protect data sets on DASD and tape. Unless there is an explicit qualification, all of the information in this topic applies to both DASD and tape.

To protect data sets, create data set profiles. These profiles can be either discrete or generic. See [z/OS Security Server RACF Command Language Reference](#) for information about how to use the ADDSD and PERMIT commands to create data set profiles.

Automatic direction of application updates for the DATASET class require special consideration. See [“Controlling automatic direction of passwords” on page 439](#) and [“Considerations for the DATASET class” on page 431](#).

### Rules for defining data set profiles

When you define data set profiles to RACF, you can use either standard or nonstandard naming conventions. If you use nonstandard naming conventions, the data set naming convention table and the single-level data set names option are ways to help “fit” RACF standard naming conventions.

The descriptions of naming conventions are followed by rules for protecting and allocating user and group data sets.

#### Standard data set naming conventions

By default, RACF expects a data set name (and the data set profile name) to consist of at least two qualifiers. RACF also expects the high-level qualifier of the data set profile name to be either a RACF-defined user ID or a RACF-defined group name.

If you and your implementation team have chosen to define data set profiles under the standard RACF naming conventions, you can create a group for each high-level qualifier that is not a user ID, and permit users to protect any data set that has that high-level qualifier by giving them CREATE authority in that group.

RACF can help enforce standard naming conventions at your location in several ways. These ways require users to use your predefined naming convention so that their data sets are RACF-protected.

- RACF has a PROTECTALL option on the SETROPTS command that allows a user to create or access a data set only if the data set is RACF-protected, by either a discrete or generic profile. See [“RACF-protecting all data sets \(PROTECTALL option\)” on page 119](#) for more information.
- If your installation does not use PROTECTALL, use a RACROUTE REQUEST=DEFINE exit routine to ensure that a predefined generic profile exists before allowing a user to create a data set.
- When your users have the ADSP attribute, they can create or protect only data sets whose names begin with their own user ID, or for which they have CREATE or higher authority in the RACF group corresponding to the high-level qualifier of the data set name.

#### Table-driven data set naming conventions

You can use the naming convention table to set up and enforce a data set naming convention other than that used by RACF. The table can:

- Supply a qualifier to be used as the high-level qualifier for authorization checking
- Convert data set names to RACF naming convention form for RACF use
- Convert names in RACF form to the installation's format for external display

- Enforce a naming convention by not allowing the definition of data sets that do not conform to an installation's rules
- Reduce RACF overhead by determining whether a data set is a user or group data set.

You can create a naming convention table (module ICHNCV00), which RACF uses to check and modify (internally to RACF) the data set name in all commands and macros that process data set names. You can use the table to selectively rearrange data set names to *fit* the RACF convention without actually changing those names.

Naming convention processing is done by RACF immediately before the preprocessing/naming convention installation exits are called. (The exits can still be used for additional processing.)

The RACF table-driven naming convention feature largely replaces the need for the ICHCNX00 exit routine. (The naming convention table is processed before each call to ICHCNX00.)

For more information on creating and using the RACF naming convention table, see [z/OS Security Server RACF System Programmer's Guide](#).

## Protecting data sets that have single-qualifier data set names

If some of the data sets in your installation have names that consist of a single qualifier, you can still RACF-protect those data sets. To get RACF protection for single-qualifier names, issue the SETROPTS command with the PREFIX operand. This command defines a high-level qualifier to be used as a prefix for single-qualifier names and activates the facility. Then, when RACF processes requests for the data set, RACF internally modifies single-qualifier names by adding the prefix, making the data set names acceptable to RACF routines.

In subsequent references to the profile, all RACF commands and the RACF report writer expect to see the prefix followed by a period and the single-level data set name. All SMF log records and all messages from RACF contain the RACF-modified version of the data set name.

**Important:** If you do not issue the SETROPTS command with the PREFIX operand, a system ABEND occurs when a user attempts to create a data set with a single-qualifier name. This abend occurs only when creating a discrete profile as part of data set allocation.

**Note:** The real data set names option (specified by the REALDSN operand on the SETROPTS command) applies only to name conversions made by the naming conventions table or installation exit routines. This option has no effect on single-qualifier data set names (unless they have been modified by the naming conventions table or an exit routine), whose "real data set names" continue to be the prefixed ones.

For more information on specifying the prefix, see [“Protecting data sets with single-qualifier names \(PREFIX option\)”](#) on page 122.

## Protecting user data sets

A *user data set* is a data set whose high-level qualifier is a RACF user ID. The following rules apply to user data sets:

- In general, all RACF-defined users can protect their own data sets. However, some SETROPTS options can restrict the ability of users to define and change profiles. See [“Restricting changes to security labels \(SECLABELCONTROL option\)”](#) on page 133.
- A user can RACF-protect a data set for another user under any of the following conditions:
  - The user who is protecting the data set has the SPECIAL attribute. A discrete or generic profile can be created.
  - The user who is protecting the data set has the group-SPECIAL attribute, and the high-level qualifier of the data set name is a user within the group-SPECIAL user's scope of authority. A discrete or generic profile can be created.
  - The user who is protecting a data set has the OPERATIONS attribute (or the group-OPERATIONS attribute if the data set is within his scope of authority) *and* is simultaneously creating the data set.

In this case, the user can create a discrete profile:

- Through ADSP
- By specifying the PROTECT operand on the TSO ALLOCATE command that creates the data set
- By specifying the PROTECT=YES OR SECMODEL=*profile-name* operands on the JCL DD statement that creates the data set
- The REQUEST=DEFINE preprocessing exit routine allows RACF protection.

## Protecting group data sets

A *group data set* is a data set whose high-level qualifier is a RACF group name. A RACF-defined user can RACF-protect a group data set under any of the following conditions:

- The user has JOIN, CONNECT, or CREATE authority in the group.
- The user has the SPECIAL attribute (or the group-SPECIAL attribute for that group) and the request is made using the ADDSD command.
- The user has the OPERATIONS attribute and is not connected to the group.
- The REQUEST=DEFINE preprocessing exit routine is used to override normal RACF authorization requirements.

## Controlling the creation of new data sets

Using data set profiles, you can control whether users can create (allocate) new data sets.

For cataloged data sets, creating, deleting, or renaming the data set involves access not only to the data set profile protecting the data set, but also to the catalog in which the data set is cataloged. In general, users need the following:

- To add entries to the catalog, users need authority to create the data set according to the following specifications and UPDATE authority to the catalog.
- To delete entries from the catalog, users need ALTER authority to the protecting profile or to the catalog.

For more information, see [“Protecting catalogs” on page 166](#) and [z/OS DFSMS Managing Catalogs](#).

The following cases describe how RACF can be used to control the creation of new user and group data sets.

A user can create a new user data set in the following situations:

- The data set is protected by an existing generic profile and the user does not have ADSP.  
The creation is allowed if (1) the user has ALTER authority to the data set through the generic profile or global access checking, or (2) the data set is the user's own data set. RACF does not create a profile.
- The data set name is not covered by an existing generic profile and the user does not have ADSP.  
If PROTECTALL is not in effect, the creation is allowed, but RACF does not create a profile. See Note 2.
- The user has ADSP and the data set is the user's own data set.  
The creation is allowed and RACF creates a discrete profile for the data set.
- The REQUEST=DEFINE preprocessing exit routine allows RACF protection.
- The user has the OPERATIONS attribute. If the user has the group-OPERATIONS attribute (that is, the user is connected to a group with the OPERATIONS attribute), the high-level qualifier of the new data set must be the ID of a user who is within the scope of that group.

A user can create a new group data set in the following situations:

- The data set name is protected by an existing generic profile and the user does not have ADSP.  
The creation is allowed if at least one of the following is true:
  - The user has ALTER authority to the data set through the generic profile or global access checking.
  - The user has CREATE authority in the group.

RACF does not create a profile.

- The data set name is not covered by an existing generic profile and the user does not have ADSP.

If PROTECTALL is not in effect, the creation is allowed, but RACF does not create a profile. See Note 2.

- The user has ADSP and the data set belongs to a group of which the user is a member.

The creation is allowed only if the user has CREATE authority in the group. If the creation is allowed, RACF creates a discrete profile for the data set.

- The REQUEST=DEFINE preprocessing exit routine allows RACF protection.
- The user has the OPERATIONS attribute except when both of the following are true:
  1. The user is connected to the group with less than CREATE authority.
  2. The user has less than ALTER access to the data set if it protected by a generic profile.

If the user has the group-OPERATIONS attribute (that is, the user is connected to a superior group with the OPERATIONS attribute), the group for which the new data set is being created must be within the scope of that superior group.

If PROTECTALL is not in effect, any user without ADSP can create a data set whose high-level qualifier is neither a RACF user ID (user data set) nor a RACF group name (group data set), but the data set cannot be RACF-protected. Note that a dummy group (a group that has no users connected to it) can be defined for the high-level qualifier of these data sets so that they can then be RACF-protected.

**Note:**

1. In all cases, if the user specifies the PROTECT=YES or SECMODEL parameter on the JCL DD statement, or the PROTECT or SECMODEL operand on the TSO ALLOCATE command (these operands request that RACF create a discrete profile), RACF treats the user the same as a user with ADSP. However, because the use of these operands is voluntary, an installation cannot use the operands to control the creation of data sets.
2. If PROTECTALL is in effect at your installation, a user cannot create a new data set unless the data set is RACF-protected by either a discrete or generic profile. However, instead of rejecting all creation requests for unprotected data sets, PROTECTALL also allows installations to issue warning messages. For more information on the PROTECTALL option, see [“RACF-protecting all data sets \(PROTECTALL option\)” on page 119](#).

## Data set profile ownership

Each data set profile defined to RACF requires a RACF-defined user or group as the owner of the profile. The owner (if a user) has full control over the profile, including the access list.

If the owner of the data set profile is a group, users with group-SPECIAL in that group have full control over the profile.

Ownership of data set profiles is assigned when the profiles are defined to RACF. Note that ownership of a data set profile does not mean that the owner can automatically access that data set. To access a data set, the owner must still be authorized in the profile's access list, unless the high-level qualifier of the profile name is the owner's user ID.

In some cases, the OWNER field of a discrete data set profile can be changed simply by renaming the data set. For details, see [z/OS Security Server RACF System Programmer's Guide](#).

## Data set profiles

The following topics describe what occurs when data sets are protected by profiles.

### Protection through discrete profiles

Users can protect data sets with discrete profiles in the following ways:

- Automatically when they create a permanent data set, if they have the ADSP attribute and ADSP is active on the system
- When they specify the PROTECT or SECMODEL parameter on a JCL DD statement for a new data set, or the PROTECT or SECMODEL operand on the TSO ALLOCATE command for a new permanent DASD data set
- When they issue the ADDSD command with the SET operand for permanent existing data sets

Two steps occur when a user defines a data set with a discrete profile. Only when RACF has completed both of the following steps is the data set protected:

1. RACF sets an indicator to notify the system that the data set is RACF-protected. This condition is called *RACF-indicated*.

The indicator is in the DSCB for a non-VSAM DASD data set and in the catalog entry for a VSAM data set. The indicator for a tape data set is in the tape volume profile for the volume that contains the data set.

**Note:** See *z/OS Security Server RACF System Programmer's Guide* for information on moving RACF-indicated data sets to other systems and using utilities with RACF-protected data sets.

2. RACF adds the discrete profile to the RACF database.

For tape data sets, RACF also creates a discrete tape volume profile, unless a tape volume profile already exists for the volume or the TAPEVOL class is not active.

**Note:**

1. Scratching a DASD data set that is RACF-protected with a discrete profile causes RACF to delete the data set profile from the RACF database.
2. Specifying DISP=DELETE for a tape data set only causes the data set to be uncataloged if it was cataloged; it does not remove RACF protection from the data set.

## Protection through generic profiles

By using generic profiles, your installation can reduce both the number of profiles required to protect data sets and the size of the RACF database, thus making RACF protection easier to administer. In addition, generic profiles are loaded into storage when first needed, are not deleted when the data set they protect is deleted, and are not volume-specific (that is, data sets protected by a generic profile can reside on any volume).

You can define a generic profile to protect data sets in one of the following ways:

- By issuing the ADDSD command and specifying the generic characters \*, %, or, if enhanced generic naming is active, \*\* in the profile name. Profile names that contain generic characters can protect a number of similarly named data sets.
- By issuing the ADDSD command and specifying the GENERIC operand. Use this operand when the profile name you specify does not contain any generic characters, in which case it is a fully qualified generic profile. A fully qualified generic profile protects only those data sets whose name matches the profile name exactly. For example, you might define a fully qualified generic profile to protect data sets with the same name that reside on different volumes.

## Rules for generic data set profile names

The following topics describe the rules for creating, activating, and modifying generic profile names used for data sets.

### When you can specify generic profile names

You can create a profile with a generic name when either of the following is true for the class of the profile:

- The SETROPTS GENERIC(DATASET) option is in effect. Not only does this option allow the creation of generic profiles, it also causes RACF to use generic profiles during authorization checking.

- The SETROPTS GENCMD(DATASET) option is in effect. In this case, generic profiles can be created and modified, but RACF does not use them during authorization checking. This is intended for use when migrating from discrete profiles to generic profiles.

Some of the rules for generic characters are different between general resource and data set generic profiles. For more information, see [“Rules for generic profile names” on page 188](#) and [z/OS Security Server RACF Command Language Reference](#).

The following rules apply to generic data set profile names:

- Valid generic characters are \*, %, and \*\*:
  - Specify % in the profile name to match any single non-blank character (except a period) in the same position of the resource name.
  - Specify \* or \*\* in the profile name to match more than one character in the same position of the resource name. For data set profile names, you can specify \*\* only if the SETROPTS EGN option is in effect. For a complete description, with examples, of how to specify \* and \*\*, see [z/OS Security Server RACF Command Language Reference](#).
- For profiles in the DATASET class, the high-level qualifier of the profile name can neither contain nor be a generic character. Here are some examples:

**ABC.EF\***

Valid

**ABC.EF.\*\***

Valid

**A%C.EFG**

Invalid

**\*.EFG**

Invalid

**ABC\*.XYZ**

Invalid

**\*\* .XYZ**

Invalid

**Note:** You might see data set names with the high-level qualifiers of &&TEMP and \*\*SYSUT. These data sets are created internally by the IEHMOVE program and should not be used for any other reason.

RACF enforces the rule that data set qualifiers can be no longer than eight characters. Therefore, in generic data set profiles, the generic characters \* and \*\* cannot be used to match qualifiers that are longer than eight characters.

## When to do a generic refresh

After you define or change generic profiles, activate your changes by entering:

```
SETROPTS GENERIC(classname) REFRESH
```

## Choosing between discrete and generic data set profiles

When you create a profile in the DATASET class, you can create either a discrete or generic profile.

Choose a *generic* profile for the following reasons:

- If you want to protect more than one data set with the same security requirements. The data sets protected by a generic profile must have some identical characters in their names. The profile name contains one or more generic characters (\*, \*\*, or %).
- If you have a single data set that might be deleted, then recreated, and you want the protection to remain the same, you can create a fully qualified generic profile. The name of a fully qualified generic profile matches the name of the data set it protects. Unlike a discrete profile, a fully qualified generic profile is not deleted when the data set itself is deleted.



Choose a *discrete* profile for the following reasons:

- To protect one data set that has unique security requirements. The name of a discrete profile matches the name of the data set it protects.
- To allow changes to a data set profile to take effect immediately, without needing to refresh in-storage copies of the profile.

**Note:**

1. All of the members of a partitioned data set are protected by one profile, the profile that protects the data set.
2. All of the components of a VSAM data set are protected by one profile, the profile that protects the cluster name. You do not need to create profiles that protect the index and data components of a cluster.
3. For a generic profile, unit and volume information is ignored because the data sets that are protected under the generic profile can reside on many different volumes.

## Generic profile checking for the DATASET class

The rules for access-authorization checking of generic profiles for data sets are as follows:

- Generic profiles are not checked unless generic profile checking is active for the DATASET class. To activate it, enter:

```
SETROPTS GENERIC(DATASET)
```

**Guideline:** Once you activate generic profile checking for the DATASET class and define generic data set profiles, avoid deactivating it with the NOGENERIC operand. RACF will not use your previously defined generic profiles for authorization checking while NOGENERIC is in effect.

- If generic profile checking is in effect for the DATASET class, RACF examines the profiles as follows:
  - For a discrete profile (if the caller indicates that the data set is RACF-indicated).
  - For a fully qualified generic profile.
  - For other generic profiles in the order of *most specific to least specific* profile name. See [Table 9 on page 153](#) and [Table 10 on page 154](#).
- After a profile is found, RACF uses information in the profile to do authorization checking. For a complete description, see [“Authorization checking for RACF-protected resources” on page 718](#).

If the data set is RACF-indicated, RACF first checks for a discrete profile. If a discrete profile does not exist, RACF examines the generic profiles in the order of *most specific to least specific* profile name. Therefore, if a discrete profile does not exist, RACF uses the most specific matching generic profile.

If the data set is *not* RACF-indicated, RACF examines the generic profiles in the order of *most specific to least specific*, and uses the most specific matching generic profile.

**Note:** To determine which generic profile is the most specific match to a particular data set name, you can use the LISTDSD command with the GENERIC option.

[Table 9 on page 153](#) and [Table 10 on page 154](#) list some generic profiles from the DATASET class. This figure represents the order in which RACF checks the generic profiles when it performs access-authorization checking. (This order is also the order that RACF commands such as SEARCH would list these generic profiles.)

Table 9. Sample data set profile names in order from most specific to least specific (EGN off)

Profile name	Profile type	Data sets being accessed		
		SALES.DATA	SALES.DATA.TEST	SALES.YEARLY.QUOTA
SALES.A	Fully qualified generic			
SALES.DATA	Discrete	X		

## data sets

Table 9. Sample data set profile names in order from most specific to least specific (EGN off) (continued)

Profile name	Profile type	Data sets being accessed		
		SALES.DATA	SALES.DATA.TEST	SALES.YEARLY.QUOTA
SALES.DATA	Fully qualified generic	X		
SALES.DATA.%	Generic			
SALES.DATA.*	Generic		X	
SALES.DATA%	Generic			
SALES.DATA*	Generic	X	X	
SALES.DAT%	Generic	X		
SALES.DAT*	Generic	X	X	
SALES.DISK.*	Generic			
SALES.YEARLY.QUOTA	Discrete			X
SALES.YEARLY.QUOTA	Fully qualified generic			X
SALES.YEARLY.*	Generic			X
SALES.%ATA	Generic	X		
SALES.*.QUOTA	Generic			X
SALES.*.QUOTA*	Generic			X
SALES.*	Generic	X	X	X

**Note:** RACF ignores a discrete profile if a data set is not RACF-indicated. Any data set that has a discrete profile must be RACF-indicated.

Table 10. Sample data set profile names in order from most specific to least specific (EGN on)

Profile name	Profile type	Data sets being accessed		
		SALES.DATA	SALES.DATA.TEST	SALES.YEARLY.QUOTA
SALES.A	Fully qualified generic			
SALES.DATA	Discrete	X		
SALES.DATA	Fully qualified generic	X		
SALES.DATA.%	Generic			
SALES.DATA.*	Generic		X	
SALES.DATA.**	Generic	X	X	
SALES.DATA%	Generic			
SALES.DATA*	Generic	X		
SALES.DAT%	Generic	X		
SALES.DAT*	Generic	X		
SALES.DISK.*	Generic			
SALES.YEARLY.QUOTA	Discrete			X
SALES.YEARLY.QUOTA	Fully qualified generic			X
SALES.YEARLY.*	Generic			X
SALES.%ATA	Generic	X		
SALES.*	Generic	X		

Table 10. Sample data set profile names in order from most specific to least specific (EGN on) (continued)

Profile name	Profile type	Data sets being accessed		
		SALES.DATA	SALES.DATA.TEST	SALES.YEARLY.QUOTA
SALES.*.QUOTA	Generic			X
SALES.*.QUOTA*	Generic			X
SALES.**.DATA	Generic	X		
SALES.**.QUOTA	Generic			X
SALES.**	Generic	X	X	X

**Note:** RACF ignores a discrete profile if a data set is not RACF-indicated. Any data set that has a discrete profile must be RACF-indicated.

To find out which profiles could protect a data set, perform the following steps:

1. Find out if there is a discrete profile protecting the data set:

```
LISTDSD DATASET('data-set-name')
```

If a discrete profile exists for the data set, this command lists the contents of the profile.

2. If no discrete profile protects the data set, issue the LISTDSD command again with the GENERIC option:

```
LISTDSD DATASET('data-set-name') GENERIC
```

If a generic profile exists for the data set, this command lists the contents of the profile.

3. There might well be other generic profiles that have the potential to protect the data set. These profiles are listed by the SEARCH command in the order that RACF would use them. Because all data set profiles begin with a user ID or group name, you can use the FILTER operand to show only those profiles that could protect a data set, as shown in the following examples:

```
SEARCH CLASS(DATASET) FILTER(userid.**)  
SEARCH CLASS(DATASET) FILTER(groupname.**)
```

To see which of two generic profiles is more specific, compare the profile names, character by character. Where they first differ, if one has a discrete character and the other has a generic character, the one with the discrete character wins. If both have a generic character where they differ, then:

- If one has a % and the other has a \* or \*\*, the one with % wins.
- If one has a \* and the other has a \*\*, the one with \* wins.

If two profile names fit except for one character position, the following is the order in which RACF examines them:

```
blank  
.  
$ (X'5B')  
# (X'7B')  
@ (X'7C')  
A-Z  
0-9  
%  
*
```

**Tip:** The characters \$, #, and @ might be displayed differently on terminals outside the United States. Therefore, use the characters with the hexadecimal equivalents shown.

For example, the following profile names all fit in the first three character positions (A.B), and are shown in the order RACF examines them:

```
A.B  
A.B.B  
A.BA
```

```
A.BZ
A.B0
A.B9
A.B%
A.B*
```

When in doubt about the search order, create sample profiles and check the order of profile names shown by the SEARCH command.

## Generic profile performance

Your system programmer can collect and analyze performance information related to generic profiles and then specify, or ask you to specify, certain RACF options to customize how RACF processes generic profiles. For details, see [Using generic profiles in z/OS Security Server RACF System Programmer's Guide](#).

## Using SETROPTS PROTECTALL and SETROPTS GENERIC(DATASET) together

If PROTECTALL is in effect at your installation, generic profile checking should also be in effect. This allows you to create or access a data set if one of the following conditions is met:

- The data set is protected by a discrete profile.
- The data set is protected by a generic profile.
- The access is allowed by global access checking.

For users with alter authority, RACF allows renaming a data set from a name covered by a global entry to another name covered by a global entry. Similarly, renaming is allowed from a name covered by one generic profile to a name covered by another generic profile. Renaming is not allowed from a name covered by a generic profile to one covered by a global entry, because this could allow the user to remove protection from the data set.

If PROTECTALL is in effect and generic profile checking is not, only users who have ADSP or specify PROTECT=YES can create new data sets.

After defining, altering, or deleting a generic profile, the following command ensures that the profile is in effect during authorization checking:

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

RACF is invoked whenever a data set is accessed (whether or not the data set is RACF-indicated) and whenever DASD space is allocated for a data set (whether or not the user has the ADSP attribute or has specified PROTECT=YES on the JCL statement). When RACF is invoked for a data set that is not RACF-indicated, RACF checks only predefined generic profiles and the global access checking table. If PROTECTALL is not in effect and RACF cannot find an appropriate generic profile or a matching entry in the global access checking table, RACF accepts the access request by default.

**Important:** Data sets that are not RACF-indicated but are protected by a generic profile are *not* protected if they are transferred (in any way) or available (such as through shared DASD) to another system that does not have RACF and appropriate predefined generic profiles.

## Authority to modify generic profiles

To modify a generic profile, a user must be the profile owner, or have the SPECIAL (or group-SPECIAL, if applicable) attribute, or have a user ID identical to the profile's high-level qualifier. Unless one of these conditions is met, the user cannot alter the generic profile, even if the user has ALTER access authority to the profile. Note that the access list in a generic profile does not apply to the profile itself. See [z/OS Security Server RACF Command Language Reference](#) for descriptions of the authorities needed to issue particular RACF commands.

## Conditional access lists for data set profiles

RACF allows installations to specify conditional access lists for data sets. You can require that a user or job enter the system from a particular device when accessing data sets. To do this, specify one or more device identifiers using one of the following methods.

- By specifying WHEN(TERMINAL(...)) on the PERMIT command, you can require that a user be logged on to a particular terminal.

For this support to take effect, the TERMINAL class must be active.

- By specifying WHEN(CONSOLE(...)) on the PERMIT command, you can require that a user be logged on to a particular console.

For this support to take effect, the CONSOLE class must be active.

- By specifying WHEN(JESINPUT(...)) on the PERMIT command, you can require that the batch job accessing the data set has been submitted from a particular JES input device.

For this support to take effect, the JESINPUT class must be active.

- By specifying WHEN(APPCPORT(...)) on the PERMIT command, you can require that a user enter the system from a particular partner LU.

For this support to take effect, the APPCPORT class must be active.

- By specifying WHEN(SERVAUTH(...)) on the PERMIT command, you can require that a user enter the system from a particular network security zone (containing IP addresses).

For this support to take effect, the SERVAUTH class must be active.

**Note:** If an access list contains more than one condition, *any* of the conditions allows the specified access. For example, if you enter the PERMIT command with WHEN(CONSOLE(01) TERMINAL(20)) specified, you allow the access when *either* console 01 *or* terminal 20 is used.

## Universal access authority (UACC) for data sets

Each data set profile you define with RACF requires a universal access authority (UACC). The UACC is the default access authority that RACF gives to users and groups that are not defined in the profile's access list. If one of these users or groups requests access to a data set that is protected by the profile, RACF grants or denies the request based on the UACC. UACC coverage also extends to users that are not defined to RACF and batch jobs that are not associated with a RACF-defined user. A batch job has no user ID associated with it in the following cases:

- There is no user ID propagation in the system, and no user ID or password was specified.
- The release of JES that is installed does not support user ID propagation.
- The job originated from an NJE, RJE, or card reader, and no USER parameter was specified on the JOB statement.

In some cases, jobs originating from NJE can have a user ID, depending on the NODES class profiles that are defined on your system.

If you specifically assign an access authority to a user or group, the authority you specify overrides the UACC assigned to the data set. Also, if the access checking defined in the global access checking table is higher than the UACC assigned to the data set, the entry in the global access checking table overrides the UACC.

For a given data set:

- If you set UACC to NONE, all users are refused access to the data set because they are not authorized to access the data set through an access list, global access checking, the OPERATIONS attribute, or the WARNING indicator.
- If you set UACC to READ, EXECUTE, UPDATE, CONTROL, or ALTER, all users can access the data set at the specified level of authority, unless they are specifically excluded by security classification checking or an entry in the standard access list, or the user ID has the RESTRICTED attribute.

**Note:** If you have users who are not defined to RACF, you can use ID(\*) instead of UACC to ensure that only RACF-defined users access the resource. The following examples illustrate the difference between UACC(READ) and ID(\*) ACCESS(READ).

- To allow *all* users on the system to use a data set, specify UACC(READ) for the profile, as follows:

```
RDEFINE profile-name UACC(READ)
```

- To allow *only RACF-defined* users on the system to use a data set, specify UACC(NONE) for the profile, and then issue the PERMIT command with ID(\*) and ACCESS(READ) specified:

```
RDEFINE profile-name UACC(NONE)
PERMIT profile-name ID(*) ACCESS(READ)
```

## Automatic profile modeling for data sets

You can set up automatic modeling for new data set profiles (whether discrete or generic). You can do this for:

- Data set profiles for selected users
- Data set profiles for selected groups
- GDG data sets

## Automatic profile modeling for user data set profiles

You can specify a model data set profile to be used whenever new user data set profiles are created for a specific user. Information from the model profile is copied to any data set profile with the specified user ID as high-level qualifier.

To do this, follow these steps:

1. For each user for which modeling is to be done, specify the profile that is to be used as a model:

```
ALTUSER userid MODEL(model-profile-name)
or
ADDUSER userid MODEL(model-profile-name)
```

**Note:** When specifying the MODEL operand, do *not* specify the user's user ID on the model profile name.

2. If necessary, create a model data set profile:

```
ADDSD 'userid.model-profile-name' MODEL
...other appropriate operands such as UACC and AUDIT...

PERMIT 'userid.model-profile-name' ID(appropriate-users-or-groups)
ACCESS(access-authority)
```

**Note:**

- a. With the MODEL operand specified, no actual data set need exist with the specified profile name. A generic profile cannot be a model profile.
  - b. A profile created with the MODEL operand is not intended to actually protect a data set (and does not cause an existing data set to be RACF-indicated). However, if a data set with the same name exists, the model profile might be used to protect that data set. Therefore, choose a profile name such that the profile does not match any data sets.
3. When you are ready to start using model profiles for user data sets, issue the SETROPTS command with MODEL(USER) specified:

```
SETROPTS MODEL(USER)
```

4. After the SETROPTS command has been issued, if a user creates a user data set profile for another user, and that profile had the MODEL operand specified, information from the model profile is *always* copied into the new user data set profile.

**Example:**

The following commands set up a model profile named SUE.SAMPMOD for user SUE. The model specifies a UACC of NONE and gives READ access to SAM, JOE, and GROUP1.

```
(1) ALTUSER SUE MODEL(SAMPMOD)
(2) ADDSD 'SUE.SAMPMOD' MODEL UACC(NONE)
(3) PERMIT 'SUE.SAMPMOD' ID(SAM JOE GROUP1) ACCESS(READ)
(4) SETROPTS MODEL(USER)
```

User SUE then issues the following command.

```
(5) ADDSD 'SUE.DATA' UACC(READ)
```

In this example:

- (1) indicates to RACF that automatic profile modeling is to be used for new profiles beginning with SUE.
- (2) creates a profile named SUE.SAMPMOD. With the MODEL operand specified, no actual data set named SUE.SAMPMOD needs to exist. However, if a data set named SUE.SAMPMOD does exist, it is protected by the profile named SUE.SAMPMOD.
- (3) specifies an access list for profile SUE.SAMPMOD.
- (4) turns on automatic profile modeling for all of the users who have the MODEL operand set in their user profiles.
- (5) creates profile SUE.DATA with UACC(READ). RACF copies the access list from SUE.SAMPMOD (SAM, JOE, and GROUP1 have READ access). With UACC(READ) specified on the ADDSD command, the UACC(NONE) value from SUE.SAMPMOD is not used. Note that the copied information can be changed during the copy. See [“Possible changes to copied profiles when modeling occurs”](#) on page 37.

## Automatic profile modeling for group data set profiles

You can specify a model data set profile to be used whenever new group data set profiles are created in a specific group. Information from the model profile is copied to any data set profile with the specified group name as high-level qualifier.

To do this, perform the following steps:

1. For each group for which modeling is to be done, specify the profile to be used as a model by entering one of the following commands.

```
ALTGROUP groupname MODEL(model-profile-name)
ADDGROUP groupname MODEL(model-profile-name)
```

**Note:** When specifying the MODEL operand, do *not* specify the group's group name on the model profile name.

2. If necessary, create a model data set profile:

```
ADDSD 'groupname.model-profile-name' MODEL
...other appropriate operands such as UACC and AUDIT...

PERMIT 'groupname.model-profile-name' ID(appropriate-users-or-groups)
ACCESS(access-authority)
```

**Note:**

- a. With the MODEL operand specified, no actual data set need exist with the specified profile name.
- b. A profile created with the MODEL operand is not intended to actually protect a data set (and does not cause an existing data set to be RACF-indicated). However, if a data set with the same name exists, the model profile might be used to protect that data set. Therefore, choose a profile name such that the profile does not protect any data sets.

3. When you are ready to start using model profiles for group data sets, issue the SETROPTS command with MODEL(GROUP) specified:

```
SETROPTS MODEL(GROUP)
```

4. After the SETROPTS command has been issued, if a user creates a group data set profile for a group for which the MODEL operand has been specified, information from the model profile is *always* copied into the new group data set profile.

**Example:**

The following commands set up a model profile named GROUP1.SAMPMOD for group GROUP1. The model specifies a UACC of NONE and gives READ access to SAM, JOE, and GROUP1.

```
(1) ALTGROUP GROUP1 MODEL(SAMPMOD)
(2) ADDSD 'GROUP1.SAMPMOD' MODEL UACC(NONE)
(3) PERMIT 'GROUP1.SAMPMOD' ID(SAM JOE GROUP1) ACCESS(READ)
(4) SETROPTS MODEL(GROUP)
```

A user then issues the following command.

```
(5) ADDSD 'GROUP1.DATA' UACC(READ)
```

In this example:

- (1) indicates to RACF that automatic profile modeling is to be used for new profiles beginning with GROUP1.
- (2) creates a profile named GROUP1.SAMPMOD. With the MODEL operand specified, no actual data set named GROUP1.SAMPMOD needs to exist. However, if a data set named GROUP1.SAMPMOD does exist, it is protected by the profile named GROUP1.SAMPMOD.
- (3) specifies an access list for profile GROUP1.SAMPMOD.
- (4) turns on automatic profile modeling for all groups that have the MODEL operand set in their group profiles.
- (5) creates profile GROUP1.DATA with UACC(READ). RACF copies the access list from GROUP1.SAMPMOD (SAM, JOE, and GROUP1 have READ access). With UACC(READ) specified on the ADDSD command, the UACC(NONE) from GROUP1.SAMPMOD is not used. Note that the copied information can be changed during the copy. See [“Possible changes to copied profiles when modeling occurs” on page 37](#).

## Automatic profile modeling for GDG data sets

You can use automatic profile modeling for GDG data sets. For more information, see [“Protecting GDG data sets” on page 161](#).

## Password-protected data sets

When a data set is both password-protected and RACF-protected, access to the data set is authorized through RACF authorization checking. If an authorization request for a password-protected data set is satisfied by a RACF global access table entry or a RACF data set profile, password checking is ignored.

When a data set is password-protected but not RACF-protected, access to the data set is authorized through password protection. When a RACF-protected data set is moved to a system without RACF support, you cannot perform authorization checking. Therefore, after you have installed RACF, your users might need to maintain password protection only for those data sets that:

- Are not RACF-protected
- Are RACF-protected and are used on other systems that do not have RACF support

Password protection is not used for SMS-managed data sets. Therefore, if your installation has procedures that use password protection for data sets, you must modify these procedures accordingly.



## Protecting GDG data sets

You can RACF-protect GDG (generation data group) data sets in one of the following ways:

- You can define a generic profile to protect all members of a GDG. This is the preferred method and it is the same method for protecting non-GDG data sets with a generic profile. For example, a profile of the form `GDG.basename*` protects all members of a GDG and the base entry for the GDG in the catalog.

Note that, if enhanced generic naming is in effect, a profile of the form `GDG.basename.**` provides the same protection.

Table 11 on page 161 shows examples of generic profiles that you can define to protect GDG data sets.

Table 11. Protecting GDG data sets using generic profiles

Generic profile name	EGN	Protected GDG names
GDG.BASENAME*	Off	GDG.BASENAME GDG.BASENAME.G0123V00
GDG.BASENAME.**	On	GDG.BASENAME GDG.BASENAME.G0123V00

**Note:** For GDG profiles, with enhanced generic naming active, you can no longer define a profile name such as `GDG.ABCDEFGH*` whose last qualifier contains an asterisk as the ninth character. Externally, an existing profile name of this format is shown as `GDG.ABCDEFGH.**`. Internally, no conversion is required because the two names are equivalent. However, you should examine existing CLISTs that generate commands to ensure that any profile names that appear in those commands are in the correct format.

- You can define discrete profiles to protect GDG data sets in the same way that you define discrete profiles to protect non-GDG data sets.

**Note:** Catalog management also checks authority to the GDG base name. You should create a discrete profile for the GDG base with the unit and volume of the catalog on which the GDG base resides. This protects the GDG for catalog and uncatalog functions.

- You can use the `MODEL(GDG)` operand on the `SETROPTS` command to specify that each member of a GDG can use a common profile identified by the GDG base name. The owner of the GDG data set can establish a base (index) name profile containing an access list that is accessible by all related users and groups. When `MODEL(GDG)` is in effect and `REQUEST=AUTH` processes a RACF-indicated GDG data set, RACF first looks for a profile with the base name, and, if one exists, uses this common profile.

If you want individual access lists, do not create the profile for the base name. If the GDG base name is not defined in the RACF database, RACF uses the profile for the individual GDG name (which is the same as the RACF-processing for non-GDG data sets).

**Note:**

- To use GDG modeling, each generation must be RACF-indicated.
- Catalog management also checks authority to the GDG base name. You should create a discrete profile for the GDG base with the unit and volume of the catalog on which the GDG base resides. This protects the GDG for catalog and uncatalog functions.

## Protecting data sets that have duplicate names

You can use a fully qualified generic profile to protect data sets with the same name that reside on different volumes.

Alternatively, you can use separate, discrete profiles to define data sets having the same name. Support for data sets with duplicate names allows authorized users to:

- Move and copy RACF-protected data sets from one volume to another (for example, with the IEHMOVE system utility)
- Establish separate discrete profiles (including the access list and statistics and logging options) for data sets having the same name
- Protect data sets that have the same name and reside on non-shared volumes (such as SYS1.LINKLIB) on a loosely coupled system that uses a shared RACF data set

RACF differentiates between data sets having the same name by examining the volume serial number of each separately protected data set.

### Non-VSAM DASD data sets

For non-VSAM data sets, RACF uses the serial number of the volume on which the data set resides.

### VSAM data sets

For VSAM data sets, RACF uses the volume serial number of the catalog for the VSAM or integrated catalog facility in which the data set is cataloged. If multiple catalogs for the integrated catalog facility reside on the same volume and contain entries for duplicate VSAM data sets, only one of the data sets can be protected by a discrete profile.

### Tape data sets

If TAPEDSN is active and RACF is maintaining a TVTOC for the tape volume (TAPEVOL is active), RACF does not permit duplicate data set names on the same volume, or on different volumes if the volumes are part of the same multivolume data set group. This restriction applies even if the data sets are not protected by RACF.

## Disallowing duplicate names for data set profiles

You can prevent identically named data sets from being defined to RACF with separate, discrete profiles by modifying the installation-replaceable module ICHSECOP. For information on modifying ICHSECOP, see *z/OS Security Server RACF System Programmer's Guide*.

If you disallow duplicate data set profile names, data sets with the same name must be defined to RACF for protection in one common discrete profile with multiple volume serials. In this case, a data set shares the data set profile (including the access list, and statistics and logging options) with other data sets that have the same name.

Note that a fully qualified generic profile can also be used to protect data sets with identical names, regardless of which volumes they reside on.

## Using the PROTECT operand or SECMODEL for non-VSAM data sets

To create a discrete profile for a new tape or non-VSAM DASD data set (if you do not have the ADSP attribute), specify the PROTECT or SECMODEL parameter on the JCL DD statement that identifies the data set (or for DASD data sets, the PROTECT or SECMODEL operand on the TSO ALLOCATE command). Note that the normal reason for a user to use PROTECT or SECMODEL instead of ADSP is that most of the user's data sets do not require discrete profiles because they are covered by generic profiles.

## Protecting multivolume data sets with discrete profiles

You can protect a multivolume data set with either a discrete or a generic profile. If a generic profile protects the data set, the fact that the data set is multivolume is irrelevant.

To create a discrete profile for a multivolume tape or non-VSAM DASD data set, you must define each volume of the data set to RACF. RACF stores the volume serial numbers in the data set's profile. When the data set is extended to another volume or deleted from a volume, that volume's serial number is automatically added to or deleted from the data set profile.

**Note:** You cannot rename a multivolume non-VSAM DASD data set that is protected by a discrete profile. If the data set is protected with a generic profile, it can be renamed if the new name is also covered by a generic profile.

For more information on handling multivolume data sets in addition to the following considerations, see *z/OS Security Server RACF System Programmer's Guide*.

## Non-VSAM DASD data set considerations

When a multivolume physical sequential DASD data set is opened for input, RACF does not require that each volume on which the data set resides be defined in the data set profile.

When an existing multivolume physical sequential data set is opened for output, a RACF-protection consistency check is performed. All volumes of the data set that are processed by end-of-volume (when a volume switch occurs) must indicate the same RACF-protection status as the first volume opened. That is, if the first volume is RACF-protected (the DSCB indicator is on and the volume is defined in the data set profile), succeeding volumes must be RACF-protected as part of the same profile. If the first volume is not RACF-protected, succeeding volumes must not be RACF-protected.

For multivolume non-physical sequential DASD data sets, RACF performs authorization checking for each volume on which the data set resides.

## Tape data set considerations

When an existing multivolume data set is opened for output, a RACF consistency check is performed. All volumes of the data set must be RACF-protected, or all volumes of the data set must not be RACF-protected.

When a volume label is changed (destroyed) during a volume label rewrite, a REQUEST=DEFINE,TYPE=DELETE is issued for the old volume serial number.

## VSAM data set considerations

For VSAM data sets, extending the data set to a new volume causes RACF to protect the new volume, even though RACF does not add the serial number to the data set profile. The profile for VSAM data sets contains only the volume serial number of the catalog entry for the data set.

## Setting ADDCREATOR/NOADDCREATOR options for both DASD and tape

When specified in a generic profile, ALTER allows users to create new data sets that are covered by that profile. If the SETROPTS NOADDCREATOR option is in effect, the user who created the profile is not automatically added to the profile's access list. Even when the SETROPTS NOADDCREATOR option is in effect, when discrete DATASET or TAPEVOL profiles have been created using RACROUTE REQUEST=DEFINE (including RACDEF), the profile creator's ID is automatically added to the list. For more information, see [“Automatic addition of creator's user ID to access list” on page 139](#).

## Protecting DASD data sets

---

This topic gives additional information that applies to data sets on DASD. You should already be familiar with the information contained in [“Protecting data sets” on page 147](#).

## Access authorities for DASD data sets

You permit users and groups to access a RACF-protected data set by:

- Adding them to the access list of the discrete or generic profile that applies to the data set and
- Giving them one of the access authorities described in [Table 12 on page 164](#).

Table 12 on page 164 describes the access authorities associated with data set profiles. Many operations for cataloged data sets involve access not only to the data set profile protecting the data set, but also to the catalog in which the data set is cataloged. For access authorities required by users who are creating,

deleting, or renaming data sets, see [“Controlling the creation of new data sets”](#) on page 149. For more information about authorizing users to perform data set and catalog operations with protected catalogs, see the following documents:

- [z/OS DFSMS Access Method Services Commands](#)
- [z/OS DFSMS Using Data Sets](#)
- [z/OS DFSMS Managing Catalogs](#)

Table 12. Access authorities for DASD data sets

Authority	Access
NONE	Does not allow users to access the data set.
EXECUTE	For a private load library, allows users to load and execute, but not read or copy, programs (load modules) in the library.  <b>Note:</b> For more information about EXECUTE authority, see <a href="#">“Using EXECUTE access for programs and libraries in ENHANCED mode”</a> on page 316.
<b>Note:</b> Anyone who has READ, UPDATE, CONTROL, or ALTER authority to a protected data set can create a copy of it. As owner of the copied data set, that user has control of the security characteristics of the copied data set and can downgrade it. For this reason, you should assign a UACC of NONE, and then selectively permit a small number of users to access your data set, as their needs become known. (For information on how to permit selected users or groups to access a data set, see <a href="#">z/OS Security Server RACF Command Language Reference</a> .)	
READ	Allows users to access the data set for reading only. (Note that users who can read the data set can copy or print it.)
UPDATE	Allows users to read from, copy from, or write to the data set. However, UPDATE does not authorize a user to delete, rename, move, or scratch the data set.  Allows users to perform normal VSAM I/O (not improved control interval processing) to VSAM data sets.
CONTROL	For VSAM data sets, it allows users to perform improved control interval processing. This is control-interval access (access to individual VSAM data blocks), and the ability to retrieve, update, insert, or delete records in the specified data set.  For non-VSAM data sets, CONTROL is equivalent to UPDATE.
ALTER	Allows users to read, update, delete, rename, move, or scratch the data set.  When specified in a discrete profile, ALTER allows users to read, alter, and delete the profile itself <i>including the access list</i> . <sup>2</sup>  <b>Note:</b> ALTER does not allow users to change the owner of the profile using the ALTDSD command. However, if a user with ALTER access authority to a discrete data set profile renames the data set, changing the high-level qualifier to his or her own user ID, then both the data set and the profile are renamed, <i>and</i> the OWNER of the profile is changed to the new user ID.  When specified in a generic profile, ALTER gives users <i>no</i> authority over the profile itself, but allows users to create new data sets that are covered by that profile.

<sup>2</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

## Suggestions for assigning access authorities to DASD data sets

When protecting catalogs, be sure that users and groups have a sufficient level of access authority to each protected entity along the path to a data set that they are required to access.

The level of RACF access authority that a user or group requires to perform operations on VSAM data sets or catalogs is similar to the level of authorization required when passwords are used. For more information, see [“Comparison of password and RACF authorization requirements for VSAM” on page 165](#).

For a discussion of the levels of RACF access authority required to perform operations against catalogs, see [z/OS DFSMS Managing Catalogs](#).

## Erasing of scratched (deleted) DASD data sets

Installations can control the erasure of security-sensitive data set extents with the ERASE operand (erase-on-scratch) on the SETROPTS command. If a DASD data set profile has the erase indicator set, ERASE specifies that data management is to erase (overwrite) the contents of any scratched or released data set extents that are part of a DASD data set protected by that profile.

Users can set the erase indicator in both discrete and generic profiles by using the ADDSD and ALTDSD commands. Users can specify erasure for both single volume and multiple volume DASD data sets. However, to have the data set erased when scratched, the installation must also activate erase-on-scratch with the SETROPTS command.

The SETROPTS command has several options on the ERASE operand that allow an installation to override user specifications. These erase-on-scratch options allow an installation to:

- Specify that *all* DASD data set extents are always erased when scratched or released, regardless of the erase indicator in the profile. (This includes temporary data sets.) When this option is selected, installation exit routines *cannot* prevent any data set from being erased by overriding this option.
- Specify a security level (SECLEVEL) at which all data sets at this security level or higher are always erased when scratched or released, regardless of the erase indicator in the profile.
- Specify that only data sets that have the erase indicator in their profiles are erased when scratched or released.
- Specify that no data sets under RACF control are erased when scratched or released.

### Note:

1. RACF does not perform the actual erasure, but maintains an indicator for data management.
2. If you have not specified that you want *all* data sets erased, you can still provide for the erasing of sensitive temporary data sets by using the naming conventions table or RACF exit routines to conditionally convert the temporary data set names to the form of a permanent name that is covered by a profile that specifies erase-on-scratch.
3. In addition to the RACF-controlled erasure, any VSAM data set with a catalog entry that specifies erase is erased.

## Comparison of password and RACF authorization requirements for VSAM

The RACF authorization requirements are the same as the password requirements for most VSAM operations. The RACF authorization levels of ALTER, CONTROL, UPDATE, and READ correspond to the password levels of MASTER, CONTROL, UPDATE, and READ, respectively. As an example, deleting a VSAM data set requires the MASTER-level password of either the data set or the catalog that describes the data set. Deleting a RACF-protected VSAM data set requires ALTER authorization to the data set or the catalog. There are a few exceptions to the one-to-one correspondence of the RACF and password authorization levels. Access requirements for these exceptions can be found in the documents pertaining to the operations being performed.

## Protecting catalogs

You can protect many catalog functions including catalog locking and SMS-related functions, by creating profiles in the FACILITY class. For information on creating these profiles and authorizing users to perform catalog operations on protected catalogs, see the following documents:

- [z/OS DFSMS Managing Catalogs](#)
- [z/OS DFSMS Access Method Services Commands](#)
- [z/OS DFSMS Using Data Sets](#)

## Protecting DASD system data sets

When you are planning to RACF-protect system data sets, consider:

- The way in which the system uses the data set
- The way in which your users normally use the data set
- The level of protection you want for the data set

The system data sets can be divided into two categories: data sets for which RACF protection is bypassed when the system accesses them, and data sets for which RACF protection is enforced when the system accesses them.

### Bypassed RACF protection

For a data set of this type, RACF-protection is bypassed when the system accesses the data set to perform its normal system function, but is enforced when a user attempts to access the data set for normal data set operations.

For example, when the program libraries defined in LNKSTxx are opened during IPL, RACF protection is bypassed. A user can fetch any program stored in these libraries during IPL, but if the user attempts to open one of the libraries, RACF protection is enforced. Assuming SYS1.LINKLIB is defined in LNKSTxx, it can be RACF-protected giving UPDATE access authority to the system programmers who maintain the data set. You can use UACC(NONE) if you do not want anyone other than the system programmers to open the library. The UACC(NONE) specification does not prevent any user from executing any program contained in SYS1.LINKLIB, but it does prevent users from, for example, specifying SYS1.LINKLIB as part of JOBLIB, STEPLIB, or on the TSO CALL command.

Examples of other system data sets that fall into this category are:

```
SYS1.CMDLIB
SYS1.DUMPnn
SYS1.LOGREC
SYS1.LPALIB
SYS1.MANn
SYS1.NUCLEUS
SYS1.PARMLIB
SYS1.PROCLIB
SYS1.SVCLIB
```

System data sets that are frequently accessed by all users (for example, SYS1.HELP and SYS1.MACLIB) are good candidates for inclusion in a global access checking table.

### Enforced RACF protection

For a data set of this type, RACF protection is enforced when the system accesses the data set for its normal system function on behalf of a specific user. When you protect this type of data set, any user who requests the system function associated with the data set must have a sufficient level of access authority to the data set for the function to work correctly.

For example, when you RACF-protect the SYS1.HELP data set, you should give all users READ access authority to the data set because all users need to be allowed to read system help information. You can give READ access authority by placing "SYS1.HELP"/READ in the global access checking table. The system programmers who maintain the data set can be given ALTER access authority by way of a discrete profile or a fully qualified generic profile.

Examples of other system data sets that fall into this category are:

- SYS1.MACLIB
- SYS1.PARMLIB
- SYS1.SAMPLIB

**Note:** SYS1.PARMLIB is in both lists of examples because there are some system functions for which RACF protection is bypassed when accessing SYS1.PARMLIB, and some for which it is enforced. For example, TCAS requires access to SYS1.PARMLIB.

See Appendix D, “Security for system data sets,” on page 711 for guidelines about setting appropriate UACC values for system data sets.

## DASD volume authority

---

By defining profiles in the DASDVOL class, you can define DASD volumes to RACF and authorize users to perform maintenance operations (such as dump, restore, scratch, print, and rename) without having access to the data set profiles protecting the data sets. (If a user does not have the necessary DASDVOL authority, he or she must have the necessary authority in the DATASET class to each of the data sets on the volume.)

The access authority that you give to a user depends on the product that the user is using:

- If the user is using DFSMSdss, the access authority required depends on the specific action that the user is requesting (for example, DUMP with DELETE or DUMP without DELETE). For a complete description of the access authorities required, see [z/OS DFSMSdss Storage Administration](#).
- If the user is using the DADSM scratch function, ALTER access authority allows the user to scratch data sets on the volume.

**Note:** If a data set protected by a discrete profile is scratched, the discrete profile is deleted, or, in the case of a multivolume data set, the volume serial number is removed from the data set profile.

- If the user is using the Device Support Facilities (ICKDSF) program, ALTER allows the user to rename DASD volumes.
- Other products can also check for authorization in the DASDVOL class.

**Exception:** DASDVOL authority does not allow users to perform DFSMSdss logical operations on SMS-managed data sets. To allow logical operations, you can either give the user the OPERATIONS (or group-OPERATIONS) attribute or, if you have the necessary software, define the user as an authorized storage administrator. For more information on the latter alternative, see [“DFSMSdss storage administration” on page 168](#).

As an alternative to assigning the OPERATIONS or group-OPERATIONS attribute, DASDVOL authority allows you to authorize operations personnel to access only those volumes that they must maintain. Using DASDVOL authority is also more efficient for functions such as volume dumping, because only one authorization check for the volume needs to be issued, instead of individual requests for each data set on the volume. For a description of the OPERATIONS attribute, see [“The OPERATIONS attribute” on page 59](#).

If the volume serials do not readily allow the use of \* or % as generic characters in DASDVOL profile names, consider creating profiles in the GDASDVOL class. See [“Creating resource group profiles” on page 212](#).



## DFSMSdss storage administration

---

Operations personnel must routinely perform maintenance operations such as copying, reorganizing, cataloging, and scratching data. To perform these operations on RACF-protected resources, they need RACF authorization. Two methods of providing this authorization are:

- Giving the user the OPERATIONS or group-OPERATIONS attribute
- Defining profiles for volumes in the DASDVOL class and authorizing the user to access the volumes

Both of these methods have certain drawbacks. For example, the OPERATIONS or group-OPERATIONS attribute might give the user more authority to look at individual data sets than you would like. Defining DASDVOL profiles might require more administrative overhead than you would like and, in addition, does not allow a user to work with SMS-managed volumes.

If you use DFSMSdss, there is a third method you can use. You can define special FACILITY class profiles that let a user act as a DFSMSdss storage administrator. Once the profiles and permissions are set up, the user obtains authorization to the required volumes by specifying the ADMINISTRATOR option on the appropriate DFSMSdss command.

DFSMSdss storage administration is more flexible and requires less administrative work to maintain. For complete details on how to set up and use the support, see [z/OS DFSMSdss Storage Administration](#).

## Protecting data on tape

---

This topic gives detailed information that applies to tape data sets and tape volumes. You should already be familiar with the information in [“Protecting data sets” on page 147](#), which applies to both DASD and tape data sets.

This topic describes how to protect and manage access to data on tape using RACF tape volume security when you have no tape management system. If you use a tape management system, refer to your product documentation. If you use z/OS DFSMSrmm, see [z/OS DFSMSrmm Implementation and Customization Guide](#) for details.

RACF allows you to establish access requirements for data on tape to do either or both of the following:

- Control access to the tape volume by issuing SETROPTS CLASSACT(TAPEVOL).

The TAPEVOL class is not active when RACF is first installed.

- Control access to individual tape data sets on the tape volume by issuing SETROPTS TAPEDSN.

NOTAPEDSN is in effect when RACF is first installed.

Many installations implement a tape management system, such as DFSMSrmm, to satisfy their tape management requirements. Note that the implementation of a tape management system replaces the use of the RACF support provided by the TAPEVOL class and the SETROPTS TAPEDSN function.

**Guideline:** When you use a tape management system, such as DFSMSrmm, do not enable the TAPEVOL class or the SETROPTS TAPEDSN function. For more information, see [“Using DFSMSrmm with RACF” on page 168](#).

## Using DFSMSrmm with RACF

When you exploit the capabilities of DFSMSrmm, DFSMSdfp, and RACF, you can protect and manage access to data on tape using RACF profiles in the DATASET class, *without* activating SETROPTS TAPEDSN or the TAPEVOL class. You can also implement a common authorization for all data sets on a tape volume, and authorize users to overwrite tape volumes using RACF erase-on-scratch processing.



If you are new to z/OS or have implemented DFSMSrmm (or equivalent) to protect and manage access to data on tape, you need not activate SETROPTS TAPEDSN nor the TAPEVOL class. You can use the following SETROPTS options:

```
SETROPTS NOTAPEDSN NOCLASSACT(TAPEVOL)
```

If you have already implemented RACF tape volume security, DFSMSrmm supports RACF tape volume security with any combination of RACF TAPEVOL and TAPEDSN options. To support your migration to the NOTAPEDSN and NOCLASSACT(TAPEVOL) environment, DFSMSrmm provides the TPRACF(CLEANUP) option to delete TAPEVOL profiles and discrete tape DATASET profiles during the recycling of scratch tapes.

Beginning with z/OS Version 1 Release 8, DFSMSrmm supports the TAPEAUTHDSN and TAPEAUTHF1 options specified in DEVSUPxx member of SYS1.PARMLIB. (See *z/OS MVS Initialization and Tuning Reference* for information about using these options to enable tape authorization checking.)

#### **TAPEAUTHDSN**

Enables RACF authorization checking in the DATASET class for tape data. This allows authorized users to overwrite tape volumes using RACF erase-on-scratch processing. (See [“Erasing scratched or released data \(ERASE option\)”](#) on page 124.)

#### **TAPEAUTHF1**

Enables RACF authorization checking in the DATASET class for the first file on a tape volume when any file on the same tape volume is opened. This allows a common authorization for all data sets on the volume.

For details, see [z/OS DFSMSrmm Implementation and Customization Guide](#).

## **Choosing which tape-related options to use**

The following sections list considerations for each combination of tape volume (TAPEVOL) and tape data set (TAPEDSN) protection.

### **Tape data set and tape volume protection (TAPEDSN active and TAPEVOL active)**

Using the ADDSD command for a tape data set results in two discrete profiles: an automatic tape volume profile that contains a tape volume table of contents (TVTOC) and a tape data set profile. This means that RACF provides:

- Checking of the RACF security retention period (the number of days that must elapse before the data set can be deleted or overwritten).
- Verification for the full 44-character data set names.
- Protection for multiple data sets on a volume if all of the data sets have the same access requirements.
- Multivolume data set protection.
- RACF protection for the volume.
- Automatic deletion of the data set and tape volume profiles when the data set or tape volume is overwritten and discrete protection for the data set has expired.

Normally, you use the SET operand (which is the default) on the ADDSD command. If the tape volume and data set profiles get out of synchronization (that is, if the tape volume profile refers to a data set profile that does not exist or vice versa), use either the NOSET or SETONLY operand. Use NOSET if you have a data set profile but no tape volume profile. Use SETONLY if you have a tape volume profile but no data set profile.

- Having ADSP or specifying PROTECT=YES on the JCL DD statement also results in two discrete profiles, just as the ADDSD command does.
- Data management calls RACF in the DATASET class.

For tapes being opened for input, data management issues a RACROUTE REQUEST=AUTH, CLASS=DATASET, DSTYPE=T macro. For tapes being opened for output, data management issues a RACROUTE REQUEST=DEFINE, CLASS=DATASET, DSTYPE=T macro.

RACF authorizes access to protected tape data sets through RACF authorization checking. RACF bypasses any tape data set password protection. If the tape data set is not RACF-protected or the tape protection option is not active, data management authorizes access to tape data sets by password protection.

The following notes apply to ICH408I messages when you use generic TAPEVOL profiles:

- The WARNING option does not provide the expected ICH408I warning messages when added to a generic TAPEVOL profile.
- The ICH408I message that is issued when access to a tape data set is denied includes incorrect information when the data set is protected by both a DATASET profile and a generic TAPEVOL profile (without the WARNING option). When the data set is protected by only a TAPEVOL profile and access is denied, the ICH408I message correctly indicates the ACCESS ALLOWED level based on the TAPEVOL profile. However, when the data set is protected by both profiles and the DATASET profile would allow access, access is denied but the ICH408I message incorrectly indicates the ACCESS ALLOWED level based on the DATASET profile instead of the TAPEVOL profile.

### **Tape data set protection (TAPEDSN active and TAPEVOL inactive)**

- Tape volumes have no RACF protection.
- Using the ADDSD command with the TAPE operand gives only a profile in the DATASET class; there is no tape volume profile or TVTOC. This means:
  - No checking of the RACF security retention period (the number of days that must elapse before the data set can be deleted or overwritten)
  - No RACF integrity for full 44-character data set names
  - No RACF protection for multiple data sets on a volume
- Data management calls RACF in the DATASET class.

With TAPEDSN active and TAPEVOL inactive, data management still uses RACROUTE REQUEST=AUTH when tapes are opened for input, and RACROUTE REQUEST=DEFINE when tapes are opened for output. However, RACF does not use the TVTOC during its processing, but assumes that information such as data set name, file (data set) sequence number, and tape volume label are correct.

### **Tape volume protection (TAPEVOL active and TAPEDSN inactive)**

A tape volume is RACF-protected when it is explicitly defined to RACF for protection. Tape volumes are defined to RACF by (1) an authorized user issuing the RDEFINE command without the TVTOC operand, or (2) RACROUTE REQUEST=DEFINE when the TAPEVOL class is active and the PROTECT parameter is specified on a JCL DD statement or during EOVS processing.

RACF authorizes access to protected tape volumes through RACF authorization checking. RACF bypasses any tape data set password protection. If the tape volume is not RACF-protected, or the SETROPTS TAPEDSN option is not active, data management authorizes access to tape data sets by password protection.

If RACF authorizes a user to access an explicitly defined tape volume, the user has access to all of the tape data sets on the volume. Therefore, you should only place tape data sets that have similar RACF authorization requirements on the same volume.

- You can protect tapes only by volume, by using discrete tape volume profiles (PROTECT=YES or the RDEFINE command). However, you can specify PROTECT=YES for multiple data sets on the same volume (the profile is reused).
- Using the ADDSD command for a tape data set results in an error message.
- Data management calls RACF in the TAPEVOL class.

## No tape volume or tape data set protection (TAPEVOL inactive and TAPEDSN inactive)

- You have no RACF protection for data on tape, either by volume or by data set.
- Using the ADDSD command for a tape data set results in an error message. However, you can use the RDEFINE, RALTER, RDELETE, and RLIST commands for tape volume profiles, which provide protection if TAPEVOL or TAPEDSN are activated.
- Data management does not call RACF.

## Protecting existing data on tape (SETROPTS TAPEDSN in effect)

To protect an existing data set on a tape volume, issue the ADDSD command with the TAPE operand. (This requires that the TAPEDSN option be in effect.) If the data set is *cataloged*, you need to specify only the data set name.

The following example shows how to protect a cataloged tape data set named USER01.TEST.DATA:

```
ADDSD 'USER01.TEST.DATA' TAPE
```

If the cataloged tape data set resides on more than one volume (a multivolume tape data set), RACF uses the data set name specified on the ADDSD command and the information supplied in the catalog to protect the data set on all of the volumes on which it resides.

To protect an existing tape data set that is *uncataloged*, issue the ADDSD command with the TAPE operand and specify:

- The data set name
- The tape volume on which the data set resides
- The unit name
- The file sequence number of the data set on the tape

For example, suppose you want to protect an uncataloged tape data set named USER03.TEST.DATA with a discrete RACF profile. The data set resides on a tape volume labeled 123456 and has a file sequence of 1. To protect this data set, enter:

```
ADDSD 'USER03.TEST.DATA' TAPE UNIT(TAPE) VOLUME(123456) FILESEQ(1)
```

From this information, RACF builds a discrete profile for both the data set and the tape volume. When you issue the ADDSD command to protect an existing tape data set, RACF creates an automatic tape volume profile. For more information, see [“Tape volume profiles that contain a TVTOC” on page 175](#).

Note that when you issue the ADDSD command to RACF-protect an uncataloged tape data set, you protect that data set only on the volume that you specify.

If you have an uncataloged tape data set that resides on more than one volume, you can RACF-protect this data set with a discrete profile using several commands. For example, suppose you want to RACF-protect a tape data set named USER02.TEST.DATA that resides on volumes 111111 and 222222.

1. To protect that portion of the data set residing on volume 111111, issue the ADDSD command:

```
ADDSD 'USER02.TEST.DATA' TAPE UNIT(TAPE) VOLUME(111111) FILESEQ(1)
```

2. To protect that portion of the data set residing on volume 222222, issue the ALTDSD command with the ADDVOL operand as follows:

```
ALTDSD 'USER02.TEST.DATA' ADDVOL(222222)
```

### Note:

- a. You can protect only one volume at a time with the ALTDSD command and the ADDVOL operand. If your data set resides on more than two volumes, issue the ALTDSD command and specify the

appropriate volume on the ADDVOL operand for each additional volume. For a tape data set with an entry in the TVTOC, the maximum number of volumes the data set can span is 42.

- b. Before you can use the ALTDSD command to protect a portion of a multivolume data set, at least one other portion of that data set must already be RACF-protected.
- c. RACF ignores this command if you specify a generic profile name for the data set.

## Protecting new data on tape

Your installation can provide RACF protection for new tape data sets by using one or more of the following methods.

- A user can specify PROTECT=YES on the JCL DD statement when creating a new tape data set.

RACF builds a discrete profile for the newly created data set and the tape volume on which the data set resides (unless the tape volume is already defined with the TVTOC option).

When TAPEDSN and TAPEVOL are both active, a discrete data set profile is defined. If the tape volume profile already exists, the TVTOC is updated. If the tape volume profile does not already exist, RACF defines an automatic tape volume profile with a TVTOC.

When TAPEDSN is not active and TAPEVOL is active, a discrete TAPEVOL profile is defined.

- You can assign the ADSP attribute to a user and issue the SETROPTS ADSP command.

When the user creates a new tape data set, RACF automatically builds a discrete profile for the data set as well as the tape volume on which the data set resides (unless the tape volume is already defined with the TVTOC option).

## Protecting tape volumes

This topic describes how to RACF-protect tape volumes using the RDEFINE command with or without the TVTOC operand.

You can provide RACF protection for tape data sets by creating tape volume profiles that protect the tape volumes on which the data sets reside.

### *Defining tape volumes with a TVTOC*

To provide protection for tape data sets, you (or an assigned administrator) can predefine individual tape volumes to RACF using the RDEFINE command with the TAPEVOL class and TVTOC operand. Tape volumes defined with the RDEFINE command and TVTOC operand are called *scratch pool volumes*.

When RACF processes the RDEFINE command with the TVTOC operand, it places the user ID of the command issuer in the access list of the volume with ALTER authority. A scratch pool volume can be used by any RACF-defined user for output (for writing). When the first user writes a data set to a scratch pool volume, RACF places the user ID of that user in the access list of the volume with ALTER authority. After RACF creates the volume's access list, only the command issuer, the first user of the volume, and any users added to the access list with UPDATE authority can write additional data sets to the volume.

For example, to define a tape volume labeled TX0050 with the attribute that it can hold a TVTOC and assign it a UACC of NONE, enter:

```
RDEFINE TAPEVOL TX0050 TVTOC UACC(NONE)
```

After you define a tape volume with a TVTOC, you can use generic profiles to protect data sets that reside on that volume. To define a generic profile for data sets, use the ADDSD command and specify the profile name.

The following example shows how to define the generic profile USER03.\*.

```
ADDSD 'USER03.*'
```

### **Note:**

1. The user ID of the issuer of RDEFINE is placed automatically on the access list with ALTER only if SETROPTS ADDCREATOR is in effect.
2. The TAPEVOL class must be active for the RDEFINE command to succeed. For more information, see [“Activating tape volume protection \(TAPEVOL option\)”](#) on page 123.
3. The TVTOC operand applies only to discrete tape volume profiles.
4. When you issue the RDEFINE command with the TVTOC operand, you create a nonautomatic tape volume profile. For more information, see [“Tape volume profiles that contain a TVTOC”](#) on page 175.
5. When you issue the ADDSD command, you can predefine a generic data set profile, or define a generic profile after the data set and TVTOC entry have been created. You can also use existing generic profiles that were created to protect DASD data sets. If you are using generic data set profiles for tape data sets, you should specify a retention period in those profiles because the SETROPTS retention period is not used.
6. The access authorities that apply to tape volume profiles are as follows:

**NONE**

Allows no access to data on the tape volume.

**READ**

Allows users to read from the tape volume.

**UPDATE**

Allows users to read from the tape volume, and to write additional data sets to the volume.

**CONTROL**

Is equivalent to UPDATE.

**ALTER**

Allows users to read from the tape volume, to write additional data sets to the volume, and to create or destroy tape volume labels through OPEN or end-of-volume operations. For discrete tape volume profiles, allows users to change the profile, including the access list.<sup>3</sup>

### ***Authorizing access to a data set on a tape volume with a TVTOC***

RACF maintains an entry in the TVTOC for each data set that a user writes to a scratch pool volume. The data set can be:

- Protected by a discrete profile, an appropriate generic profile, or both
- Not protected by any profile

When a user requests access to a data set on the tape volume, RACF performs access checking as follows:

1. RACF checks the user's authority to the volume on which the data set resides. If the user has sufficient authority to the volume, RACF grants access to the data set. If the user does not have sufficient authority to the volume, access checking proceeds with Step 2.
2. RACF checks to see if the data set is RACF-indicated. (For more information on RACF-indicated data sets, see [“Protection through discrete profiles”](#) on page 150.) If the data set is RACF-indicated, access checking proceeds with Step 3; if not, access checking proceeds with step 4.
3. The data set is RACF-indicated. RACF checks for a discrete profile that protects the data set. If RACF does not find a discrete profile, access checking proceeds with Step 4. If RACF finds a discrete profile and the user has sufficient authority to the data set, RACF grants access. If the user does not have sufficient authority to the data set, RACF denies access.
4. RACF searches for an appropriate generic profile. If RACF finds a generic profile and the user has sufficient authority to access the data set, RACF grants the request. If the user does not have sufficient authority to access the data set, RACF fails the request. If RACF does not find a generic profile, RACF fails the request.

<sup>3</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

## Defining tape volumes without a TVTOC

You can also define tape volumes without using the TVTOC operand. When you define a tape volume in this manner, RACF does not maintain a TVTOC to control access to data sets on the volume. Instead, RACF controls access to data sets on the tape volume using only the access list in the volume's profile. Users with at least READ authority to the volume can read any data on the tape. Users with at least UPDATE authority to the volume can write data on the tape.

The following sequence of commands shows how to define a tape volume without a TVTOC and how to control access to the data sets on that volume.

1. To define and protect a tape volume, issue the RDEFINE command with the appropriate operands and assign a UACC of NONE to the volume.

```
RDEFINE TAPEVOL profile-name UACC(NONE)
```

For example, to define a tape volume labeled 123456 and assign it a UACC of NONE, issue the following command.

```
RDEFINE TAPEVOL 123456 UACC(NONE)
```

The RDEFINE command adds a profile for the tape volume to the RACF database.

2. To allow a user access to the volume for the purpose of creating data sets, issue the PERMIT command with the appropriate operands and give the user UPDATE access authority. For tape volume 123456, enter the command as follows.

```
PERMIT 123456 CLASS(TAPEVOL) ID(userid or groupname) ACCESS(UPDATE)
```

UPDATE access authority allows a user to read and write data sets to the tape volume. You should *not* assign ALTER access authority to a general user because ALTER allows a user to overwrite the tape label.

3. If other users want to access the data on the tape volume, issue the PERMIT command with the appropriate operands and access authority. For example, to give another user READ access authority to tape volume 123456, issue the following command.

```
PERMIT 123456 CLASS(TAPEVOL) ID(userid or groupname) ACCESS(READ)
```

Note that a user must have sufficient authority to issue the PERMIT command. Because you gave the user who requested the tape volume UPDATE access authority, that user does not have sufficient authority to allow other users to access the tape volume.

4. When a user has finished working with the tape volume, issue the PERMIT command and specify the RESET(ALL) operand. RESET(ALL) deletes the entire current standard and conditional access lists from the tape volume's profile. For tape volume 123456, enter the command as follows.

```
PERMIT 123456 CLASS(TAPEVOL) RESET(ALL)
```

If you delete only the access lists from a tape volume profile, you retain RACF protection for data on the volume. (In this case, no users can access the data.) If you delete the tape volume profile itself, you have no RACF protection for data on the volume. (Any user can access the data.)

## Security levels and security categories for tapes

As an option, you can add security level and security category protection to the tape volume's profile. To achieve this additional protection, use the RALTER command with the appropriate operands.

For example, to add a security level of CONFIDENTIAL and security categories of PLANNING and STATUS to the profile of tape volume 123456, enter:

```
RALTER TAPEVOL 123456 SECLEVEL(CONFIDENTIAL)
      ADDCATEGORY(PLANNING, STATUS)
```

When a user creates data sets on tape volume 123456, the user should ensure that each data set has the same security level and security categories as specified on the RALTER command.

To delete the security level and all of the security categories from the volume's profile (for example, when a user has finished working with a tape volume), issue the RALTER command with the NOSECLEVEL and DELCATEGORY(\*) operands. For tape volume 123456, enter the command as follows:

```
RALTER TAPEVOL 123456 NOSECLEVEL DELCATEGORY(*)
```

## Security labels for tapes

To add a security label of LABEL1 to the profile of tape volume 123456, enter:

```
RALTER TAPEVOL 123456 SECLABEL(LABEL1)
```

When a user creates data sets on tape volume 123456, the user should ensure that each data set has the same security label (LABEL1) as was specified on the RALTER command that defined the tape volume.

**Note:** When the SECLABEL class is active, and both the SETROPTS MLS(FAILURES) and SETROPTS MLACTIVE options are in effect, the security label of the tape volume profile is used for all data sets on the volume. For tape volume profiles without TVTOCs, the security label is assigned when the tape volume profile is defined. Tape volume profiles with TVTOCs are assigned a security label when the first data set is written to the tape.

To delete the security label from the volume's profile (for example, when a user has finished working with a tape volume), issue the RALTER command rather than the RDELETE command with the NOSECLABEL operand. For tape volume 123456, enter the command as follows:

```
RALTER TAPEVOL 123456 NOSECLABEL
```

## Tape volume profiles that contain a TVTOC

If tape data set protection and the TAPEVOL class are both active, RACF creates and maintains a tape volume table of contents (TVTOC) that is part of the tape volume profile in the RACF database. The following topic describes the TVTOC.

### Tape volume table of contents (TVTOC)

RACF creates and maintains the TVTOC for tape volumes that:

- Are defined using the RDEFINE command with the TVTOC operand
- Contain tape data sets protected by using the ADDSD command
- Contain tape data sets protected by specifying PROTECT=YES on the JCL DD statement
- Contain tape data sets created by a RACF-defined user who has the ADSP attribute

The TVTOC contains the following information:

- The number of data sets on the volume
- The full 44-character name used when creating the data set (from the DSN operand of the JCL DD statement)
- The volume serial number of each volume on which the data set resides (from the VOL operand of the DD statement)
- The file (data set) sequence number (from the LABEL operand of the JCL DD statement)
- The RACF internal data set name (from the naming conventions table or an installation exit routine)
- The data set creation date
- For each data set on the volume, an indicator that is set if the data set is protected by a discrete profile

You can list the information in the TVTOC of a tape volume profile by using the RLIST command. For more information, see [z/OS Security Server RACF Command Language Reference](#).

RACF makes entries in the TVTOC when a user:

- Opens a new data set on a predefined tape volume, or
- Protects a new or existing tape data set using RACF

RACF then uses the information during access checking.

**Note:**

1. The maximum number of entries for data sets that a TVTOC can contain is 500.

**Important:** Processing that creates large numbers of TVTOC entries and large access lists might result in an attempt to exceed the maximum profile size.

2. The maximum number of volumes that any data set on the tape with an entry in the TVTOC can span is 42.
3. The maximum number of volumes that any data set on tape without a TVTOC can span is limited only by the maximum profile size.

When both TAPEDSN and TAPEVOL are active, RACF can create two different types of TVTOC profiles:

- An automatic TVTOC tape volume profile
- A nonautomatic TVTOC tape volume profile
- The NOSET option on the DELDSD command can be used to remove a discrete tape data set profile without deleting the tape volume profile. For more information, see [z/OS Security Server RACF Command Language Reference](#).

The sections that follow describe these profiles.

### Automatic TVTOC tape volume profiles

RACF creates an *automatic TVTOC tape volume profile* when one of the following occurs:

- A RACF-defined user has the ADSP attribute and creates a tape data set on a non-RACF-defined tape volume.
- A RACF-defined user creates a tape data set on a non-RACF-defined tape volume by specifying PROTECT=YES on the JCL DD statement.
- A RACF-defined user protects an existing tape data set on a non-RACF-defined tape volume using the ADDSD command with the appropriate operands.

When RACF creates an automatic tape volume profile, RACF does not use modeling, except possibly for the owner field as specified as follows. The tape volume profile that RACF creates contains the following fields:

- Owner: The user ID creating the profile, unless a different owner is specified by REQUEST=DEFINE or an ADDSD command, or a discrete data set profile is being created and the model profile specifies an owner
- Universal access authority (UACC)
- Access list: The creating user ID with ALTER authority
- Audit criteria: FAILURES(READ)
- RESFLG: Indicates the profile is automatic
- TVTOC: The tape volume table of contents

You can change any of these fields by using the RALTER or PERMIT command. (The most likely change is adding other users to the access list so that they can define data sets on the tape volume.)

When the security retention periods for all data sets on a volume that is protected by an automatic tape volume profile have expired and a user uses the volume for output, RACF deletes the volume's profile. When a user creates a new data set on such a tape volume and specifies PROTECT=YES on the JCL DD statement or has the ADSP attribute, RACF creates a new discrete tape volume profile with a TVTOC and generates a discrete profile for the data set. If the user does not specify PROTECT=YES on the JCL DD



statement or have the ADSP attribute, RACF does not create new profiles for the volume or the data set. Therefore, the volume and any data sets on it are no longer RACF-protected and any user can read or write data on the volume.

## Nonautomatic tape volume profiles

RACF creates a *nonautomatic tape volume profile* when:

- A user predefines a tape volume using the RDEFINE command with the TVTOC operand, or
- A user modifies an automatic tape volume profile with the ALTDSD or PERMIT command

When the security retention periods for all data sets on a volume that is protected by a nonautomatic tape volume profile expire, RACF does not delete the volume's profile. However, the volume that is protected by the profile can be reused. As users write new data sets to the volume, RACF updates the volume's TVTOC to reflect the addition of new data sets even if the data sets are not RACF-protected.

## Predefining tape volume profiles for tape data sets

Rather than defining individual tape volumes for use by specific users, installations can predefine scratch pool volumes with tape volume profiles for use by any user. An installation tape librarian can predefine tape volume profiles to RACF by using the RDEFINE command with the TVTOC operand and optionally, the SINGLEDSN operand. The TVTOC operand indicates that RACF creates a TVTOC the first time the tape is opened for output. The SINGLEDSN operand indicates that the tape volume can contain only one data set.

Predefining tape volumes when TAPEVOL and TAPEDSN are both active has the following advantages:

- To get a tape volume profile with a TVTOC, users do not have to have ADSP, use PROTECT=YES in the JCL, or manually define tape data sets with the ADDSD command.
- It is easier for users to use generic profiles for tape data sets. (If a user creates a tape data set and the user has ADSP or specifies PROTECT=YES, RACF always creates a discrete profile for the tape data set.)

To predefine tape volumes, the installation tape librarian selects new or newly degaussed tape volumes in the scratch pool for use with RACF tape data set protection. The librarian defines these tape volumes to RACF with a nonautomatic discrete profile by using the RDEFINE command and the TVTOC operand. (If you do not specify the TVTOC operand, the default is NOTVTOC.) RACF puts the user ID of the person who defines the tape volume (presumably the tape librarian) in the access list with ALTER authority. This action gives the librarian complete control over the profile and the tape volume. RACF puts the user ID in the access list with ALTER authority only if SETROPTS ADDCREATOR is in effect. If SETROPTS NOADDCREATOR is in effect, the tape librarian needs to ensure that they are the owner of the profile and should issue the PERMIT command to give themselves ALTER authority if they need to have complete control over the profile and the tape volume.

When the first user creates a tape data set on a predefined tape volume, RACF builds a TVTOC in the tape volume profile and places the user ID of this person in the access list with ALTER authority. If the volume is defined with the SINGLEDSN operand, no one can write additional data sets on the volume. If the volume is not defined with the single-data-set option, only this user (and the tape librarian) can add additional data sets to the volume without further authorization. Other users can add data sets to the volume only if they have been placed in the volume's access list with at least UPDATE authority.

When the tape librarian needs more tape volumes for the scratch pool, the librarian can issue the SEARCH command with the EXPIRES operand to find tape volumes for which the security retention period for all of the data sets is expired (or close to expiring). The librarian can then use the RDELETE and RDEFINE commands to redefine these tape volumes.

Unlike DASD data sets, tape data sets are not deleted. A tape data set exists until it is overwritten by another program or by a utility such as IEHINITT. Specifying DISP=DELETE for a tape data set only causes the data set to be uncataloged if it was cataloged. DISP=DELETE does not remove RACF protection from the data set or delete the data on the tape volume.

## RACF security retention period processing (TAPEDSN must be active)

Before RACF allows a user to write to a tape that is protected by a tape volume profile containing a TVTOC, RACF checks whether the security protection for the current data on the tape volume has expired. To determine whether the RACF security retention period has expired, RACF uses one of the following:

- The RACF security retention period stored in the data set profile (specified using the RETPD operand on the ADDSD or ALTDSD command)
- If the data set profile does not contain a security retention period, one of the following:
  - For discrete profiles, RACF uses the creation date stored in the TVTOC and the default security retention period established by your installation using the RETPD operand on the SETROPTS command.
  - For generic profiles, RACF uses a zero value. This results in the data set being expired. For generic profiles, the default security retention period is *not* checked. Therefore, you must ensure that all generic profiles that protect tape data sets include a retention period. (Make sure to specify the RETPD operand on the ADDSD command for generic profiles.)

If a user wants to overwrite an existing tape data set with a data set having a different name before the existing data set's RACF security retention period has expired, the user must do one of the following:

- Explicitly delete the data set profile using the DELDSD or RDELETE command
- Have at least UPDATE authority to the volume

If the user has sufficient authority to a tape volume or tape data set, the user can overwrite an existing data set using one of the following:

- The same data set name
- A data set name defined to RACF to which the user has authority
- A data set name not defined to RACF

If the RACF security retention period for an existing tape data set has not expired and the user does not have sufficient authority to overwrite it, RACF issues a message indicating that the user does not have sufficient authority to the volume or data set.

When a user specifies PROTECT=YES on the JCL DD statement, RACF updates the TVTOC to reflect the creation of the new data set. RACF also generates a discrete profile to protect the new data set and deletes any existing discrete profile that protected the overwritten data set.

A user can specify the security retention period for a tape data set by one of the following methods:

- For a data set protected by either a discrete or generic profile, by using the RETPD operand on the ADDSD or ALTDSD command
- For a data set protected by a discrete profile, by specifying the EXPDT or RETPD operand on the JCL DD statement

For discrete profiles, if a user does *not* specify a security retention period for a tape data set, the retention period can be provided by one of the following:

- Profile modeling
- An installation exit routine
- A system-wide default set by the RETPD operand on the SETROPTS command

For generic profiles that protect tape data sets, the user must assign a security retention period to the profile by specifying the RETPD operand on the ADDSD or ALTDSD command. (If the security retention period is omitted, a zero value is used and the profile is treated as if it expired.)

When RACF is installed, the default security retention period is RETPD(0). If your installation specifies a different default security retention period for tape data sets, RACF uses the specified value in any of the following situations:

- When a user specifies RETPD=0 on the JCL DD statement

- When a user specifies EXPDT=*current-date* on the JCL DD statement
- When a user does not specify the EXPDT/RETPD JCL operands

**Note:** The RACF security retention period is independent of the data set retention period specified by the EXPDT/RETPD JCL operand. However, the two retention periods are initially the same if the user who creates the data set has ADSP or specifies PROTECT=YES on the JCL DD statement. You can modify the security retention period in the data set profile by using the ALTDSD command.

If a tape volume contains more than one data set, RACF protects each data set independently. RACF achieves this protection by not allowing users with UPDATE authority to one or more of the data sets to rewrite any data set until one of the following occurs:

- The profiles for all of the data sets that sequentially follow that data set on the tape volume have been deleted.
- The security retention periods for all of the data sets that sequentially follow that data set on the tape volume have expired.

Note, however, that users who have at least UPDATE authority to the volume can write to the volume unconditionally.

In response to RDELETE or DELDSD commands, RACF deletes tape volume profiles and the discrete tape data set profiles for all data sets residing on tapes when all of the data sets that the TVTOC points to have expired. For generation data groups (GDGs), RACF does not automatically delete RACF protection of the volumes containing the oldest generation when a new generation is defined. Because residual data remains on a tape volume even after the security retention period of the RACF profiles has expired, installations should consider degaussing tape volumes on which all of the data sets have an expired security retention period. The librarian can then redefine these tape volumes to RACF using the RDEFINE command with the TVTOC operand and, thereby, reenter the volumes into the common scratch pool.

## Authorization requirements for tape data sets when both TAPEVOL and TAPEDSN are active

When TAPEVOL is active, users with ALTER authority to a tape volume have full control over the volume profile, including the volume's access list. ALTER authority gives the user the ability to create and delete data sets on the volume and rewrite the tape volume label.

**To open a RACF-protected tape data set for input** (for reading), the user must have at least READ authority to the data set or the volume. When a RACF-protected volume is opened for input and the user does not have the authority necessary to write to the data set, a message might be issued to the system operator to remove the write-enable ring (file protect ring). (The authority necessary to open a RACF-protected tape data set for output is described as follows.) For more information, see [“IEC.TAPERING profile in the FACILITY class” on page 180](#).

**To open a RACF-protected tape data set for output** (for writing), the user must have UPDATE authority to the tape volume, or the following authority:

- **To rewrite or add to a data set without changing the data set name**, the user requires UPDATE authority to the data set. If the data set is not the last data set on the tape volume, all of the subsequent data sets must have passed their security retention periods or be explicitly deleted using the DELDSD or RDELETE command.
- **To overwrite an existing data set on a tape** with a data set of a different name, the security retention periods for the data set and any subsequent data sets must have expired. The user must also have authority to create a data set with the specified name (the authority checks are the same as for DASD data sets).
- **To add a new data set to the end of a tape**, the user requires UPDATE authority to the tape volume, and the volume profile must allow more than a single data set. The user must also have authority to create a data set with the specified name (the authority checks are the same as for DASD data sets).

**Note:** If a data set is in the TVTOC of a tape volume profile, but is not covered by a discrete profile, a generic profile, or an entry in the global access checking table, the data is not RACF-protected.

## Authorization requirements for tape data sets when TAPEVOL is inactive and TAPEDSN is active

**To open a RACF-protected tape data set for input** (for reading), the user must have at least READ authority to the data set. When a RACF-protected tape data set is opened for input and the user does not have the authority necessary to write to the data set, a message might be issued to the system operator to remove the write-enable ring (file protect ring). (The authority necessary to open a RACF-protected tape data set for output is described as follows.) For more information, see [“IEC.TAPERING profile in the FACILITY class” on page 180](#).

**To open an existing RACF-protected tape data set for output** (for writing), the user must have at least UPDATE authority to the data set. To create a new tape data set for output or, to catalog an existing tape data set after opening it for output, the user must have ALTER authority to the data set.

## Authorization requirements for tape data sets when TAPEVOL is active and TAPEDSN is inactive

When TAPEVOL is active, users with ALTER authority to a tape volume have full control over the volume profile, including the volume's access list. ALTER authority gives the user the ability to create and delete data sets on the volume and to rewrite the tape volume label.

**To open a tape data set on a RACF-protected tape volume for input** (for reading), the user must have at least READ authority to the tape volume. When a data set on a RACF-protected tape volume is opened for input and the user does not have the authority necessary to write to the data set, a message might be issued to the system operator to remove the write-enable ring (file protect ring). (The authority necessary to open a tape data set on a RACF-protected volume for output is described below.) For more information, see [“IEC.TAPERING profile in the FACILITY class” on page 180](#).

**To open a tape data set on a RACF-protected tape volume for output** (for writing), the user must have at least UPDATE authority to the volume.

## JCL changes

To protect tape data sets, installations should provide generic profiles or code PROTECT=YES (when TAPEVOL is active) for each data set that requires protection. Data management allows you to specify PROTECT=YES for each data set on a tape volume.

## Installations with DFSMSHsm

DFSMSHsm, the hierarchical storage manager, works with the volume level of protection, so DFSMSHsm tape volume profiles should not contain a TVTOC.

Because a TAPEVOL profile can span a maximum of 10,000 tape volumes, DFSMSHsm includes a way to extend its pool of backup and migration tape volumes. To take advantage of this extension method, add the DFHSM profile to the HSMHSM profile in the TAPEVOL class.

For information about DFSMSHsm, see [z/OS DFSMSHsm Storage Administration](#).

## IEC.TAPERING profile in the FACILITY class

Depending on the release of z/OS DFSMS, the type of tape drive, and any tape management system that are installed on your system, you can allow users to open tape data sets for input without removing the write-enable ring (or equivalent) by creating a profile to protect a resource called IEC.TAPERING in the FACILITY class and allowing users to have READ access authority to this resource.

### Important:

- You should only allow access to the IEC.TAPERING resource for users who can be trusted not to abuse the authority to write to tapes they are allowed to read.

- For IBM 3490 tape drives and for IBM 3480 tape drives with the IDRC feature, this profile is not checked. Instead, the tape device cannot use WRITE operations when the user has only READ authority.

See the following example for setting up an IEC.TAPERING profile:

1. Create a profile to protect the IEC.TAPERING resource:

```
RDEFINE FACILITY IEC.TAPERING UACC(NONE)
```

**Note:** If you want to allow this for all users on your system, specify UACC(READ) and omit the following PERMIT command.

2. Permit users or groups, as appropriate:

```
PERMIT IEC.TAPERING CLASS(FACILITY) ID(userid or groupname)
ACCESS(READ)
```

For more information, see [z/OS DFSMS Using Magnetic Tapes](#).

## Password-protected tape data sets

Your installation can provide password protection to data sets that reside on tape volumes. When you define a tape volume to RACF using the RDEFINE command, data management does not verify password protection when a user requests access to a data set residing on this tape volume. When you define a tape volume to RACF through ADSP or PROTECT=YES in the JCL statement, the user or operator must supply the correct password for the password-protected tape data set on the volume. Passwords should be maintained for tape data sets on RACF-protected tape volumes if they are to be used (1) on systems that do not have RACF installed or (2) on RACF systems where tape protection might not be active. To maintain passwords for tape data sets, password information can be specified on the LABEL operand along with the PROTECT parameter on the same JCL DD statement. Also, the password indicators in tape header and trailer labels are set for RACF-protected tape volumes based on password information specified on the LABEL operand. All password-protected data sets on a RACF-protected tape volume must have the same level of password protection.

## Using the PROTECT parameter for tape data set or tape volume protection

To define a tape data set or a tape volume to RACF for protection by a discrete profile, specify the PROTECT parameter on the JCL DD statement that identifies a tape data set on the volume. If generic profile checking is active, do not use the PROTECT parameter unless a discrete profile is required for the data set. If the data set is also password-protected, the password must be supplied before the tape volume is RACF-protected.

## Multivolume tape data sets

When a multivolume tape data set is opened for input, RACF performs authorization checking for those volumes that are RACF-protected.

When a multivolume tape data set is opened for output, and the first volume is RACF-protected, all succeeding volumes are RACF-protected as part of the same volume set.

When attempting to effect changes to multivolume tape data sets, use the RALTER command rather than the RDELETE command. If the resource you specify is a member of a tape volume set, RACF deletes the definitions for *all* of the volumes in the set.

A single profile is maintained in the RACF database for a tape volume set. Thus all volumes in the volume set share the same access list and the same statistics and auditing options. (For additional information on protecting multivolume tape data sets, see [“Protecting existing data on tape \(SETROPTS TAPEDSN in effect\)”](#) on page 171.)

## RACF authorization of bypass label processing (BLP)

Your installation can specify JES initialization parameters to allow bypass label processing (BLP). For details, see [z/OS JES2 Initialization and Tuning Reference](#).

Other factors, such as the use of a tape management system or certain other system parameters, also affect tape bypass label processing. If your installation uses a tape management system, see its product documentation. Also, see [z/OS MVS Initialization and Tuning Reference](#) for information about the TAPEAUTHDSN parameter in the DEVSUPxx member of SYS1.PARMLIB.

If your system does not support BLP processing, the system converts all BLP requests to requests for nonlabeled tapes. If a labeled tape is mounted to satisfy this specification, RACF performs authorization checking and, if the user has sufficient authority, the label is destroyed. For more information, see [“Tape data set and tape volume protection for nonlabeled \(NL\) tapes” on page 183](#).

If your system supports BLP processing, RACF provides installations with the ability to control the use of the BLP option on JCL DD statements. To control who can use BLP, perform the following steps:

1. Activate the TAPEVOL class.
2. Define a profile in the FACILITY class to protect the ICHBLP resource, and grant users READ or UPDATE authority, as appropriate.

**To open a tape for input and bypass label processing** when the TAPEVOL class is active, the user must have at least READ authority to the volume (if the volume is defined) as well as to the ICHBLP resource in the FACILITY class.

**To open a tape for output and bypass label processing**, the user must have at least UPDATE authority to the volume (if the volume is defined) as well as to the ICHBLP resource in the FACILITY class.

RACF checks the user's authority to the ICHBLP resource when the user attempts to access a tape with an IBM standard or ANSI label (even if BLP is specified on the LABEL operand of the DD statement for the tape volume).

RACF performs BLP authorization checking only if the TAPEVOL class is active. If TAPEVOL is not active, data management does not call RACF to perform BLP or tape access checking.

If RACF finds an ICHBLP profile, RACF verifies that the user has sufficient authority to use bypass label processing. If the user does not have sufficient authority, RACF fails the request.

If RACF does not find an ICHBLP profile or if the user has sufficient authority to use bypass label processing, RACF performs authorization checking on the volume. If the user has sufficient authority to the volume, RACF grants the request. Otherwise, RACF fails the request.

**Note:** RACF performs authorization checking on a volume based on the volume serial number specified on the JCL statement. Proper authorization checking, therefore, depends on the operator mounting the correct volume.

## Authorization requirements for labels

The following rules apply to RACF-protected tape volume labels:

- To rewrite a tape volume label without additional authority, the user must have ALTER authority to the volume.
- To destroy a tape volume label (when converting from standard labels to nonstandard or nonlabeled tapes), the user must have ALTER authority to the volume.
- To create a tape volume label (when writing a standard labeled data set to a nonlabeled or nonstandard labeled volume), the user must have ALTER authority to the volume.

## Tape data set and tape volume protection with nonstandard labels (NSL)

Data management does not do authorization checking for nonstandard labeled tapes. However, if the user is going to destroy standard labels, data management calls RACF to determine whether the tape volume

is protected. If the tape is RACF-protected and TAPEDSN is active, the user must have ALTER authority to the volume and to the data sets on the volume. For more detailed information, consult the DFSMS documents.

## Tape data set and tape volume protection for nonlabeled (NL) tapes

The following topics describe the authorization requirements for opening unlabeled tapes.

### Opening an unlabeled tape for input

To open an unlabeled tape for input, the user must have at least READ authority to one of the following:

- To the volume
- If TAPEDSN is active, to the data sets that might have previously been defined to RACF within the TVTOC for the volume. Because this is a nonlabeled tape volume, RACF does not automatically create or maintain TVTOC data set entries. However, TVTOC data set entries can be manually created and maintained with the ADDSD command.

If data management finds a labeled tape when opening the volume for input, it rejects the request.

### Opening an unlabeled tape for output

To open an unlabeled tape for output when TAPEDSN is active, the user must have at least UPDATE authority to the tape volume and to all data sets on the volume. If a labeled tape is encountered when opening an output tape and the user has ALTER authority to the volume, the volume label is destroyed and RACF protection for the volume is deleted. For additional details, see [z/OS DFSMS Using Magnetic Tapes](#).

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

#### Note:

1. Because protection is on a volume level, you should specify PROTECT=YES in the DD statement for only one data set on the volume.
2. RACF protection of nonlabeled tapes does not depend on tape data set protection being active.
3. RACF does not maintain the TVTOC for nonlabeled tapes. But if the TVTOC contains any entries, RACF uses these entries during authorization checking.
4. You cannot RACF-protect nonlabeled tapes that have a volume serial number of "Lnnnnn".





## Chapter 8. Protecting general resources

This topic provides in-depth information about protecting general resources.

RACF-protected resources can be divided into two categories: data sets and general resources. General resources are all of the resources that are defined in the class descriptor table. For example, general resources include DASD and tape volumes, load modules (programs), terminals, and others.

See [Chapter 13, “Protecting programs,” on page 301](#) for information about controlling programs, program libraries, and program access to data.

### Defining profiles for general resources

The RACF commands that you can use to work with general resource profiles are shown in [Table 13 on page 185](#).

*Table 13. RACF commands used with general resource profiles*

Activity	Command
Defining	RDEFINE
Listing	RLIST
Changing	RALTER
Allowing or denying access	PERMIT with CLASS( <i>classname</i> )
Searching	SEARCH with CLASS( <i>classname</i> )
Deleting	RDELETE

**Note:** For the authority needed to issue any of these commands, see [z/OS Security Server RACF Command Language Reference](#).

### Summary of steps for defining general resource profiles

This summary presents the steps required by RACF to define general resource profiles. Note that specific instructions for most resources supported by the supplied general resource classes are contained elsewhere in this document.

1. Determine which resources are to be protected by the profile. This involves the following information:

- The general resource class, such as TAPEVOL or TERMINAL
- The profile name:
  - If you specify a generic profile name, the profile can protect more than one resource.

Using generic profiles instead of discrete profiles can greatly reduce the effort of maintaining the profiles. In general, create generic profiles to cover most resources, using discrete profiles only for exceptions.

Also, consider creating a profile to be used as a model, especially if you are specifying complex access lists. Models can be used when creating any kind of resource profile (discrete or generic), and modeling can be done across classes. To model, specify the FROM operand on the RDEFINE command. To model across classes, you should also specify the FCLASS operand. Before using modeling, see [“Possible changes to copied profiles when modeling occurs” on page 37](#).

**Note:** To specify generic profile names, either generic command processing or generic profile checking (the SETROPTS GENCMD or SETROPTS GENERIC option) must be in effect for the class. For example, for the TERMINAL class:

### SETROPTS GENERIC(TERMINAL)

- If you specify a discrete profile name, the profile can protect only one resource.
- The rules for specifying profile names for most supplied general resource classes are described elsewhere in this document.

#### Note:

- a. For some kinds of resources, (such as terminals, DASD volumes, and CICS, IMS, and SDSF classes), you should consider using resource group profiles instead of generic profiles. Creating resource group profiles can save a significant amount of work. See [“Creating resource group profiles” on page 212](#) for more information.
  - b. You can use RACFVARS profiles to specify values for variables (indicated by an ampersand &) in profile names. For more information, see [“Using RACF variables in profile names \(RACFVARS class\)” on page 216](#).
- Decide which access is to be allowed to all users on the system who are not otherwise limited. In RACF, this is called the universal access authority (UACC). This has the same meaning as the access authority on access lists (see Step [“3” on page 187](#)). In most cases, the UACC should be NONE or READ.
  - Decide which user or group is to be the owner of the new resource profile. By default, this is the user who creates the profile.
    - If the owner is a user, the owner can list, modify, or delete the resource profile. Note that being the owner of a resource profile does not, by itself, allow a user to have access to the resource or resources that are protected by the profile. For more information, see Step [“17” on page 721](#) in [“Authorizing access to RACF-protected resources” on page 719](#).
    - If the owner is a group, the authority of a user who has a group-level attribute in that group (such as group-SPECIAL or group-AUDITOR) extends to resources that are protected by this profile.
  - Decide which user, if any, should be notified by a message when users make unsuccessful attempts to access resources that are protected by the profile (NOTIFY operand).
  - Decide whether RACF should log access attempts to resources that are protected by the profile (AUDIT operand).

**Note:** To see the results of the logging done by RACF, use the RACF report writer. For more information, see [z/OS Security Server RACF Auditor's Guide](#).

- If your installation is using some form of security classification, do one of the following:
  - If security labels are used on your system, decide which security label (if any) to assign to the profile.  
When security labels are being used on your system, be aware that security levels and categories are ignored.
  - If security levels are used on your system, decide which security level (if any) to assign to the profile.
  - If security categories are used on your system, decide which security categories (if any) to assign to the profile.
  - If your installation has written RACF installation exits to use the LEVEL operand, decide which value to specify for LEVEL.
- Depending on the class of the resource, the profile might have additional fields for which you should assign values. For example:
  - Profiles in the APPCLU class have SESSION segments
  - Profiles in the TAPEVOL class have the SINGLEDSN and TVTOC operands
  - Profiles in the TERMINAL and GTERMINL classes have the WHEN and TIMEZONE operands (both optional). WHEN determines the times and days a terminal can be used.

**Note:** This WHEN is not the same as the WHEN operand in a conditional access list.

See the appropriate topic of this document or the description of the RDEFINE command in [z/OS Security Server RACF Command Language Reference](#) for specific information on these operands.

- To copy an existing profile, specify the name of the existing profile on the FROM operand. If the existing profile is in a different class, specify FCLASS also.

2. Create the general resource profile using the RDEFINE command:

```
RDEFINE classname profile-name other-operands
```

**Note:** To change a general resource profile, use the RALTER command.

3. If specific users or groups are to have specific access to the resource, use the PERMIT command to create one or both of the access lists:

- Each entry in the standard access list states which access (such as NONE or READ) a specific user or group has:

```
PERMIT profile-name CLASS(classname)  
      ID(userid or group) ACCESS(access-authority)
```

- Each entry in the conditional access list states which access (such as NONE or READ) a specific user or group has, and also states which condition a user must meet to get the specified access:

```
PERMIT profile-name CLASS(classname)  
      ID(userid or group) ACCESS(access-authority)  
      WHEN(condition)
```

**Note:**

- a. Access authorities that you can specify with UACC or specifically assign to users vary from class to class, and are described in the topics of this document that describe the specific classes.
  - b. Not all classes are described in this document. (For example, the DSNR class is not described in this document.) Also, in some classes (notably the FACILITY class), the access required by some resource managers to specific profiles is described in the documentation of the resource manager. Therefore, go to that resource manager to find the descriptions of that class. (In the case of DSNR, which is used by Db2z/OS, see [“RACF and Db2” on page 251.](#))
  - c. The profile creator might be automatically added to the access list with ALTER authority. For more information, see [“Automatic addition of creator's user ID to access list” on page 139.](#)
4. If you have not already done so, activate the resource class:

```
SETROPTS CLASSACT(classname)
```

5. For performance benefits, consider doing one of the following:

- Allow all users on the system to have access to the resource at some level (such as READ or UPDATE) by creating a global access checking table entry that has a name similar to the new resource profile.

See [“Setting up the global access checking table” on page 194.](#)

- Reduce I/O to the RACF database by requesting that RACF keep all profiles in the class in storage:

```
SETROPTS RACLIST(classname)
```

**Note:** This is required for some classes.

## Choosing between discrete and generic profiles in general resource classes

Most general resource classes (except the PROGRAM class) give you a choice of creating either a discrete profile or a generic profile. Refer to [“Examples of defining load modules as controlled programs” on page 322](#) for more information on creating profiles in the PROGRAM class.

Choose a *generic* profile to protect more than one resource with the same security requirements.

**Note:**

1. You can use the characters \*, \*\*, or % to specify in which way (if at all) the resources protected by the profile have identical characters in their names.
2. If you use the character & in a profile name, there must be a corresponding RACFVARS profile. See [“Using RACF variables in profile names \(RACFVARS class\)”](#) on page 216.

Choose a *discrete* profile to protect one resource with unique security requirements. The name of a discrete profile has no generic characters.

## Disallowing generic profile names for general resources

You can disallow generic profiles in a resource class. For a dynamic class that you define yourself, see [“Defining a dynamic class with generics disallowed”](#) on page 272. For a static class, your system programmer can define the class in the installation CDT by executing the ICHERCDE macro using the GENERIC=DISALLOWED parameter. (See [z/OS Security Server RACF Macros and Interfaces](#) for information about using the ICHERCDE customization macro.)

Before you request a static class defined with GENERIC=DISALLOWED, determine whether the new class shares a POSIT value with other classes. If so, ensure that all other classes sharing the POSIT value also have GENERIC=DISALLOWED for static classes or GENERIC(DISALLOWED) for dynamic classes.

## Choosing among generic profiles, resource group profiles, and RACFVARS profiles

Table 14 on page 188 gives some considerations for choosing among generic profiles, resource group profiles, and RACFVARS profiles.

*Table 14. Choosing among generic profiles, resource group profiles, and RACFVARS profiles*

How to choose	Reference
Use generic profiles when the names of the resources have logically matching characters.	<a href="#">“Rules for generic profile names”</a> on page 188 and <a href="#">z/OS Security Server RACF Command Language Reference</a> .
Use resource group profiles if the names of the resources do not have logically matching characters and there is a resource grouping class (such as GTERMINL or GDASDVOL).	<a href="#">“Creating resource group profiles”</a> on page 212.
Use RACFVARS profiles if the names of the resources do not have logically matching characters and there is no resource grouping class.	<a href="#">“Using RACF variables in profile names (RACFVARS class)”</a> on page 216.

## Rules for generic profile names

There are a few rules that apply to naming generic profiles.

### When you can specify generic profile names

You can create a profile with a generic name when either of the following is true for the class of the profile:

- The SETROPTS GENERIC option is in effect. Not only does this option allow the creation of generic profiles, it also causes RACF to use generic profiles during authorization checking.
- The SETROPTS GENCMD option is in effect. In this case, generic profiles can be created and modified, but RACF does not use them during authorization checking. This is intended for use when migrating from discrete profiles to generic profiles.

## Generic naming

You can specify an asterisk (\*) within a profile name to represent one qualifier of a resource name. You can also use a double asterisk (\*\*) to represent zero or more qualifiers within a general resource generic profile or at the end of such a profile. Use of the double asterisk (\*\*) in general resource generic profiles is not controlled by the SETROPTS EGN option which applies only to the data set profiles.

Some of the rules for generic characters are different between general resource and data set generic profiles. See [z/OS Security Server RACF Command Language Reference](#) for more information.

## Other rules for generic profile names

The following rules apply to profile names:

- Valid generic characters are \*, %, and \*\*:
  - Specify % in the profile name to match any single non-blank character (except a period) in the same position of the resource name.
  - Specify \* or \*\* in the profile name to match more than one character in the same position of the resource name. For a complete description and examples of how to specify \* and \*\*, see [z/OS Security Server RACF Command Language Reference](#).
  - The %\* combination requires special attention.

New profiles with an ending %\* are not allowed nor are profiles named %\*. The RDEFINE command returns an error message.

Existing profiles with an ending %\* are usable, but they should be deleted before creating any new profiles with a middle or beginning \* or \*\*. The RALTER and RDELETE commands accept %\* to enable you to make the changes.

Instead of using an ending %\*, create new profiles ending with \* for similar function (change AB.C%\* to AB.C\*).

If you have an existing profile whose entire name is %\*, you should create a new profile whose new name is \*\*.

### Note:

1. The above considerations also apply to generic members of grouping classes.
  2. When creating the new profiles, consider using the FROM operand for continued use of the same access list.
- For any particular general resource class, the profile naming conventions are defined by how the resource name is specified on the call to RACF. When your application programmers are designing the resource names for use in their invocations of the security product, they should be aware of the problems involved with using \*, %, or & in resource names. For more information, see [z/OS Security Server RACROUTE Macro Reference](#).

As you define general resource profiles, users must observe the naming conventions for that particular class. For some classes, the naming conventions are described in this document. However, both IBM and other vendor products can issue RACROUTE REQUEST=AUTH. You must check the documentation produced for those products for authoritative information on how those products call RACF. You should gather the following information from the calling product's documentation:

- When the call to RACF is done. In other words, what user action causes the call to RACF?  
Some further questions to ask: Also, are there settings in the product that cause the call to occur? Are there installation exits that can prevent the call, or change the results of the call?
- What is the class name used on the call to RACF?
- What is the resource name used on the call to RACF? If you are using discrete profiles, this is the profile name. If you are using generic profiles, you must know how many qualifiers (portions of the name that are separated by periods) there are, and what the qualifiers mean, so that you can specify meaningful profile names.

**Note:** If you do not follow the resource naming convention established by the caller of RACF, you might create profiles that are never used. For example, if you create a discrete profile with less than the correct number of qualifiers, the profile is never used during RACF authorization checking.

- What do the access authorities (READ, UPDATE, CONTROL, ALTER) mean? Remember, these values are hierarchical (UPDATE is higher than READ, and so on), and do not necessarily mean what the English word means. For example, for terminals, READ means "allowed to logon", not "allowed to read information".

## Generic profile checking of general resources

The rules for access-authorization checking of generic profiles for general resources are similar to those for the DATASET class.

- Generic profiles are not checked unless generic profile checking is in effect for the class. To do this, issue the following command.

```
SETROPTS GENERIC(classname)
```

**Guideline:** After you activate generic profile checking for a class and define generic profiles in it, avoid deactivating generics with the NOGENERIC operand. RACF does not use your previously defined generic profiles for authorization checking while NOGENERIC is in effect.

- If the class is not active, RACF does not check for profiles. RACF returns the default return code of the class to the resource manager. For a complete description, see [“Authorization checking for RACF-protected resources” on page 718](#).
- If more than one profile covers a particular resource, RACF searches for profiles in the following order:
  - Discrete profile
  - Matching generic profiles (see [Table 15 on page 190](#))

Table 15. Sample general resource profile names in order from most specific to least specific

Profile name	Profile type	Resources being accessed			
		COPY	COPY.PAPER	COPY.PAPER.TEST	COPY.WEB.FINAL
COPY.A	Discrete				
COPY.WEB.FINAL	Discrete				X
COPY.WEB.*	Generic				X
COPY.PAPER	Discrete		X		
COPY.PAPER.TEST	Discrete			X	
COPY.PAPER.%	Generic				
COPY.PAPER.*	Generic			X	
COPY.PAPER.**	Generic		X	X	
COPY.PAPER%	Generic				
COPY.PAPER*	Generic		X	X	
COPY.PAPE%	Generic		X		
COPY.PAP*	Generic		X	X	
COPY.PRINT.*	Generic				
COPY.&X (where: &X = PAPER in RACFVARS profile)	Generic		X		
COPY.&Y (where: &Y = WEB.FINAL in RACFVARS profile)	Generic				X
COPY.%APER	Generic		X		

Table 15. Sample general resource profile names in order from most specific to least specific (continued)

Profile name	Profile type	Resources being accessed			
		COPY	COPY.PAPER	COPY.PAPER.TEST	COPY.WEB.FINAL
COPY.*.FINAL	Generic				X
COPY.*.FINAL*	Generic				X
COPY.**.FINAL	Generic				X
COPY.**.PAPER	Generic		X		
COPY.*	Generic		X	X	X
COPY.**	Generic	X	X	X	X
COPY***	Generic	X	X	X	X
**.*	Generic		X	X	X
***	Generic	X	X	X	X
*	Generic	X	X	X	X
**	Generic	X	X	X	X

To determine which profiles have the potential to protect any particular resource, use the FILTER or MASK operands on the SEARCH command to generate a list of profiles that might match the resource. For example, you might specify the user's user ID on the FILTER operand to limit the list of profiles displayed:

```
SEARCH CLASS(JESSPOOL) FILTER(**.userid.**)
```

In general, the list of profiles generated by the SEARCH command is the order in which RACF searches for a matching profile. To review the list:

1. Find all profiles that match the resource name.
2. If no profile names match, check for profile names that include an ampersand (&) (RACF variables). You must list the RACFVARS profile to determine the value of a RACF variable:

```
RLIST RACFVARS variable-name
```

Also, the SEARCH command does not list grouping profiles (such as GTERMINL) that protect the resource. To do this, use the RESGROUP operand on the RLIST command.

```
RLIST member-class resource-name RESGROUP
```

See [“Which profiles protect a particular resource?” on page 213](#).

If these methods do not find a profile, the resource is not protected.

3. If only one profile matches, it protects the resource.
4. Otherwise, find two profiles that both match the resource name. Then, compare them character by character. Where they first differ, if one has a discrete character and the other has a generic character, the one with the discrete character wins. If both have a generic character where they differ and:
  - If one has an & and the other has a %, \*, or \*\*, the & wins.
  - If one has a % and the other has an \* or \*\*, the one with % wins.
  - If one has an \* and the other has a \*\*, the one with \* wins.

#### Notes:

- There are exceptions to these guidelines. For example, the guidelines suggest that COPY.\* is more specific than COPY.\*\*.PAPER, because the \* wins over the \*\*. But, the opposite is true (see [Table 15 on page 190](#)). The SEARCH command shows these resource names in the correct order.
- The following guideline is generally true:

## General resources

Given two generic profiles that match a resource, the one whose first generic character is farther from the beginning of the name is used.

If two profile names match except for one character position, RACF examines them in the following order:

```
blank
$ (X'5B')
# (X'7B')
@ (X'7C')
A-Z
0-9
& (X'50')
%
*
```

For example, the following profile names all match in the first three character positions (A.B), and are shown in the order RACF examines them:

```
A.B
A.B.B
A.BA
A.BZ
A.B0
A.B9
A.B&X
A.B%
A.B*
```

When in doubt about the search order, create sample profiles and check the order of profile names shown by the SEARCH command.

## Generic profile performance

Your system programmer can collect and analyze performance information related to generic profiles and then specify, or ask you to specify, certain RACF options to customize how RACF processes generic profiles. For details, see [Using generic profiles in z/OS Security Server RACF System Programmer's Guide](#).

## Granting access authorities

You can grant (or deny) user or group access to a RACF-protected resource either explicitly, by assigning the specific user or group access authority with the appropriate command, or implicitly, with the universal access authority (UACC).

Each resource that you protect with RACF requires a UACC, which is the default access authority for the resource. All users in the system who are not specifically authorized in the access list of that resource profile, except users defined with the RESTRICTED attribute, can still access the resource with the authority specified by UACC (unless the UACC is NONE). These users include users not defined to RACF.

**Note:** Users with the RESTRICTED attribute can access the resource when they are specifically authorized in the access list with the sufficient authority.

If you specifically assign a user or group an access authority to a resource, the specified authority overrides the UACC specified for the resource.

Valid authorities that you can specify with UACC or specifically assign to users or groups vary from class to class, and are described in the topics of this document that describe the specific classes.

**Note:** Not all classes are described in this document. (For example, the DSNR class is not described in this document.) Also, in some classes, the access required by some resource managers to specific profiles is described in the documentation of the resource manager.

[Table 16 on page 193](#) shows additional meanings for several access authorities for general resources.



Table 16. ALTER, NONE, and CONTROL, UPDATE, and READ access authorities for general resources

Variable	Value
ALTER	<p>For discrete profiles, the specified user or group has full control over the resource and the resource profile, and can authorize other users and groups to access the resource.<sup>4</sup></p> <p>For generic profiles, only the profile owner, users with the SPECIAL attribute, and group-SPECIAL users whose groups own the profile have control over the resource profile and can authorize other users and groups to access the resource.</p> <p>For both profiles, full resource access is allowed.</p>
NONE	The specified user or group is not permitted to access the resource or list the profile.
CONTROL, UPDATE, READ	These access authorities allow listing of selected portions of the profile and grant resource access in various ways, depending on the class.

## Limiting the size of your access lists

If you need to authorize a large number of users to a resource, you must consider the limitations on the size of the access list. The access list of each profile is limited to 65,535 bytes. Each user or group you add to the access list uses 11 bytes. Therefore, the maximum number of entries is 5957. To minimize the impact of these limitations, you can create groups and add the groups, rather than the individual users, to the access list.

There are additional considerations if you must authorize many users for a resource in a class that can be processed to an in-storage profile using the SETROPTS RACLIST command or the RACROUTE REQUEST=LIST macro. A single in-storage profile is limited to 65,535 bytes. Each entry in the access list uses 9 bytes in storage. Therefore, the maximum number of access list entries is 7273, a larger number than the same profile can contain on the database. However, because the in-storage profile includes other information in addition to the access list, such as installation data, application data, and the conditional access list, the maximum number of entries in the access list might be fewer than 7273.

If you use resource member and grouping profiles, define a given member name only one time. If you define the same member name more than once, for example, in multiple grouping profiles using the ADDMEM command or in both a member profile and a grouping profile, it will be difficult to determine the resulting security attributes for that member after RACLIST processing merges the profiles. RACF also merges the access lists of each profile, making it difficult for you to determine the number of access-list entries you have used. In addition, the combined number of access-list entries might cause the profile to become too large to be processed, and RACLIST processing might fail.

## Conditional access lists for general resource profiles

You can authorize users to conditionally gain access to general resources in three ways: through port-of-entry, SMF system identifier, and application-specific criteria.

- By port of entry:

You can require that a user or a job have entered the system from a particular device when accessing general resources.

- You can require that a user be logged on to a particular terminal by specifying WHEN(TERMINAL(...)) on the PERMIT command.

The TERMINAL class must be active for this support to take effect.

- You can require that a user be logged on to a particular console by specifying WHEN(CONSOLE(...)) on the PERMIT command.

The CONSOLE class must be active for this support to take effect.

- You can require the batch job accessing the resource to have been submitted from a particular JES input device by specifying WHEN(JESINPUT(...)) on the PERMIT command.

<sup>4</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

The JESINPUT class must be active for this support to take effect.

- You can require that a user enter the system from a particular partner LU by specifying WHEN(APPSPORT(...)) on the PERMIT command.

The APPSPORT class must be active for this support to take effect.

- You can require that a user enter the system from an IP address contained in a particular network access security zone by specifying the name of the SERVAUTH profile protecting that network access security zone on the WHEN(SERVAUTH(...)) operand of the PERMIT command.

The SERVAUTH class must be active for this support to take effect.

- You can require that a user enter the system from an IP address contained in a particular network access security zone only when executing a particular program by specifying the program on the WHEN(PROGRAM) operand of the PERMIT command, and by specifying the name of the SERVAUTH profile protecting that network access security zone as the resource.

The PROGRAM and SERVAUTH classes must be active for this support to take effect.

**Note:** If an access list contains more than one condition, *any* of the conditions allows the specified access. For example, if you enter the PERMIT command with WHEN(CONSOLE(01) TERMINAL(20)) specified, you allow the access when *either* console 01 *or* terminal 20 is used.

### Example:

To ensure that an operator (or group of operators) can issue certain operator commands only when logged on at a particular console, enter:

```
PERMIT profile-name CLASS(OPERCMDS) ID(user or group) ACCESS(READ)
      WHEN(CONSOLE(console-id))
```

- By SMF system ID:

- You can require a user to access a program from a particular system by specifying WHEN(SYSID(*system-identifier*)) on the PERMIT command:

```
PERMIT profile-name CLASS(PROGRAM) ID(user or group) ACCESS(READ)
      WHEN(SYSID(system-identifier))
```

This conditional access list entry is only valid for the PROGRAM class.

See [“Program control by SMFID in BASIC or ENHANCED mode” on page 305](#) for more information.

By CRITERIA:

- A user or job can be allowed to use a resource through the use of a CRITERIA by specifying WHEN(CRITERIA(*criteria-name(criteria-value)*)) on the PERMIT command. The *criteria-name* and *criteria-value* must match the criteria-name and criteria-value passed to RACF on the RACROUTE REQUEST=FASTAUTH authorization check. The resource manager issuing the authorization check is responsible for the *criteria-name* and *criteria-value*. See the resource manager's documentation for further information. The class you specify on the PERMIT command must be RACLISTed for this support to take effect.

### Example:

To allow members of group STUDENT to SELECT from the table USER01.HOMEWORK\_GRADES in the Db2z/OS DSND subsystem when they run with the Db2z/OS role TEACHING ASSISTANT, enter:

```
PERMIT DSND.USER01.HOMEWORK_GRADES.SELECT CLASS(MDSNTB) ID(STUDENT)
      WHEN(CRITERIA(SQLROLE('TEACHING ASSISTANT')))) ACCESS(READ)
```

## Setting up the global access checking table

You can use global access checking to improve the performance of RACF authorization checking for selected resources. Global access checking should be used for *public resources* that are accessed

frequently. For example, an entry in the global access checking table can allow all users on the system to have READ access to the SYS1.HELP data set.

The global access checking table is maintained in storage and is checked early in the RACF authorization checking sequence. If an entry in the global access checking table allows the requested access to a resource, RACF performs no further authorization checking. This can avoid I/O to the RACF database to retrieve a resource profile, and can result in substantial performance improvements.

Global access checking is used for authorization processing invoked by the RACROUTE REQUEST=AUTH macro. It is not used for authorization processing invoked by the RACROUTE REQUEST=FASTAUTH macro.

## How global access checking works

When a user requests access to a resource for which a RACROUTE REQUEST=AUTH macro is issued, and global access checking is in effect for the class of the resource, RACF searches the global access checking table for a matching entry. If there is a matching entry, RACF compares the access authority requested by the user (READ, UPDATE, CONTROL, or ALTER) to the access authority associated with the resource in the global access checking table.

If the requested access is less than or equal to the authority specified in the table entry for the resource, global access checking grants the requested access immediately, without checking the profile protecting the resource. Otherwise, normal RACF authorization checking is performed. Global access checking can only permit accesses, not deny them.

Global access checking is bypassed for users who have the RESTRICTED attribute. See [“Defining restricted user IDs” on page 75](#) for more information.

**Important:** Because RACF performs global access checking before many of the other kinds of access authority checks, such as security label checking or access list checking, global access checking might allow access to a resource you are otherwise protecting. To avoid a security exposure to a sensitive resource, do not create an entry in the global access checking table for a resource that is protected by a profile containing a security level, security category, or security label. (If the security label in the profile is SYSLOW, a global access checking table entry with an access authority of READ can be created.)

## Candidates for global access checking

The following resources are candidates for *public* access and therefore for global access checking:

- SYS1.BROADCAST
- SYS1.HELP
- SYS1.PROCLIB
- ISPF/PDF libraries
- ISPF libraries (panels, skeletons, tutorial, and so forth)
- Tools libraries

**Note:** User IDs with the RESTRICTED attribute that require access to resources like these must be specifically authorized in the access list for each required resource with sufficient access authority.

## Creating global access checking table entries

To create an entry in the global access checking table:

1. Plan the entries for the global access checking table, using the following guidelines:
  - a. Identify resource profiles that are accessed frequently and for which a performance benefit is desired.
  - b. If there are resource profiles with UACC other than NONE, consider adding similar entries to the global access checking table. Using this "matched pair" approach, each entry would have the same name as a profile, and the access specified in the entry would generally match the UACC of the profile. Do *not* add a global access checking table entry if any of the following are true:

- The profile has a security level, security category, or security label (other than SYSLOW).

**Note:** If the profile has a security label of SYSLOW, the global access checking table entry can have an access of READ.

- The profile has an entry in the standard access list that is *lower* than the access level of the global access checking table entry.
- The profile has an entry in a conditional access list that is *more restrictive* than the access level of the global access checking table entry.
- The profile requests auditing of successful access attempts at or below the level specified in the corresponding global access checking table entry.

For example, if you have a data set profile PHONE.DIRECT with UACC(READ) and AUDIT(FAILURES(UPDATE)) specified, you might create a global access checking table entry for it as follows:

```
RALTER GLOBAL DATASET ADDMEM('PHONE.DIRECT'/READ)
```

However, if there are users or groups in the standard access list of profile PHONE.DIRECT with an access authority of NONE (which is lower than the UACC), do *not* create a global access checking table entry. A global access checking table entry would allow these users and groups to read the phone directory.

- c. If you have resources that are protected by a generic profile with UACC other than NONE, and others that are protected by a more specific (generic or discrete) profile that has specific access requirements such as an access list, consider adding two entries: one for the larger set of resources (with access authority equal to the UACC of the profile) and the other for the smaller set of resources (with access authority of NONE).

For example, if you have a profile of SYS1.\* with a UACC(READ), but you also have some specific profiles with more restrictive entries, such as SYS1.XYZ with UACC(READ) and an access list with JOE/NONE, create two entries:

```
SYS1.XYZ/NONE
SYS1.**/READ
```

The entry with /NONE does not fail any attempts but stops requests for SYS1.XYZ from being granted by the SYS1.\* entry.

See the examples later in this topic for other possible entries.

2. Add the resource class to the global access checking table using the RDEFINE command with the GLOBAL operand and the class name:

```
RDEFINE GLOBAL classname
```

3. To allow global access checking for a specific resource, add an entry to the global access checking table using the RALTER command as follows:

```
RALTER GLOBAL classname ADDMEM(resource-name/access-level)
```

where:

***resource-name***

is the equivalent of a profile name in the class specified. If generic command processing is in effect for *classname* (through the SETROPTS GENCMD command), *resource-name* on the ADDMEM operand can include the generic characters \*, \*\*, or %. In general, the rules for specifying these characters are the same as the rules for specifying these characters in generic profile names except that generic characters are allowed in any qualifier (even if not allowed in certain qualifiers of the profile names).

After they have been added, generic entries in the global access checking table are used in global access checking even if generic profile checking is turned off (through the SETROPTS NOGENERIC command).

The resource name can include the name qualifiers &RACUID (RACF user ID) or &RACGPID (*current connect group*). For example, the following entry allows users to have ALTER access to data sets that begin with their own user IDs.

```
RALTER GLOBAL DATASET ADDMEM('&RACUID.**'/ALTER)
```

**Note:** This entry does not *change* a user's access to his or her own data sets, but speeds the process by which RACF grants the access. (It also prevents any auditing of such access attempts.)

The resource name can also include variables defined in the RACFVARS class. Note that when defining a resource name for a profile in a mixed-case class, you must enter the character strings &RACUID, &RACGPID, and any RACFVARS variable names in upper case. For more information on RACFVARS usage, see [“Using RACFVARS with mixed-case classes”](#) on page 220.

The qualifier &RACGPID allows the user's current connect group to be used in the same way. For example, the following allows all users to have READ access to group data sets for their current connect group:

```
RALTER GLOBAL DATASET ADDMEM('&RACGPID.**'/READ)
```

**Note:** If the current connect group is found in the global access table and list-of-groups processing is in effect, list-of-groups checking is ignored.

#### **access-level**

can be NONE, READ, UPDATE, CONTROL, or ALTER.

For more information on specifying the ADDMEM operand, see [z/OS Security Server RACF Command Language Reference](#).

4. When you are finished updating the global access checking table, issue the SETROPTS command with the GLOBAL operand for each class affected.

```
SETROPTS GLOBAL(classname)
```

#### **Guidelines:**

- Save a listing of the global access checking table. This can help you recover from the accidental deletion or alteration of the global access checking table or its entries. You can use the RLIST command to make this listing quickly.
  - Write an EXEC that contains the commands you use to create the global access checking table. The EXEC should include the RLIST command to provide an independent record of the actual table created. Also, if the global access checking table is accidentally deleted (using the RDELETE command), the EXEC can readily be used to regenerate the table.
5. **Important:** For each entry in the global access checking table, create a similar resource profile. Such a "matched pair" approach can help ensure the continuation of protection if global access checking becomes disabled. For example:

```
RDEFINE classname resource-name UACC(access-level)
```

At the end-of-volume (EOV) processing, RACROUTE REQUEST=AUTH is issued with OLDVOL specified for authority checking with the DATASET and TAPEVOL classes. This check bypasses the global access table checking and uses resource profile definitions for authority checking. By not having a "matched pair" approach, you might get different results.

## **Adding an entry to the global access checking table**

To add an entry to the global access checking table, issue the RALTER command with the ADDMEM operand, then refresh the in-storage global access checking table. For example:

```
RALTER GLOBAL classname ADDMEM(resource-name/level)
SETROPTS GLOBAL(classname) REFRESH
```

Deleting an entry from the global access checking table

To delete an entry from the global access checking table, issue the RALTER command with the DELMEM operand, then refresh the in-storage global access checking table. For example:

```
RALTER  GLOBAL classname  DELMEM(resource-name/level)
SETROPTS GLOBAL(classname) REFRESH
```

**Important:** Do not use the RDELETE command unless you intend to delete the entire global access checking table for that class.

Examples of creating global access checking table entries

The following examples show you how to create entries in the global access checking table for:

- The SYS1.HELP data set
- The SYS1.BROADCAST data set
- Group data sets
- The master catalog and user catalogs

Example 1: The SYS1.HELP data set

To allow all users to have READ access to SYS1.HELP, enter:

```
SETROPTS GLOBAL(DATASET)
RDEFINE  GLOBAL DATASET
RALTER   GLOBAL DATASET ADDMEM('SYS1.HELP'/READ)
ADDSD    'SYS1.HELP' UACC(READ)
SETROPTS GLOBAL(DATASET) REFRESH
```

Example 2: Group data sets

To specify that all users are to have UPDATE access authority to data sets whose high-level qualifier is the user's current connect group, enter:

```
RDEFINE GLOBAL DATASET ADDMEM('&RACGPID.**'/UPDATE)
```

**Note:** The &RACGPID entries provide more flexibility than the GRPACC attribute. [Table 17 on page 198](#) shows some points of comparison.

Table 17. Comparison of GRPACC attribute with &RACGPID.\*\* entry in global access checking table

GRPACC attribute	&RACGPID.** entry
Applies only to group data set profiles created by the user while the user has the GRPACC attribute.	Applies to all group data sets for all users.
Always allows UPDATE access.	Can allow READ, UPDATE, CONTROL, or ALTER access.
Works by changing access lists in group data set profiles. These can be changed individually later.	Works the same way for all users in all connect groups. Changes affect all users.
Applies only to resources in the DATASET class.	Can apply to any class that might include a group name in profile names (such as TAPEVOL).

Example 3: The master catalog and user catalogs

With the exception of a very select group, users should only be allowed to READ the master catalog. To allow this, enter:

```
RALTER GLOBAL DATASET ADDMEM('CATALOG.MASTER.**'/READ)
ADDSD   'CATALOG.MASTER.**' UACC(READ)
PERMIT  'CATALOG.MASTER.**' ID(SYSGROUP) ACCESS(CONTROL)
```

**Note:**

1. The exact form of the names specified on these commands depends on the naming conventions at your installation. This example assumes that catalog names take the form:

```
CATALOG.MASTER.MVSESA.Vvvvvvv for a master catalog
and
CATALOG.applc.Vvvvvvv for application-specific catalogs
(for example, TSO or Db2)
```

2. The access authority that is required to update and maintain the catalog depends on the DFP release that is installed on your system.

For user catalogs, most users should be allowed to add entries as they create data sets. To allow this, enter:

```
RALTER GLOBAL DATASET ADDMEM('CATALOG.**'/UPDATE)
ADDSD 'CATALOG.**' UACC(UPDATE)
```

## Stopping global access checking for a specific class

To stop global access checking for a specific class, issue:

```
SETROPTS NOGLOBAL(classname)
```

## Listing the global access checking table

To list the global access checking table, do one of the following:

- For a list showing the entries in a particular class, enter the following command.

```
RLIST GLOBAL classname
```

This shows the entries in the order in which they are searched by RACF.

- See the DSMON report that lists the global access checking table described in [z/OS Security Server RACF Auditor's Guide](#) for a list that shows all entries in the table.

## Special considerations for global access checking

When using global access checking, consider the following:

- Global access checking is used for authorization processing invoked by the RACROUTE REQUEST=AUTH macro. It is not used for authorization processing invoked by the RACROUTE REQUEST=FASTAUTH macro.
- Global access checking is bypassed for access requests by users with the RESTRICTED attribute. See [“Defining restricted user IDs” on page 75](#).
- RACF authorization checking via RACROUTE REQUEST=AUTH searches the global access checking table for a matching entry, ignoring profiles in the class. If no global access checking table entry matches the search, or if the access specified in the entry is less than the access being requested, RACF then searches for a matching profile in the class. This processing occurs regardless of whether or not the class is RACLISTed (by either SETROPTS RACLIST or RACROUTE REQUEST=LIST).
- RACF searches the global access checking table for an entry that best matches the name of the resource, much as RACF searches for a matching profile. The output from the RLIST command shows the order used.
- The group resource classes (such as GTERMINL) are ineligible for global access checking.
- When global access checking allows a request to access a data set, that data set is considered to be protected by RACF, and therefore any OS password processing and prompting that would otherwise have occurred is bypassed.
- When global access checking allows a request, RACF maintains no statistics.



- When global access checking allows a request, RACF performs no logging other than that requested by the SETROPTS LOGOPTIONS command.
- RACF bypasses global access checking if the PROFILE, CSA, or PRIVATE operand is specified on the request for RACF authorization checking (RACROUTE REQUEST=AUTH).
- Updated global access checking table entries become effective with the next IPL or after execution of the SETROPTS command with the GLOBAL(*classname*) operand (with or without the REFRESH operand).
- The only use for an access of NONE in the global access table is to force RACF to look for a profile. This would typically be used when you have access list entries which have a lower access level than a data set's UACC, or when you want to ensure that auditing or security classification checking takes place for a specific data set.
- When RACF is enabled for sysplex communication, the SETROPTS GLOBAL and SETROPTS GLOBAL(*classname*) REFRESH commands are propagated to the other members of the sysplex data sharing group.
- A global access table entry for JESSPOOL suppresses logging based on the AUDIT options set in the resource profile. However, this entry might or might not suppress other types of logging, depending on the application accessing the resource and details of the application's design.

For example, you might define a global access table entry for JESSPOOL containing the ADDMEM operand with the &RACUID value in the second qualifier to allow user's to access to their own spool data sets without logging. However, RACF might log accesses depending on the application that users use to access their spool data sets.

## Field-level access checking

---

You can use RACF to control which users can access data in RACF profiles at the field level through *field-level access checking*. To do this, you create profiles in the FIELD class and permit users to the profiles.

Using field-level access checking, you can:

- Allow a user or group to modify a particular field (or segment) in all profiles of a particular type. For example, you can define a profile to control access to the ACCTNUM field of the TSO segment of user profiles. If you give a user UPDATE authority to this profile, the user can modify the ACCTNUM field in all user profiles.
- Allow all users to read or modify a particular field (or segment) of their own user profiles. To do this, specify ID(&RACUID) on the PERMIT command.
- Allow a user to modify or list a particular field (or segment) only in profiles that they have RACF command processor authority to modify BASE segment data.

You need not use field-level access checking to authorize READ access for users with the SPECIAL, AUDITOR, or ROAUDIT attribute. These users are authorized to list all fields of all segments for any RACF profile.

**Note:** RACF command processors and panels support field-level access checking only for fields in segments other than the base segments of RACF profiles. However, the ICHEINTY and RACROUTE REQUEST=EXTRACT macros can support field-level access checking for fields in any segment of any RACF profile. If your installation has written its own programs that use these macros to access the RACF database, you can modify these programs to implement field-level access checking.

To use field-level access checking, perform the following steps:

1. Define profiles in the FIELD class:

```
RDEFINE FIELD profile-name UACC(NONE)
```

where *profile-name* has the following format:

```
profile-type.segment-name.field-name
```



where:

***profile-type***

is one of the following:

- DATASET for data set profiles
- GROUP for group profiles
- USER for user profiles
- *class-name* for general resource profiles

***segment-name***

is one of the following:

- BASE for BASE segments (this is supported only by user-written code)
- CDTINFO for CDTINFO segments
- CFDEF for CFDEF segments
- CICS for CICS segments
- CSDATA for CSDATA segments
- DCE for DCE segments
- DFP for DFP segments
- DLFDATA for DLFDATA segments
- EIM for EIM segments
- ICSF for ICSF segments
- ICTX for ICTX segments
- IDTPARMS for IDTPARMS segments
- JES for JES segments
- KERB for KERB segments
- LANGUAGE for LANGUAGE segments
- LNOTES for LNOTES segments
- MFA for MFA segments
- MFPOLICY for MFPOLICY segments
- NDS for NDS segments
- NETVIEW for NETVIEW segments
- OMVS for OMVS segments
- OPERPARM for OPERPARM segments
- PROXY for PROXY segments
- OVM for OVM segments
- SESSION for SESSION segments
- SIGVER for SIGVER segments
- SSIGNON for SSIGNON segments
- STDATA for STDATA segments
- SVFMR for SystemView segments
- TME for TME segments
- TSO for TSO segments
- WORKATTR for WORKATTR segments

**Note:** This is also the operand that is used on RACF commands to work with the segment.

***field-name***

is the name associated with the field in the RACF profile segment to be controlled.

Each field is administered by a RACF command operand. To find the field name that corresponds to a command operand, see [Table 18 on page 205](#).

**Example:** To control access to all fields in the TSO segment of all user profiles, issue the RDEFINE command and specify USER.TSO.\* as the profile name. Before issuing this command, however, see the following note.

```
RDEFINE FIELD USER.TSO.* UACC(NONE)
```

**Note:** The profile name USER.TSO.\* is a generic profile name. Before you issue the RDEFINE command, generic profile checking for the FIELD class must be active. If it is not active, issue the SETROPTS GENERIC(FIELD) command before you define the generic profile.

When you specify a UACC of NONE, you prevent all users from accessing the TSO segment in all user profiles, including their own. Likewise, if you specify a UACC of READ, you allow all users to read the information contained in all fields of the TSO segment for all user profiles.

To control access to a specific field in the TSO segment of user profiles, issue the RDEFINE command and specify the name that is associated with the field as the third qualifier in the profile name.

**Example:** Based on [Table 18 on page 205](#), to control access to the ACCTNUM field, create a profile specifying TACCNT as the *field-name* qualifier:

```
RDEFINE FIELD USER.TSO.TACCNT UACC(NONE)
```

**Note:** A user with UPDATE access to this profile is authorized to change the account number field in a TSO segment by specifying the ACCTNUM operand on the TSO option of the ALTUSER command:

```
ALTUSER userid TSO(ACCTNUM(account-number))
```

2. Allow specific users or groups to have the appropriate access to the field. For example:

```
PERMIT USER.TSO.TLPROC CLASS(FIELD) ID(TSOADM) ACCESS(UPDATE)
```

This example shows how to authorize user ID TSOADM to change the logon procedure (TLPROC field) in the profiles of all TSO users.

**Note:** You can also specify the value &RACUID with the ID operand on the PERMIT command for FIELD profiles. When you enter this value on the PERMIT command, you allow all users access to the specified field or segment of their own user profiles. For example, if you issue the following command, you allow all users to read the TLPROC field in the TSO segment of their own user profiles.

```
PERMIT USER.TSO.TLPROC CLASS(FIELD) ID(&RACUID) ACCESS(READ)
```

3. When you are ready to start using the protection defined in the profiles, activate the FIELD class:

```
SETROPTS CLASSACT(FIELD)
```

**Note:** If you do not activate the FIELD class and you activate SETROPTS RACLIST processing for the FIELD class, only SPECIAL users can access fields in segments (other than the base segment) of RACF profiles.

4. You must activate SETROPTS RACLIST processing for the FIELD general resource class. For a complete description of this function, see [“SETROPTS RACLIST processing” on page 127](#).

```
SETROPTS RACLIST(FIELD)
```

**Note:** After you activate SETROPTS RACLIST processing for the FIELD class, when ever you change a FIELD profile, you must refresh SETROPTS RACLIST processing for the FIELD class for the change to take effect.

```
SETOPTS RACLIST(FIELD) REFRESH
```

A user with access to a FIELD class profile for a given segment or field can manipulate that field in all profiles. Field level access can be optionally restricted such that access to a particular segment or field, as granted by FIELD profiles, is limited to profiles to which the user has BASE-segment access. BASE-segment access is obtained by way of profile ownership, group-special, or other means, as determined by the RACF command being processed.

To activate the optional BASE-segment authority requirement to field-level access checking, define a new profile in the FIELD class.

```
RDEFINE FIELD FLAC.SKIP.BASECHECK UACC(NONE)
```

If the FLAC . SKIP . BASECHECK profile exists, and a command-issuing user lacks READ access to it, the field level access is granted only if the user performing the profile operation has BASE-segment access as well as authorization to the appropriate FIELD class profile. The ability to list or modify the DATA('data') field of a profile can be used as an indicator of having sufficient BASE-segment access.

### Example:

GSADM1 already has GROUP-SPECIAL authority to the group, which is the OWNER of user2 and has no administrative authority over user1. Set up field level access checking on GSADM1 to manage the TSO segment for user2, but not user1.

```
SETOPTS CLASSACT(FIELD) GENERIC(FIELD) RACLIST(FIELD)
RDEFINE FIELD USER.TSO.* UACC(NONE)
RDEFINE FIELD FLAC.SKIP.BASECHECK UACC(NONE)
PERMIT USER.TSO.* CLASS(FIELD) ID(GSADM1) ACCESS(UPDATE)
SETOPTS GENERIC(FIELD) RACLIST(FIELD) REFRESH
(from GSADM1)
ALTUSER USER2 TSO(PROC(OMVSPROC))
(success)
ALTUSER USER1 TSO(PROC(OMVSPROC))

IRR52127I Field level access checking failed for segment segment-name.
ICH51012I RACF AUTHORITY DENIED BY FIELD LEVEL ACCESS CHECKING.
ICH21004I {userid | DFLTGRP | OWNER | USER} NOT ALTERED.
```

There are two ways to set up the FLAC . SKIP . BASECHECK profile. If the profile is defined with UACC (READ), field-level-access checking processes without taking BASE-segment authorization into consideration. Namely, anyone with FIELD access to a particular field can access that field for all profiles that are defined to RACF. If there is a need to scope the administrative abilities of selected administrators to access non-BASE fields based on profile ownership or group-special, the administrator's access should be specified with the value NONE using the PERMIT command.

Alternatively, the FLAC . SKIP . BASECHECK profile can be given UACC (NONE). This immediately limits all use of field-level-access to users who have BASE-segment access in accordance to the profile manipulation rules as specified by the command processors. Users who require system-wide access to non-BASE fields, should be given READ access to FLAC . SKIP . BASECHECK using the PERMIT command.

Use of the FLAC . SKIP . BASECHECK option is only compatible with the following RACF command processors.

```
ADDUSER
ALTUSER
LISTUSER
ADDGROUP
ALTGROUP
LISTGROUP
RDEFINE
RALTER
RLIST
ADDSD
```

ALTDSD  
LISTSDS

Other programs that use ICHEINTY or RACROUTE REQUEST=EXTRACT to manipulate non-base segments and fields behave differently than the commands previously listed because the RACF command processors listed are the final arbiters of whether a user has access to manipulate the BASE-segment of a profile. Additional programs, such as those using the ICHEINTY or RACROUTE REQUEST=EXTRACT interfaces, cannot make this determination.

If a user-defined program uses ICHEINTY or RACROUTE to manipulate non-BASE segment and other data, and the `FLAC . SKIP . BASECHECK` profile is defined, users with READ access to `FLAC . SKIP . BASECHECK` execute successfully after considering the user's access to profiles in the FIELD class. Users with access of NONE to `FLAC . SKIP . BASECHECK` will fail to manipulate all non-BASE segment and field information, even if they are allowed to perform the same operations using the RACF commands. If the `FLAC . SKIP . BASECHECK` profile is not defined, a call to ICHEINTY is executed as if the user has READ access to `FLAC . SKIP . BASECHECK`.

Users with access of NONE to `FLAC . SKIP . BASECHECK` are still able to alter the fields in their own user profiles that have UPDATE permission that is granted to &RACUID. The new scoping rules in effect due to having NONE access to `FLAC . SKIP . BASECHECK` do not apply when using ALTUSER to alter an individual's own profile and when &RACUID is on the access list of the fields being updated.

Users with access of NONE to `FLAC . SKIP . BASECHECK` are still able to list those fields in their own user profiles to which they have been granted READ permission. The new scoping rules in effect due to having NONE access to `FLAC . SKIP . BASECHECK` do not apply when using LISTUSER to list an individual's own profile.

Table 18. Fields in RACF segments that correspond to RACF command operands

To control the use of this operand: <sup>1</sup>	Specify this value as the <i>field-name</i> qualifier:
<b>CDTINFO segment in general resource profiles (CDT class):</b>	
CASE	CDTCASE
DEFAULTRC	CDTDFTRC
DEFAULTUACC	CDTUACC
FIRST	CDTFIRST
GENERIC	CDTGEN
GENLIST	CDTGENL
GROUP	CDTGROUP
KEYQUALIFIERS	CDTKEYQL
MACPROCESSING	CDTMAC
MAXLENGTH	CDTMAXLN
MAXLENX	CDTMAXLX
MEMBER	CDTMEMBR
OPERATIONS	CDTOPER
OTHER	CDTOTHER
POSIT	CDTPOSIT
PROFILESALLOWED	CDTPRFAL
RACLIST	CDTRACL
SECLABELSREQUIRED	CDTSLREQ
SIGNAL	CDTSIGL
<b>CFDEF segment in general resource profiles (CFIELD class):</b>	
ACEE	CFACEE
TYPE	CFDTYPE
MAXLENGTH	CFMXLEN
MAXVALUE	CFMXVAL
MINVALUE	CFMNVAL
FIRST	CFFIRST
OTHER	CFOTHER
MIXED	CFMIXED
HELP	CFHELP
LISTHEAD	CFLIST
VALREXX	CFVALRX
<b>CICS segment in user profiles:</b>	
OPCLASS	OPCLASS and OPCLASSN <sup>2</sup>
OPIDENT	OPIDENT
OPPRTY	OPPRTY
RSLKEY	RSLKEY and RSLKEYN <sup>2</sup>
TIMEOUT	TIMEOUT
TSLKEY	TSLKEY and TSLKEYN <sup>2</sup>
XRFSOFF	XRFSOFF
<b>CSDATA segment in user and group profiles:</b>	
<i>custom-field-name</i>	<i>custom-field-name</i>
<b>DCE segment in user profiles:</b>	

Table 18. Fields in RACF segments that correspond to RACF command operands (continued)

To control the use of this operand: <sup>1</sup>	Specify this value as the <i>field-name</i> qualifier:
AUTOLOGIN DCENAME HOMECCELL HOMEUUID UUID	DCEFLAGS DCENAME HOMECCELL HOMEUUID UUID
<b>DFP segment in data set profiles:</b>	
RESOWNER DATAKEY ENCRYPTTYPES	RESOWNER DATAKEY ENCTYPES
<b>DFP segment in user and group profiles:</b>	
DATAAPPL DATACLAS MGMTCLAS STORCLAS	DATAAPPL DATACLAS MGMTCLAS STORCLAS
<b>DLFDATA segment in DLFCLASS class profiles:</b>	
RETAIN JOBNAMEs	RETAIN JOBNAMEs and JOBNMCNT <sup>2</sup>
<b>EIM segment in user profiles:</b>	
LDAPPROF	LDAPPROF
<b>EIM segment in FACILITY and LDAPBIND class profiles:</b>	
DOMAINDN KERBREGISTRY LOCALREGISTRY OPTIONS X509REGISTRY	DOMAINDN KERBREG LOCALREG OPTIONS X509REG
<b>ICSF segment in CSFKEYS, GCSFKEYS, XCSFKEY, and GXCSFKEY class profiles:</b>	
ASYMUSAGE SYMEXPORTABLE SYMEXPORTCERTS SYMEXPORTKEYS SYMCPACFWRAP SYMCPACFRET	CSFAUSE CSFSEXP CSFSCLBS and CSFSCLCT <sup>2</sup> CSFSKLBS and CSFSKLCT <sup>2</sup> CSFSCPW CSFSCPR
<b>ICTX segment in LDAPBIND class profiles:</b>	
USEMAP DOMAP MAPREQUIRED MAPPINGTIMEOUT	USEMAP DOMAP MAPREQ MAPTIMEO
<b>IDTPARMS segment in IDTDATA class profiles:</b>	

Table 18. Fields in RACF segments that correspond to RACF command operands (continued)

To control the use of this operand: <sup>1</sup>	Specify this value as the <i>field-name</i> qualifier:
SIGTOKEN	IDTTOKN
SIGSEQNUM	IDTSEQN
SIGCAT	IDTCAT
SIGALG	IDTSALG
SIGLABELPRIMARY	IDTLABP
SIGKIDPRIMARY	IDTKIDP
IDTTIMEOUT	IDTTIMEO
ANYAPPL	IDTANYAP
PROTALLOWED	IDTPROTA
<b>JES segment in JESJOBS class profiles:</b>	
KEYLABEL	KEYLABEL
<b>KERB segment in user profiles:</b>	
ENCRYPT	ENCRYPT
KERBNAME	KERBNAME
MAXTKTLFE	MAXTKTLFE
<b>KERB segment in REALM class profiles:</b>	
CHECKADDRS	CHKADDRS
DEFTKTLFE	DEFTKTLFE
ENCRYPT	ENCRYPT
KERBNAME	KERBNAME
MAXTKTLFE	MAXTKTLFE
MINTKTLFE	MINTKTLFE
<b>LANGUAGE segment in user profiles:</b>	
PRIMARY	USERNL1
SECONDARY	USERNL2
<b>LNOTES segment in user profiles:</b>	
SNAME	SNAME
<b>MFA segment in MFADEF class profiles:</b>	
MFA	MFDATA
<b>MFPOLICY segment in MFADEF class profiles:</b>	
FACTORS	MFFCTRS and MFFCTRN <sup>2</sup>
TOKENTIMEOUT	MFTIMEO
REUSE	MFREUSE
<b>NDS segment in user profiles:</b>	
UNAME	UNAME
<b>NETVIEW segment in user profiles:</b>	

Table 18. Fields in RACF segments that correspond to RACF command operands (continued)

To control the use of this operand: <sup>1</sup>	Specify this value as the <i>field-name</i> qualifier:
IC	IC
CONSNAME	CONSNAME
CTL	CTL
MSGRECV	MSGRECV
OPCLASS	OPCLASS and OPCLASSN <sup>2</sup>
DOMAINS	DOMAINS and DOMAINSN <sup>2</sup>
NGMFADMN	NGMFADMN
NGMFVSPN	NGMFVSPN
<b>OMVS segment in group profiles:</b>	
GID	GID
<b>OMVS segment in user profiles:</b>	
ASSIZEMAX	ASSIZE
CPUTIMEMAX	CPUTIME
FILEPROC	FILEPROC
HOME	HOME
MEMLIMIT	MEMLIMIT
MMAPEAREMAX	MMAPEARE
PROCUSERMAX	PROCUSER
PROGRAM	PROGRAM
SHMEMMAX	SHMEMMAX
THREADSMAX	THREADS
UID	UID
<b>OPERPARM segment in user profiles:</b>	
ALTGRP <sup>3</sup>	OPERALTG
AUTH	OPERAUTH
AUTO	OPERAUTO
CMDSYS	OPERCMDS
DOM	OPERDOM
KEY	OPERKEY
HC	OPERHC
INTIDS	OPERINT
LEVEL	OPERLEVL
LOGCMDRESP	OPERLOGC
MFORM	OPERMFRM
MIGID <sup>3</sup>	OPERMGID
MONITOR	OPERMON
MSCOPE	OPERMSCP and OPERMCNT <sup>2</sup>
ROUTCODE	OPERROUT
STORAGE	OPERSTOR
UD <sup>3</sup>	OPERUD
UNKNIDS	OPERUNKN
<b>OVM segment in group profiles:</b>	
GID	GID
<b>OVM segment in user profiles:</b>	



Table 18. Fields in RACF segments that correspond to RACF command operands (continued)

To control the use of this operand: <sup>1</sup>	Specify this value as the <i>field-name</i> qualifier:
FSROOT HOME PROGRAM UID	FSROOT HOME PROGRAM UID
<b>PROXY segment in user and FACILITY class profiles:</b>	
BINDDN LDAPHOST	BINDDN LDAPHOST
<b>SESSION segment in APPCLU class profiles:</b>	
CONVSEC INTERVAL LOCK SESSKEY	CONVSEC KEYINTVL SLSFLAGS SESSKEY
<b>SIGVER segment in PROGRAM class profiles:</b>	
SIGREQUIRED FAILLOAD SIGAUDIT	SIGREQD FAILLOAD SIGAUDIT <sup>4</sup>
<b>SSIGNON segment in PTKTDATA class profiles:</b>	
KEYENCRYPTED KEYMASKED ENCRYPTKEY KEYLABEL NOLEGACYKEY EPTKEYLABEL TYPE TIMEOUT REPLAY	SSKEY SSKEY SSKEY SSKEY SSKEY PTKEYLAB PTTYPE PTTIMEO PTREPLAY
<b>STDATA segment in STARTED class profiles:</b>	
USER GROUP PRIVILEGED TRACE TRUSTED	STUSER STGROUP FLAGPRIV FLAGTRAC FLAGTRUS
<b>SVFMR segment in SYSMVIEW class profiles:</b>	
PARMNAME SCRIPTNAME	PARMN SCRIPTN
<b>TME segment in group and data set profiles:</b>	
ROLES	ROLES and ROLEN <sup>2</sup>

Table 18. Fields in RACF segments that correspond to RACF command operands (continued)

To control the use of this operand: <sup>1</sup>	Specify this value as the <i>field-name</i> qualifier:
<b>TME segment in general resource profiles:</b>	
ROLES	ROLES and ROLEN <sup>2</sup>
GROUPS	GROUPS and GROUPN <sup>2</sup>
RESOURCE	RESOURCE and RESN <sup>2</sup>
CHILDREN	CHILDREN and CHILDN <sup>2</sup>
PARENT	PARENT
<b>TSO segment in user profiles:</b>	
ACCTNUM	TACCNT
COMMAND	TCOMMAND
DEST	TDEST
HOLDCLASS	THCLASS
JOBCLASS	TJCLASS
PROC	TLPROC
MAXSIZE	TMSIZE
MSGCLASS	TMCLASS
SECLABEL	TSOSLABL
SIZE	TLSIZE
SYS	TSCLASS
UNIT	TUNIT
USERDATA	TUDATA
<b>WORKATTR segment in user profiles:</b>	
WANAME	WANAME
WABLDG	WABLDG
WADEPT	WADEPT
WAROOM	WAROOM
WAADDR1	WAADDR1
WAADDR2	WAADDR2
WAADDR3	WAADDR3
WAADDR4	WAADDR4
WAACCNT	WAACCNT
WAEMAIL	WAEMAIL

**Note:**

- Many operands in this table have corresponding versions that include a prefix of NO. In addition, several operands have corresponding versions that include prefixes of ADD and DEL. See the [z/OS Security Server RACF Command Language Reference](#) to identify these.
- For operands that are listed with two *field-name* qualifiers:
  - To authorize READ access, define one FIELD profile specifying the first value as the *field-name* qualifier. Permit users READ access.
  - To authorize UPDATE access, define two FIELD profiles. Define one profile for each of the two *field-name* qualifiers listed. Permit users UPDATE access to both profiles.
- This setting is ignored when each system sharing the RACF database runs z/OS Version 1 Release 8 or higher.
- The SIGAUDIT field controls the audit policy related to digital signature verification of programs. Users with the AUDITOR or ROAUDIT attribute can list the SIGAUDIT field but they cannot update it unless they have UPDATE authority through field-level access checking.

## Planning for profiles in the FACILITY class

The FACILITY class can be used for a wide variety of purposes depending on the products installed on your system. If the FACILITY class is active, users might need access to particular resources to perform specific tasks. Therefore, they must have access based on the profiles protecting those resources. For example:

- READ access to IEAVECTOR allows users to use the vector facility.
- READ access to ICHBLP allows tape users to bypass label processing.
- READ access to IEC.TAPERING allows tape users to write to tape data sets without removing the write-enable ring.
- There are many other resources that can be protected using RACF profiles in the FACILITY class for use with many different subsystems and products.

You should activate the FACILITY class for the first such profile that is required on your system. You can create FACILITY profiles as needed to control who can use a number of processes on your system.

**Guideline:** Activate SETROPTS RACLIST processing for the FACILITY general resource class. When you activate this function, you improve performance because I/O to the RACF database is reduced. For a complete description of this function, see [“SETROPTS RACLIST processing” on page 127](#).

```
SETROPTS RACLIST(FACILITY)
```

If you activate SETROPTS RACLIST processing for the FACILITY class, any time you make a change to a FACILITY profile, you must also refresh SETROPTS RACLIST processing for the FACILITY class for the change to take effect.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

## Delegating help desk functions

For information about using FACILITY class profiles to delegate help desk functions, such as the authority to list user information and reset passwords, see [Chapter 27, “Authorizing help desk functions,” on page 655](#).

## Delegating authority to profiles in the FACILITY class

You can use several methods to allow another user, such as a tape librarian or storage administrator, to work with profiles in the FACILITY class:

- Assign the user as OWNER of all of the FACILITY profiles used by the function.
- Create a group representing the function and give the user group-SPECIAL authority within the group. Then assign the group as OWNER of the FACILITY profiles used by the group.
- If the SETROPTS GENERICOWNER option is in effect, give the user CLAUTH(FACILITY), create a *top* generic profile to which the user is assigned as OWNER. The SETROPTS GENERICOWNER option limits this user to creating FACILITY profiles that are more specific than the *top* generic profile.

**Guideline:** Do *not* create a top profile named \*\* in the FACILITY class, as this could lead to problems with RJE.

For more information about the GENERICOWNER option, see [“Restricting the creation of general resource profiles \(GENERICOWNER and ENHANCEDGENERICOWNER options\)” on page 113](#).

For other examples for delegating authority in the FACILITY class, see the topics shown in [Table 19 on page 212](#).

Table 19. Delegating authority in the FACILITY class

Situation	Topic
To allow users to obtain dumps when they are using programs to which they only have EXECUTE authority, using the IEAABD.DMPAUTH resource	<a href="#">“Protecting program dumps using the FACILITY class” on page 230</a>
To allow users to open tape data sets for input without removing the write-enable ring (or equivalent), using the IEC.TAPERING resource	<a href="#">“IEC.TAPERING profile in the FACILITY class” on page 180</a>
To allow users to access DFP-controlled DASD or tape data sets when those data sets are neither cataloged nor system temporary data sets, using ICHUNCAT. <i>data-set-name</i> and CATDSNS	<a href="#">“Preventing access to uncataloged data sets (CATDSNS option)” on page 120</a>
To allow migration of security functions from JES into RACF, using the RJE, RJP, and NJE NODES profiles	<a href="#">“Understanding NODES profiles” on page 469</a>

## Creating resource group profiles

Like generic profiles, *resource group* class profiles enable you to protect multiple resources with one profile. However, the resources need not have similar names.

A resource group profile is a general resource profile with the following special characteristics:

- Its name does not match the resources it protects.
- The ADDMEM operand (not the profile name itself) specifies the resources it protects.
- Its class is a resource group class or *grouping* class (for example, GTERMINL or GDASDVOL).
- The related member class (not the resource group class itself) *must* be RACLISTed. For example, the TERMINAL class must be RACLISTed, not the GTERMINL class. Depending on the class, RACLISTing is accomplished using the SETROPTS command or RACROUTE REQUEST=LIST.

**Example:** The following command protects three terminals that have *unlike* names, M01RF267, M03RF168, and M04GG148:

```
RDEFINE GTERMINL DEPT35 UACC(NONE) ADDMEM(M01RF267 M03RF168 M04GG148)
```

Several resource group classes and related member classes are supplied with RACF. Each one is marked in Appendix A, “[Supplied RACF resource classes,” on page 683](#) as a member class or a grouping class, as appropriate.

**Restriction:** Certain member classes listed in Appendix A, “[Supplied RACF resource classes,” on page 683](#) cannot be used with RACF commands because they are associated with resource grouping classes that have special uses. These classes are marked with this restriction.

To use resource group profiles, perform the following steps (terminals are used as a readily understood example):

1. Create the resource group profile:

```
RDEFINE GTERMINL profile-name UACC(NONE)
      ADDMEM(resource-name-with-or-without-generic-character...)
```

where:

**GTERMINL**

is the resource group class for terminals.

***profile-name***

is a discrete profile name of your choice (generic characters are not allowed).

**resource-name...**

is the name of the resource to be protected, for example, a terminal ID or DASD volume serial number. If you first activate generic profile checking for the related member class, you can include a generic character (\*, \*\*, or %) in the resource name.

2. Grant the appropriate access to the appropriate users and groups. In the following example, READ access is given to users in group GROUPA:

```
PERMIT DEPT35 CLASS(GTERMINL) ID(GROUPA) ACCESS(READ)
```

3. When you are ready to start using the protection defined in the profiles, activate the *member class*. For classes other than the CICS<sup>5</sup> and IMS-related classes, you must also activate SETROPTS RACLIST processing for the *member class*.

For example, for terminals, issue the following command

```
SETROPTS CLASSACT(TERMINAL) RACLIST(TERMINAL)
```

**Note:** Any time you make a change to a GTERMINL profile, you must also refresh SETROPTS RACLIST processing for the TERMINAL class for the change to take effect.

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

For CICS<sup>5</sup> and IMS-related classes, you only need to activate the class (you cannot request RACLIST processing using the SETROPTS command).

```
SETROPTS CLASSACT(TIMS)
```

**Note:** If an application uses RACROUTE REQUEST=LIST,GLOBAL=YES to RACLIST a class, you can use SETROPTS RACLIST (*classname*) REFRESH to refresh the class. This includes the CICS and IMS classes that can't be RACLISTed with the SETROPTS RACLIST command.

## Adding a resource to a profile

To add a resource to a profile, issue the RALTER command with the ADDMEM operand, and then refresh the in-storage profiles for that class. For example:

```
RALTER GTERMINL DEPT35 ADDMEM(M01RF268)
SETROPTS RACLIST(TERMINAL) REFRESH
```

## Deleting a resource from a profile

To delete a resource from a profile, issue the RALTER command with the DELMEM operand, and then refresh the in-storage profiles for that class. For example:

```
RALTER GTERMINL DEPT35 DELMEM(M01RF268)
SETROPTS RACLIST(TERMINAL) REFRESH
```

## Which profiles protect a particular resource?

RACF does not prevent you from specifying the same resource in more than one resource grouping profile. If you do so, more than one profile is used to determine the actual protection used. (See [“Resolving conflicts among grouping profiles”](#) on page 214.) It can be difficult to determine exactly what protection any one resource has.

To find out if more than one profile protects a particular resource, issue the RLIST command with the RESGROUP operand as follows:

<sup>5</sup> The FCICSFCT class is an exception. You can use SETROPTS RACLIST with that class.

```
RLIST TERMINAL resource-name RESGROUP
```

Make sure to specify the member class (such as TERMINAL or DASDVOL) on the RLIST command. The profiles that protect the terminal appear in the RLIST output under the RESOURCE GROUPS heading.

For example, assume that the following commands is issued.

```
RDEFINE GTERMINL DEPT20 ADDMEM(T1 T2 T3)
RDEFINE GTERMINL DEPT22 ADDMEM(T3)
```

To list all of the profiles that protect terminal T3, enter:

```
RLIST TERMINAL T3 RESGROUP
```

In response, RACF displays:

```
RESOURCE GROUPS
-----
DEPT20    DEPT22
```

**Note:** If a "member class" profile exists for the resource (in this example, if RDEFINE TERMINAL T3 had been issued), the RLIST output includes both the resource groups and the listing of the TERMINAL profile.

## Resolving conflicts among grouping profiles

A resource name can appear in more than one resource group and can also have a profile of its own. If a resource is protected by more than one profile, RACF resolves any conflicts by merging the information from the individual profiles. Merging occurs during RACLIST processing according to the default rules shown in [Table 20 on page 214](#) and occurs only under one of the following conditions:

- A member name is defined as a member of more than one grouping-class profile.
- A member name is defined as both a profile in the member class and a member of a grouping-class profile.

**Guideline:** Do not specify the same member name in more than one grouping-class profile. Because of the way in which profiles are merged, it might become difficult to determine exactly what protection any one resource has.

Grouping-class profiles are processed in the order that the SEARCH or RLIST command would show them. Member-class profiles, which are processed after all grouping profiles are processed, are also processed in the order that the SEARCH or RLIST command would show them.

If you want to change the order in which profiles are processed or you do not want to use the default rules for merging the information from multiple profiles, you can use the REQUEST=LIST exit routines to change them. For details about RACLIST processing and the REQUEST=LIST exit routines, see [z/OS Security Server RACF System Programmer's Guide](#) and [z/OS Security Server RACROUTE Macro Reference](#).

Table 20. Rules for merging conflicting profiles

Merging rule	Example
The most restrictive UACC is used.	If one profile has a UACC of NONE and another has a UACC of UPDATE, the UACC of NONE is selected.
For any particular user, the least restrictive of any duplicate access list entry is used.	For user STEVEH, if one profile has an access list entry of STEVEH(NONE) and another has an access list entry of STEVEH(UPDATE), the access list entry of STEVEH(UPDATE) is selected.
Auditing is done if requested by any of the duplicate profiles. All AUDIT and GLOBALAUDIT values are processed.	If one profile requests auditing and another does not, auditing is selected. If one profile requests logging of all failures and the other profile requests logging of all successes, both successes and failures are logged.

Table 20. Rules for merging conflicting profiles (continued)

Merging rule	Example
The values of the most recent profile are used for the following profile values: APPLDATA, DATA, NOTIFY, and OWNER.	If the first profile specifies NONOTIFY and another profile specifies to notify USER1, no user is notified.
The highest level is used.	If one profile specifies LEVEL(99) and another specifies LEVEL(00), the value used for level is 99.
All unique security categories are processed.	If one profile has a category of ACCTG and another profile has a category of PAYROLL, both the ACCTG and PAYROLL categories are selected.
The lowest security level is used.	If one profile has a security level of CONFIDENTIAL and another has the lower security level of ROUTINE, the security level of ROUTINE is selected.
The security label of the working profile is used. By default, this is the security label of the first profile found.	RACF chooses the security label of the first profile it encounters during RACLIST processing and ignores the security labels for subsequent profiles that must be merged. Therefore, the value of the "merged" security label depends on the order in which the profiles are processed.
The terminal timezone of the working profile is used. By default, this is the TIMEZONE value of the first profile found.	RACF chooses the timezone value of the first profile it encounters during RACLIST processing and ignores the timezone values for subsequent profiles that must be merged.

## How is WARNING mode merged for conflicting multiple profiles?

When a RACLIST is issued and multiple profiles in the grouping class are merged based on their member names, if one of the profiles is in WARNING mode, RACF uses the setting (either WARNING or NOWARNING) for the first profile member or member profile of that name it encounters.

### Note:

1. RACF processes grouping profiles before it processes member profiles.
2. A RACROUTE REQUEST=LIST selection exit can change the results.
3. The FILTER= or LIST= operands of RACROUTE REQUEST=LIST can change the results.
4. RACF ignores WARNING completely unless the issuer of RACROUTE REQUEST=LIST specified RELEASE=1.8 or higher on the macro invocation.

## Considerations for resource group profiles

When you work with resource group profiles, keep these considerations in mind:

- There are limitations on the size of resource access lists and profiles, particularly for profiles that are processed in storage by the SETROPTS RACLIST command or the RACROUTE REQUEST=LIST macro. For more information, see [“Limiting the size of your access lists” on page 193](#).
- Do *not* issue the SETROPTS RACLIST command for the resource group class (for example, GTERMINL or GDASDVOL).

Instead, specify the related member class (for example, TERMINAL or DASDVOL). When you RACLIST the TERMINAL class, RACF RACLISTs the GTERMINL class for you.

- You *cannot* use the SETROPTS command to RACLIST resource classes for these resources:
  - CICS resources (*except* FCICSFCT)
  - All IMS resources.

These CICS and IMS resources issue RACROUTE REQUEST=LIST at initialization time.

To refresh CICS classes that are *not* RACLISTed with RACROUTE REQUEST=LIST,GLOBAL=YES or SETROPTS RACLIST, issue this CICS command from the operator console:

```
CEMT PERFORM SECURITY REBUILD
```

When IMS is refreshed, the IMS classes are refreshed as well.

- You *cannot* specify generic profile names in the resource group class.
- You *can* specify generic names on the ADDMEM operand. However, you should consider defining your generics in the MEMBER class so that the RLIST command can be used to find the generic profile that protects a resource.
- A resource group profile, which is associated with only *one* resource class, *cannot* be used to group resources from two different classes.
- If you use resource grouping profiles, consider avoiding the use of the related member class.

For example, if you use GTERMINL profiles, convert entirely to using GTERMINL profiles, and delete all TERMINAL profiles. This can ease the administration of terminal authorizations. For example, the SEARCH command lists profile names for only one class at a time: GTERMINL or TERMINAL.

**Note:** Remember that you can use RLIST to find the generic that matches a name only if you use member class profiles. RLIST does not provide this support for members of grouping class profiles. Therefore, you must decide which approach is easier to administer. It might be better to define all discrete names as members of grouping profiles and all generic names as member class profiles. That allows you to use multiple SEARCH or RLIST commands when necessary.

When converting generic TERMINAL profiles to GTERMINL profiles, you can specify generic characters on the ADDMEM operand to obtain the same coverage.

## Using RACF variables in profile names (RACFVARS class)

To minimize administrative effort, use a single profile to protect multiple resources whenever you can. One way to do this is to define a generic profile, using a generic character in the profile name. Another way is to use a resource grouping class. This topic discusses a third way, which is to use a RACF variable in the profile name.

Use a RACF variable in a profile name to define one general resource profile to protect many resources with dissimilar names when no resource grouping class is available. RACF variables can be used for general resource profiles only. You cannot use them in data set profile names. A profile that contains a RACF variable in its name is considered a generic profile.

## Defining RACF variables

RACF variable names must begin with an ampersand (&), can be up to eight characters long, and cannot contain any periods (.) or generic characters. Do not define variable names that start with &RAC; they are reserved for RACF use.

**Note:** The variable values &RACUID and &RACGPID are not *RACF variables* and are not members of the RACFVARS class. These variables have specific uses that are unrelated to this topic and are described elsewhere in this document.

Several resource names can be assigned to each variable through a profile in the RACFVARS class. The resource names assigned to the variable are added as member names to the RACFVARS profile. The resource names can be up to 39 characters long and cannot contain any generic characters. All of the resources must belong to the same class and must belong to a class that accepts generic profile names.

The UACC of a RACFVARS profile controls who can display the profile to see how a particular variable is defined. A UACC of READ allows anyone to look at the profile using the RLIST command. A UACC of NONE denies the access.



PERMIT commands that are issued for a RACFVARS profile affect the administration of the profile, not access to the resource that is protected with the RACFVARS variable name. For example, to allow users to access tape volumes protected by a TAPEVOL profile called &PAYTAPE, issue a PERMIT command for the profile in the TAPEVOL class, not the profile in the RACFVARS class. However, to allow a user to change the RACFVARS profile called &PAYTAPE, give the user ALTER access authority to the profile in the RACFVARS class, not the profile in the TAPEVOL class (Note that ALTER access can be restricted by the security administrator. See, [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259).

Because the RACFVARS class requires high performance, you must RACLIST the profiles. To activate the RACFVARS class and RACLIST the profiles, enter:

```
SETROPTS CLASSACT(RACFVARS) RACLIST(RACFVARS)
```

Any time a change is made to a RACFVARS profile, the in-storage profiles for the RACFVARS class must be refreshed for the changes to take effect. To refresh the in-storage profiles for the RACFVARS class, enter:

```
SETROPTS RACLIST(RACFVARS) REFRESH
```

The class containing the profile names that are affected by your RACFVARS profile need not be RACLISTed. However, if the class is already RACLISTed or GENLISTed, you must also refresh the in-storage generic profiles for that class to activate your change to the RACFVARS profile.

#### Examples:

```
SETROPTS RACLIST(class) REFRESH
-or-
SETROPTS GENERIC(class) REFRESH
```

## Example of protecting several tape volumes using the RACFVARS class

The easiest way to show how to use the RACFVARS class is by an example.

Suppose you want to protect several tape volumes called TAP111, A22222, and B330LD, and give ALTER access to them only to a group called PAYGRP. There is no resource grouping class for the TAPEVOL class and the names of the tape volumes are unlike, so you choose the RACFVARS method to protect the tape volumes. To do this, enter:

```
RDEFINE RACFVARS &PAYTAPE UACC(NONE) ADDMEM(TAP111 A22222 B330LD)
RDEFINE TAPEVOL &PAYTAPE UACC(NONE)
PERMIT &PAYTAPE CLASS(TAPEVOL) ID(PAYGRP) ACCESS(ALTER)
SETROPTS CLASSACT(TAPEVOL RACFVARS) RACLIST(RACFVARS)
```

If you later need to add a tape volume called C44TAP to the list of protected tape volumes, enter:

```
RALTER RACFVARS &PAYTAPE ADDMEM(C44TAP)
SETROPTS RACLIST(RACFVARS) REFRESH
```

## Using RACF variables

There are several ways you can use RACF variables. They include:

- Using a RACF variable as the entire name of a profile
- Using a RACF variable as a qualifier in a profile name that has more than one qualifier
- Using the &RACLNDE variable to identify local nodes

### Using a RACF variable as the entire name of a profile

You can use a RACF variable as the entire name of a profile. For example, a TAPEVOL profile name has only one qualifier. It identifies the tape volume to protect. You can create a RACFVARS profile named &PAYTAPE that specifies several tape volumes as members. If you then define a TAPEVOL profile called &PAYTAPE, the profile protects all of the member tape volumes.

## Using a RACF variable as a qualifier

You can use a RACF variable as a qualifier in a profile name that has more than one qualifier. For example, a JESJOBS profile name identifies the job names to protect. It takes the form `SUBMIT.nodename.jobname.userid` and consists of several qualifiers. You can create a RACFVARS profile named `&PROTJOB` that specifies several job names as members. If you then define a JESJOBS profile called `SUBMIT.*.&PROTJOB.*`, the profile protects all of the member job names from any node and any user.

In a profile name, a variable name is ended by the eighth character, the end of the profile name, or one of the following characters, whichever occurs first: a period (.), another ampersand (&), a percent sign (%), or an asterisk (\*).

For example, suppose you define the following profile:

```
RDEFINE RACFVARS &ABCDEFGH ADDMEM(A B)
```

In this case, `X.&ABCDEFGH.Y.Z` matches both `X.AY.Z` and `X.BY.Z`.

## Using the &RACLNDE profile to identify local nodes

A RACFVARS profile named `&RACLNDE` can be used to identify node names that are to be treated as local nodes. This profile is extremely useful with the NODES, JESJOBS, or JESSPOOL classes, and is required for spool reload functions. For example, see [“Allowing a TSO user to cancel all jobs originating from local nodes” on page 466](#).

## How RACF uses the RACFVARS member list

When RACF authorization checking matches a resource name with the name of a general resource profile that contains a RACF variable, it locates the RACFVARS profile that matches the RACF variable. RACF then compares each character of the resource name with each character of each member name in the RACFVARS profile until it finds the first match between a sequence of characters in the resource name and a RACFVARS member name. RACF compares the members *in the order in which they occur* in the RACFVARS profile. In other words, the first name in the member list is compared first and the last name is compared last until a match is found. When a match is found, RACF substitutes the member name for the RACF variable in the name of the general resource profile and then searches for a matching general resource profile to check the access authorization.

## Administering the RACFVARS member list

Create the member list of a RACFVARS profile by issuing the ADDMEM operand of the RDEFINE command. When you specify multiple members, they are added to the RACFVARS profile in the *same* order that you specify them with the ADDMEM operand of the RDEFINE command. For example, if you specify `ADDMEM(A B)` with the RDEFINE command, the members are stored in the RACFVARS profile as `A B`.

If you issue the RALTER command to add one or more members to an existing RACFVARS profile, the new members are stored in the profile in the *reverse* of the order in which you specified them with the ADDMEM operand of the RALTER command. Additionally, if the existing profile already contains members, the new members are stored ahead of the existing members. For example, if you specify `ADDMEM(C D)` with the RALTER command to add members to an existing profile that already contains the members `A B`, the resulting member list stored in the profile is `D C A B`.

To view the members in a RACFVARS profile, issue the `RLIST RACFVARS variable-name` command. Note that the RLIST command lists the members in alphabetical order, not in the order in which they occur in the RACFVARS profile.

To view the members in the order in which they occur in a RACFVARS profile, use the output of the database unload (IRRDBU00) utility. For an example of using the DFSORT ICETOOL reporting tool to format a RACFVARS member report, see [“Creating a RACFVARS member report” on page 381](#).

**Tip:** To reorder a RACFVARS member list, first use the RLIST command to list and make note of the members of the RACFVARS profile. Then delete the profile using the RDELETE command, and reissue the RDEFINE command with the ADDMEM operand to specify the members in the new order.

**Guidelines:** Because the order of the member names in the RACFVARS profile can be a critical factor in the successful matching of a resource name with the expected general resource profile, the following guidelines apply:

- When possible, avoid specifying a member name that is a subset of another member name in the same list. When impossible to avoid, the member name that is a subset of another name should follow the name of which it is a subset.
- Minimize the number of members in a single member list.
- Simplify the name of a general resource profile that contains a RACF variable by minimizing the use of generic characters after the initial ampersand (&) of the variable name.

## Examples of debugging complex RACF variables and member lists

The following three examples illustrate working with complex RACF variables and RACFVARS member lists.

### Example 1

```
RDEFINE RACFVARS &CTM ADDMEM(TEST TESTA)
RDEFINE SURROGAT &CTM.SUBMIT UACC(NONE)
PERMIT &CTM.SUBMIT CLASS(SURROGAT) ID(USER1) ACCESS(READ)
```

The job TESTA1 submitted by USER1 on system PLPSC, with USER=TESTA on the job card, results in a failure and the following error message.

```
$HASP165 TESTA1 ENDED AT PLPSC - SECURITY VIOLATION
```

The failure occurs because RACF checking stops when the first four characters of the specified resource name, TESTA, match the first RACFVARS member, TEST, leaving the letter A. The remaining letter A is considered a specific part of the resource name and there is no corresponding specific part in the profile name to which it can be matched.

As a precaution, when adding RACFVARS members, order the member names. The member names that are a subset of other names should follow the names of which they are a subset.

In the example, TEST is a subset of TESTA. Therefore, to obtain the expected result, reverse the members in the RACFVARS member list.

```
RDEFINE RACFVARS &CTM UACC(NONE) ADDMEM(TESTA TEST)
```

**Note:** Ordering the members solves the problem in the example. However, this might not be the desired order in all cases.

### Example 2

```
RDEFINE RACFVARS &R ADDMEM(AB A)
RDEFINE ACCTNUM &R%.X UACC(NONE)
PERMIT &R%.X CLASS(ACCTNUM) ID(USER1) ACCESS(READ)
```

In this example, TSO user USER1 attempts to log on with account number AB.X, but profile &R%.X does not match. This results in the following error message:

```
IKJ56486I THE ACCOUNT NUMBER AB.X HAS NOT BEEN DEFINED FOR USE
```

The AB matches appropriately. However, no characters remain in the resource name to match with the generic character, %.

To obtain the expected result, reverse the members in the RACFVARS member list as follows:

```
RDEFINE RACFVARS &R ADDMEM(A AB)
```

or redefine the generic profile as follows:

```
RDEFINE ACCTNUM &R*.X UACC(NONE)
```

When you use any of the following to define a profile name, unexpected results can occur:

- Multiple RACFVARS
- A combination of RACFVARS and generic characters
- A combination of RACFVARS and specific names

### Example 3

```
RDEFINE SURROGAT &A&B.SUBMIT UACC(NONE)
PERMIT &A&B.SUBMIT CLASS(SURROGAT) ID(USER1) ACCESS(READ)
RDEFINE RACFVARS &A UACC(NONE) ADDMEM(AB A)
RDEFINE RACFVARS &B UACC(NONE) ADDMEM(B C)
```

The job AB1 submitted by USER1 on system PLPSC, with USER=AB on the job card, results in a failure and the following error message:

```
$HASP165 AB1 ENDED AT PLPSC - SECURITY VIOLATION
```

The failure occurs because RACF checking for the resource name AB matches the first member of &A which is AB. Because there is no part of the resource name to match the second part of the profile name specified by &B, the compare fails.

The resource name must match with a member of each of the RACFVARS used to define a profile.

To obtain the expected results, reverse the members in the RACFVARS member list of &A:

```
RDEFINE RACFVARS &A UACC(NONE) ADDMEM(A AB)
```

However, the set of resource names that was valid has now changed. For example, the specific resource name, ABB, was valid and is no longer valid.

**Guideline:** To avoid unexpected results, reduce the complexity of profiles.

If you decide to remove a member from a RACFVARS member list, be sure to issue the SETROPTS RACLIST REFRESH or GENERIC REFRESH commands for any classes that contain profiles that use the RACFVARS value affected by your change.

## Using RACFVARS with mixed-case classes

Using RACF variables in mixed-case profile names has limited use. A mixed-case profile is a profile in a class defined in the class descriptor table (CDT) with the CASE=ASIS option, and might, therefore, have a name that contains lowercase characters. The RACFVARS class itself is not defined with this option, so its profiles and member values will be treated as upper case. Therefore, when you define a mixed-case profile using a RACF variable, you must enter the variable name in upper case.

In the following example, \$MYCLASS is a mixed-case class. Notice that the profile called My.Pet.&VAR contains mixed-case characters but the variable name (&VAR) is in upper case. This allows the variable name match the RACFVARS profile name, which must be in upper case.

```
RDEFINE RACFVARS &VAR ADDMEM(CAT DOG FISH)
RDEFINE $MYCLASS My.Pet.&VAR
```

Because the &VAR profile can contain only members with uppercase names (CAT, DOG, and FISH), a resource named My . Pet . FISH is covered by these profiles, but My . Pet . Fish is not.

**Note:** Do not enter profile name My . Pet . &var because, although RACF will accept the name, the character string &var will not match the RACFVARS profile name &VAR and will not be recognized as a RACF variable.

## Controlling VTAM LU 6.2 bind

You can control which type 6.2 logical units can establish sessions with each other. This includes the ability to use RACF to specify the values used in password-on-bind processing. For more information, see “RACF and APPC” on page 248, *z/OS Communications Server: SNA Programmer's LU 6.2 Guide*, or *z/OS MVS Planning: APPC/MVS Management*.

To do this, perform the following steps:

1. Ask your VTAM system programmer for the following information for each VTAM LU 6.2 pair:
  - The network ID and the LU identifiers for each member of the pair.
  - Whether or not a password is required for session verification based on the VERIFY option specified on the VTAM APPL statement for the LU in SYS1.VTAMLST. (This password is referred to as the *session key* in your RACF definitions.)
  - Whether or not the NQNAME option is specified for the ACB. If it is specified, this indicates that network-qualified names support is enabled.

2. For each LU 6.2 pair, create two profiles in the APPCLU class.

On one system, enter one of the following RDEFINE commands. If network-qualified names support is not enabled, enter:

```
RDEFINE APPCLU local-netid.luid1.luid2 UACC(NONE)
```

If network-qualified names support is enabled, enter:

```
RDEFINE APPCLU local-netid.luid1.remote-netid.luid2 UACC(NONE)
```

On the other system, enter one of the following RDEFINE commands. If network-qualified names support is not enabled, enter:

```
RDEFINE APPCLU local-netid.luid2.luid1 UACC(NONE)
```

If network-qualified names support is enabled, enter:

```
RDEFINE APPCLU local-netid.luid2.remote-netid.luid1 UACC(NONE)
```

where:

### **local-netid, remote-netid**

are the network IDs (NETID) of the partners. These IDs are specified on the VTAM start option NETID, which is in the ATCSTRxx member of SYS1.VTAMLST.

### **luid1, luid2**

are the LU names of the partners. In each case, the first LU name specified is the local LU name, and the second LU name is the partner LU name.

For each profile created, the first LU name specified (*luid1*) is the *primary LU* on that system.

**Rule:** Do not specify an asterisk (\*) or any other generic character for the first two qualifiers (*netid* and *luid*).

3. Define the attributes of the sessions between the partners of each LU pair. You do this by defining a SESSION segment for each APPCLU profile using the SESSION option of the RDEFINE and RALTER commands. You can specify the following information in each SESSION segment:

### CONVSEC

Specifies the level or levels of security checking performed for each conversation between the partners of the LU pair.

### INTERVAL

Specifies the maximum number of days the session key is valid before it must be changed.

### LOCK

Indicates that a session between the partners of the LU pair cannot be established.

### SESSKEY(*session-key*)

Specifies the password used to verify sessions between the partners of this LU pair. If specified, the SESSKEY value must be the same in both APPCLU profiles for this LU pair. A session key might be required based on the VERIFY option specified on the VTAM APPL statement for this LU pair.

**Note:** Session keys are 64-bit data encryption standard (DES) keys. With 64-bit DES encryption, 8 of the 64 bits are reserved for use as parity bits. This means that those 8 bits are not part of the 56-key. In hexadecimal notation, the DES parity bits are: X'0101 0101 0101 0101'. Therefore, any two 64-bit keys are equivalent if their only difference is in one or more of these parity bits. For example, the following SESSKEY values, although appearing to be quite different, are equivalent because they differ only in the last bit of each byte:

```
BDF0KM4Q (X'C2C4 C6F0 D2D4 F4D8'  
CEG1LN5R (X'C3C5 C7F1 D3D5 F5D9'
```

For detailed information about using the RDEFINE and RALTER commands to define options in the SESSION segment, see [z/OS Security Server RACF Command Language Reference](#).

4. Optionally, for maintenance purposes, give users and groups appropriate access authority:

```
PERMIT profile-name CLASS(APPCLU) ID(userid or groupname)  
ACCESS(access-authority)
```

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

#### NONE

Allows no access to the profile

#### READ

Allows users to list the profile (for example, using the RLIST and SEARCH commands)

#### UPDATE

Is the same as READ

#### CONTROL

Is the same as READ

#### ALTER

Allows users to change the profile (if the profile is discrete)<sup>6</sup>

5. When you are ready to start using the protection defined in the profiles, activate the APPCLU class on every system on which you want to use the APPCLU profiles:

```
SETROPTS CLASSACT(APPCLU)
```

**Note:** You cannot issue the SETROPTS RACLIST command for the APPCLU class. VTAM does this for you (using RACROUTE REQUEST=LIST).

To activate your APPCLU profile changes for an active application, issue the VTAM MODIFY PROFILES command to refresh the RACF profiles in storage. For syntax and usage information about the MODIFY PROFILES command, see [z/OS Communications Server: SNA Operation](#).

### Example:

Suppose there are two large nodes, one in New York and the other in Tokyo.

---

<sup>6</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

Network-qualified names support is not enabled.

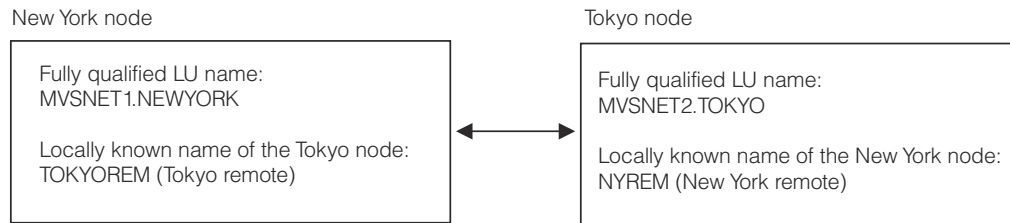


Figure 8. Example of two network LU partners

On the New York node, perform the following steps:

1. Define a profile for the Tokyo LU partner:

```
RDEFINE APPCLU MVSNET1.NEWYORK.TOKYOREM SESSION(SESSKEY(KEY1)) UACC(NONE)
```

2. Activate the APPCLU class:

```
SETROPTS CLASSACT(APPCLU)
```

On the Tokyo node, perform the following steps:

1. Define a profile for the New York LU partner:

```
RDEFINE APPCLU MVSNET2.TOKYO.NYREM SESSION(SESSKEY(KEY1)) UACC(NONE)
```

2. Activate the APPCLU class:

```
SETROPTS CLASSACT(APPCLU)
```

## Protecting applications

For applications that specify the APPL operand on the RACROUTE REQUEST=VERIFY macro, you can use a profile in the APPL class to control which users can access the application.

To do this, perform the following steps:

1. Determine the name of your application.
2. Verify with your programmer that the name of the application is specified on the APPL operand of the RACROUTE REQUEST=VERIFY macro.
3. Create a profile in the APPL class:

```
RDEFINE APPL applname UACC(NONE)
```

4. Give users and groups READ access, as appropriate.

```
PERMIT applname CLASS(APPL) ID(userid or groupname) ACCESS(READ)
```

5. If you have not already done so, activate the APPL class:

```
SETROPTS CLASSACT(APPL)
```

6. For performance reasons, request SETROPTS RACLIST or SETROPTS GENLIST processing for the APPL class.

**Note:** This might be important if many users enter the system under the control of your application (where your application issues the RACROUTE REQUEST=VERIFY macro for each user).

For information on how authorization checking takes place, see [“Authorizing access to RACF-protected applications” on page 733](#).

For details about how CICS uses APPL profiles to control access to CICS regions, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).

## Protecting DFP-managed temporary data sets

You can protect DFP-managed temporary data sets. Normally, these data sets are considered protected from any accesses except by the job or session that created them, and therefore do not need to be protected by RACF. However, the following situations could leave a temporary data set unprotected:

- A system failure
- An initiator failure or initiator termination by the FORCE command
- An automatic restart - between the failure and the restart

In these cases, if the TEMPDSN class is active, only users with the OPERATIONS attribute can scratch any residual DFP-managed temporary data sets remaining on a volume.

**Note:** The user with the OPERATIONS attribute can access the data set only to scratch the data set. No other access is allowed (such as would be allowed by READ or UPDATE access authority to the data set).

To activate the TEMPDSN class, enter:

```
SETROPTS CLASSACT(TEMPDSN)
```

When you share the RACF database with a downlevel system running z/OS V1R12 or earlier, avoid activating the TEMPDSN class when current users or jobs are using temporary data sets. It might cause users or jobs on the downlevel system to receive an ABEND, as shown in the following scenario:

1. The job or user on the downlevel system allocates a temporary data set.
2. You activate the TEMPDSN class.
3. The job or user opens the data set.
4. Because activating the TEMPDSN class restricts the authority to open a temporary data set, the user or job receives an ABEND.

## Protecting file services provided by LFS/ESA

If LAN File Services/ESA (LFS/ESA) Release 1 is installed, you can use the LFSCLASS class to control user access to LAN File Services.

The resource name contains two parts:

- A data set name
- A workstation directory name

The two parts are separated with a colon. The workstation directory name contains the directory name and at least one subdirectory, with a backslash (\) before every subdirectory name. A maximum length of 246 characters is supported.

Here is an example of using the LFSCLASS class:

- Create a profile in LFSCLASS class:

```
RDEFINE LFSCLASS MVS.DATASET.NAME:\DIR1\SUBDIR
```

- If you have not already done so, activate the LFSCLASS class:

```
SETROPTS CLASSACT(LFSCLASS)
```

**Note:** RACF supports the backslash (\) character for use with the LFSCLASS class. However, you might need to change your VTAM translation table to support the backslash (\). For more information, refer to [z/OS TSO/E Programming Guide](#) for your operating system.



## Protecting terminals

There are several methods of controlling the use of terminals that are connected to your system. The following sections describe these methods:

- “[Creating profiles in the TERMINAL and GTERMINL classes](#)” on page 225. You must give users at least READ access authority in order to allow them to use protected terminals. *You must do this before using any of the other methods for controlling terminals.*
- “[Controlling the use of undefined terminals](#)” on page 226. By specifying TERMINAL(NONE) on the SETROPTS command, you can prevent users from logging on to terminals unless the terminals are protected by profiles in the TERMINAL or GTERMINL classes.

**Important:** Do not protect undefined terminals unless you have created profiles that allow users to access the terminals they currently use.

- “[Limiting specific groups of users to specific terminals](#)” on page 227. By specifying NOTERMUACC on the ADDGROUP or ALTGROUP command, you can restrict users in those groups to terminals whose access lists specifically allow the user or the user's group to use the terminal.
- “[Limiting the times that a terminal can be used](#)” on page 227. By specifying the WHEN operand on the RDEFINE and RALTER commands for profiles in the TERMINAL and GTERMINL classes, you can specify the days and times that users can log on to terminals.
- “[Using security labels to control terminals](#)” on page 227. If the SECLABEL class is active, you can control access to terminals by specifying security labels for profiles in the TERMINAL and GTERMINL classes.
- “[Using the TSO LOGON command with the RECONNECT operand](#)” on page 227. TSO allows verification and checking so that a user can resume an interrupted session from a new terminal.

For a description of authorization checking for terminals, see “[Authorizing access to RACF-protected terminals](#)” on page 730.

## Creating profiles in the TERMINAL and GTERMINL classes

If you create a profile in the TERMINAL or GTERMINL class, you must give users at least READ access authority in order to allow them to use the protected terminal.

1. To protect a terminal using RACF, create a profile for it using the RDEFINE command. On the command, specify the universal access authority (UACC) you want to assign to the terminal. The following command defines a profile for terminal M01RF267 and specifies a UACC of NONE.

```
RDEFINE TERMINAL M01RF267 UACC(NONE)
```

On systems using VTAM, the terminal's node name is the RACF resource name. See your systems programmer for node name information.

2. Use the PERMIT command to allow users and groups to use the terminal. You must give a user at least READ access authority to the terminal. Otherwise, the user is not authorized to use the terminal. For example, the following command grants users SMITH and JONES READ access authority to terminal M01RF627.

```
PERMIT M01RF267 CLASS(TERMINAL) ID(SMITH JONES) ACCESS(READ)
```

**Important:** After you define a terminal and protect it with a UACC of NONE, no one can use the terminal until you grant users or groups READ access authority to the resource.

3. When you are ready to start using the protection defined in the profiles, activate the TERMINAL class. You should also consider activating SETROPTS RACLIST processing for the class. SETROPTS RACLIST processing helps ensure high performance when access authorities are checked. Also, if you are using GTERMINL profiles, you *must* request RACLIST processing for the TERMINAL class. You can do these two actions in one command:

```
SETROPTS CLASSACT(TERMINAL) RACLIST(TERMINAL)
```

**Note:** When you activate the TERMINAL class, RACF also activates the GTERMINL class.

**Creating a profile in the GTERMINL class:** If you want to protect several terminals in the same way, but their names do not allow you to create a generic profile, you can create a profile in the GTERMINL class for them. For example, to protect terminals M01RF267, M03RF168, and M04GG148 with one profile, you could create a profile with a name you choose, such as DEPT35:

```
RDEFINE GTERMINL DEPT35 UACC(NONE) ADDMEM(M01RF267 M03RF168 M04GG148)
```

To allow group FINANCE to use these terminals, enter:

```
PERMIT DEPT35 CLASS(GTERMINL) ID(FINANCE) ACCESS(READ)
```

**Note:** After creating or changing a GTERMINL profile, you must request SETROPTS RACLIST processing for the TERMINAL class to make the changes effective on the system.

To protect another terminal, named M01RF299, with the same profile, change the DEPT35 profile as follows:

```
RALTER GTERMINL DEPT35 ADDMEM(M01RF299)  
SETROPTS RACLIST(TERMINAL) REFRESH
```

To stop protecting terminal M03RF168 with this profile, change the DEPT35 profile as follows:

```
RALTER GTERMINL DEPT35 DELMEM(M03RF168)  
SETROPTS RACLIST(TERMINAL) REFRESH
```

## Controlling the use of undefined terminals

You can also use RACF to control the use of undefined terminals that are connected to your system. To control the use of undefined terminals, you must first activate the TERMINAL class as shown above. After the TERMINAL class is active, you can control whether users can log on to undefined terminals by issuing the SETROPTS command with the TERMINAL operand. The TERMINAL operand specifies the universal access authority, either READ or NONE, that RACF associates with undefined terminals on your system.

To allow undefined terminals to be used for logging on, enter:

```
SETROPTS TERMINAL(READ)
```

To prevent undefined terminals from being used for logging on, enter:

```
SETROPTS TERMINAL(NONE)
```

**Important:** Before you specify NONE, be sure that you define some terminals to RACF and give the appropriate users and groups proper authorization to use them. Otherwise, *no one* can log on to your system.

## Combining the SETROPTS TERMINAL command with TERMINAL profiles

If you want to control selected terminals, specify READ on the TERMINAL operand of the SETROPTS command. When you specify READ, all users can access all terminals. To control access to selected terminals, define each terminal individually and specify a UACC of NONE. Then, create an access list for each terminal that contains the user IDs of the users who require access to the terminal.

If you decide that you want to control *all* terminals, specify NONE on the TERMINAL operand of the SETROPTS command. When you specify NONE, only users and groups that you authorize to use a terminal through its access list can use it.

**Important:** Before you specify NONE, be sure that you define some terminals to RACF and give the appropriate users and groups proper authorization to use them. Otherwise, *no one* can log on to your system.

## Limiting specific groups of users to specific terminals

When defining or changing a group profile, you can specify that the group can log on only to those terminals to which the group (or individual users within the group) are specifically authorized. If the group terminal option NOTERMUACC is in effect (note that TERMUACC is the default) for a group on the ADDGROUP or ALTGROUP command, users of the group can use only those terminals to which they are specifically authorized on the access list in the TERMINAL profile protecting the terminal.

For example, if you want to allow group PAYROLL to log on only to terminals in the payroll office, protect the payroll terminals with a profile:

```
RDEFINE GTERMINL PAYTERMS ADDMEM(M02RF001 M11RF203) UACC(NONE)
```

Give the PAYROLL group READ access:

```
PERMIT PAYTERMS CLASS(GTERMINL) ID(PAYROLL) ACCESS(READ)
```

Ensure that the PAYROLL group profile has NOTERMUACC specified:

```
ALTGROUP PAYROLL NOTERMUACC
```

This prevents users in group PAYROLL from logging on to another terminal just because the profile protecting that terminal has a UACC of READ.

**Note:** If the list-of-groups option (SETROPTS GRPLIST) is in effect, RACF uses the TERMUACC/NOTERMUACC option from the user's current connect group, but RACF can grant terminal access through any of the user's connect groups.

**Tip:** When a user is connected to multiple groups and the application he uses to logon allows him to specify a group name in addition to a user ID, define NOTERMUACC on each of his group connections to ensure that the user can logon only to terminals that he or one of his connect groups is explicitly authorized to access.

## Limiting the times that a terminal can be used

RACF allows you to limit the use of specific terminals to certain days of the week and certain hours of the day. To control when the system can be accessed from the terminal, use the WHEN operand on the RDEFINE and RALTER commands for the TERMINAL or GTERMINL classes. For more information on the time and day-of-week controls, see “Limiting when a user can access the system” on [page 73](#) or the command descriptions in *z/OS Security Server RACF Command Language Reference*.

For example, to allow logons at a terminal only between 7 a.m. and 5 p.m. during the week, specify WHEN(DAYS(WEEKDAYS) TIME(0700:1700)) on the RDEFINE or RALTER command.

## Using security labels to control terminals

If the TERMINAL class is active, RACF checks a user's authority to use a terminal. If the SECLABEL class is also active, and the TERMINAL profile contains a security label, the user must log on with a security label that is equivalent to the security label of the terminal. If the user does not specify a security label when logging on, the user runs with the security label of the terminal if the user has at least READ authority to that security label, unless the terminal's security label is SYSMULTI.

You can use this to limit the sensitivity of the data that users can access from the terminal. For example, if you have some terminals that can be accessed easily by many users, you can assign those terminals a low-sensitivity security label, such as SYSLOW. This prevents users from using those terminals to access data that has a security label higher than the terminal's security label.

## Using the TSO LOGON command with the RECONNECT operand

TSO provides a line drop facility that enables a user to log on to TSO from another terminal and reconnect to the existing session by issuing the LOGON command with the RECONNECT operand.

During this logon to the new terminal, LOGON command processing requests that RACF verify the user's user ID and password, password phrase, or operator identification card. Also, if the TERMINAL class is active, the user's authority to access the new terminal is checked. Note that the user cannot change connect groups when the RECONNECT operand is used.

If verified and authorized, the user can resume the interrupted session from the new terminal.

## Protecting consoles

You can require operators to log on to and log off from MCS-managed consoles by specifying options in the CONSOLxx member of the SYS1.PARMLIB data set. For more information, see:

- [z/OS MVS Initialization and Tuning Reference](#)
- [z/OS MVS System Commands](#)
- [z/OS MVS Planning: Operations](#)

When the CONSOLE class is active and a console being used is protected by a profile in the CONSOLE class, RACF ensures that the person attempting to logon has the proper authority to do so. Using RACF, you can control the use of JES and MCS system consoles on your system. This topic describes how to protect MCS consoles. See also [“Remote workstations \(RJP/RJE consoles\)”](#) on page 491.

### Note:

1. The SETROPTS TERMINAL command does not apply to consoles.
2. The TERMUACC operand on the ADDGROUP and ALTGROUP commands does not apply to consoles.
3. You cannot specify the WHEN operand on the RDEFINE and RALTER commands for profiles in the CONSOLE class.

For a description of authorization checking for consoles, see [“Authorizing access to consoles, JES input devices, APPC partner LUs, or IP addresses”](#) on page 731.

To control the use of MCS consoles, perform the following steps:

1. Ask your system programmer for the following information:

- The name or ID of the console to be protected

**Sysplex consideration:** If you share the RACF database with downlevel systems, the console might have a 2-byte console ID rather than a console name. To protect a console ID, define the resource using the console ID in place of the console name.

- The universal access authority (UACC) to specify for the console
- The user ID or group name of the operator or operators to whom you want to grant access
- The security label to be assigned to that console (if security labels are being used)

2. Create a profile for each console using the RDEFINE command.

```
RDEFINE CONSOLE console-name UACC(NONE)
```

For example, the following command defines a profile for console CON1 and specifies a UACC of NONE.

```
RDEFINE CONSOLE CON1 UACC(NONE)
```

3. Use the PERMIT command to allow users and groups to use the console. You must give a user at least READ access authority to the console. Otherwise, the user is not authorized to use the console.

For example, the following command grants READ access authority to group OPRGRP1 and user JONES for CON1.

```
PERMIT CON1 CLASS(CONSOLE) ID(OPRGRP1 JONES) ACCESS(READ)
```

**Important:** After you define a console and protect it with a UACC of NONE, no one can log on to the console until you grant users access authority to the console profile.

For consoles, the valid access authorities are:

**NONE**

Allows no access

**READ**

Authorizes RACF-defined users to LOGON to the specified console

4. When you are ready to start using the protection defined in the profiles, activate the CONSOLE class and activate SETROPTS RACLIST processing for the class. SETROPTS RACLIST processing helps ensure high performance when access authorities are checked. You can do these two actions in the following command.

```
SETROPTS CLASSACT(CONSOLE) RACLIST(CONSOLE)
```

If the CONSOLE class is already active and RACLISTed, issue the following command to activate your CONSOLE profile changes.

```
SETROPTS RACLIST(CONSOLE) REFRESH
```

## Using security labels to control consoles

If the CONSOLE class is active, RACF checks an operator's authority to use a console. If the SECLABEL class is also active and the console has a security label, the operator must log on with a security label that is less than or equivalent to the security label of the console to use the console.

## Protecting the vector facility

If your processor has a vector facility, you can use RACF to protect it.

To do this, perform the following steps.

1. Define the IEAVECTOR profile in the FACILITY class.

```
RDEFINE FACILITY IEAVECTOR UACC(NONE)
```

This command specifies that *no* users can use the vector facility.

2. Give READ access authority to appropriate users or groups.

```
PERMIT IEAVECTOR CLASS(FACILITY) ID(user or group) ACCESS(READ)
```

3. Activate the FACILITY class (if it is not already active).

```
SETROPTS CLASSACT(FACILITY)
```

If your installation does not need to control the use of the vector facility, you can define an entry for IEAVECTOR in the global access checking table. Global access checking allows your installation to bypass normal RACF authorization checking and, thereby, minimize processing.

To define an entry for the vector facility in the global access checking table, issue the following commands.

```
RDEFINE GLOBAL FACILITY
RALTER GLOBAL FACILITY ADDMEM(IEAVECTOR/READ)
SETROPTS GLOBAL(FACILITY)
```

For more information, see [“Setting up the global access checking table” on page 194](#).

## Controlling access to program dumps

Because program dumps can contain sensitive information including controlled programs in machine form, you should consider limiting access to them. To control access to program dumps or suppress the dumps entirely, you can use RACF, operating system facilities (such as SYS1.PARMLIB), JES2, or JES3.

### Using RACF to control access to program dumps

RACF allows you to control access to program dumps selectively. To achieve this control, you can protect program dumps using either a data set profile or a resource profile in the FACILITY class for one of the following resources: IEAABD.DMPAUTH or IEAABD.DMPAKEY.

#### Protecting program dumps using a data set profile

To protect program dumps using a data set profile:

1. Create a generic profile to protect all dump data sets with a high-level qualifier of SYS1, specifying a UACC of NONE. Enter:

```
ADDSD 'SYS1.DUMP%' UACC(NONE)
```

2. Permit selected users to access the data sets protected by the SYS1.DUMP% profile by adding them to the access list with READ access authority. Enter:

```
PERMIT 'SYS1.DUMP%' ID(userid) ACCESS(READ)
```

#### Protecting program dumps using the FACILITY class

Your installation can control when user dumps (SYSUDUMP, SYSABEND, and SYSMDUMP) of address spaces are allowed by defining a profile to protect a resource called IEAABD.DMPAUTH in the FACILITY general resource class. When an IEAABD.DMPAUTH resource profile is defined in the FACILITY class, authorization checks will be made when:

- Program control is not active (SETOPTS NOWHEN(PROGRAM)), and the failing program is not a UNIX file system program.
- Program control is active (SETOPTS WHEN(PROGRAM)) and the failing program is program controlled.
- The failing program is a UNIX program that has the program control extended attribute set, or is a set-user-ID or set-group-ID program that gained privilege as a result of the uid or gid change.

See the [“Example of defining the IEAABD.DMPAUTH profile” on page 230](#) for a description of what different access levels to the IEAABD.DMPAUTH resource allow.

The following are situations that will not check the invoking users access to the IEAABD.DMPAUTH resource:

- If program control is active and the failing program is program controlled but has an OPEN data set that is protected by PADS, a user dump is not allowed.
- If program control is active and the failing program is not program controlled, a user dump is allowed.
- If the failing program is a UNIX program that is not program controlled and is not a set-user-ID or set-group-ID program that gained privilege as a result of the uid or gid change, a user dump is allowed.

To control the dumping (with SYSABEND, SYSMDUMP, and SYSUDUMP statements) of address spaces that have tasks running in a task control block (TCB) key of less than 8, a profile protecting a resource called IEAABD.DMPAKEY must be defined in the FACILITY general resource class.

**Guideline:** Define the IEAABD.DMPAUTH profile with UACC(NONE). Then, give ACCESS(READ) to specific users using the PERMIT command.

#### *Example of defining the IEAABD.DMPAUTH profile*

1. Define a profile that protects the resource IEAABD.DMPAUTH in the FACILITY class:

```
RDEFINE FACILITY IEAABD.DMPAUTH UACC(NONE)
```

2. If you want to give a user an access authority that is different from the one you specified on the RDEFINE command (in this example, an access authority of READ), enter:

```
PERMIT IEAABD.DMPAUTH CLASS(FACILITY) ID(ASMITH) ACCESS(READ)
```

When you specify an access authority on either the RDEFINE command or PERMIT command, RACF allows access to program dumps as follows:

- A user who has UPDATE or greater authority to the IEAABD.DMPAUTH resource can obtain program dumps.
- A user who has READ authority to the IEAABD.DMPAUTH resource can obtain program dumps unless a program was fetched from a library to which the user has only EXECUTE authority. A user cannot obtain a dump of a program to which the user has only EXECUTE authority.

For more information, see [“Using EXECUTE access for programs and libraries in ENHANCED mode” on page 316](#).

- A user who has less than READ authority to the IEAABD.DMPAUTH resource cannot obtain program dumps.

### ***Example of defining the IEAABD.DMPAKEY profile***

1. Define a profile protecting a resource called IEAABD.DMPAKEY in the FACILITY class:

```
RDEFINE FACILITY IEAABD.DMPAKEY UACC(NONE)
```

2. If you want to give a user an access authority that is different from the one you specified on the RDEFINE command (in this example, an access authority of READ), enter:

```
PERMIT IEAABD.DMPAKEY CLASS(FACILITY) ID(ASMITH) ACCESS(READ)
```

When you specify an access authority on either the RDEFINE command or PERMIT command, RACF allows access to program dumps as follows:

- A user who has READ or greater authority to the IEAABD.DMPAKEY resource can obtain program dumps, even when the program is running in a TCB key that is less than 8.
- A user who has less than READ authority to the IEAABD.DMPAKEY resource can never obtain program dumps when the program is running in a TCB key that is less than 8.
- A user who has READ or greater authority to the IEAABD.DMPAKEY facility can also obtain formatted GTF data in a SNAP or ABEND dump.
- A user who has less than READ authority to the IEAABD.DMPAKEY facility can not obtain formatted GTF data in a SNAP or ABEND dump.

### ***Activating the FACILITY class***

1. If the FACILITY class is not active, activate it as follows:

```
SETROPTS CLASSACT(FACILITY)
```

You only need to issue this command once. When a general resource class is active, it remains active until your installation deactivates it.

2. To avoid possible deadlocks, issue a SETROPTS RACLIST command for the FACILITY class.

#### **Example of a Deadlock:**

There are several types of deadlocks. This example describes one way a deadlock can occur.

- Task A of a job is abending.
  - z/OS needs to check the user's authority to produce a dump of the task and issues RACROUTE REQUEST=AUTH.

- RACF needs to do I/O to the RACF database to respond to the RACROUTE request.
- Task B of the same job is already performing a RACF function and has ENQed the RACF database.
- Task A must wait until task B releases the ENQ.
- Dump processing for task A has set all other tasks in the job non-dispatchable.

Under normal circumstances, task A could wait for task B to release the ENQ. However, because dump processing for the abending task prevents task B from completing, task A cannot proceed. Task B cannot complete until task A proceeds. This causes a deadlock.

## Using non-RACF methods to control access to program dumps

You can control access to program dumps using a variety of non-RACF methods, whose documentation is beyond the scope of this document. For more information about suppressing and redirecting dump output, see *z/OS MVS Diagnosis: Tools and Service Aids*.

## Controlling the allocation of devices

You can use the DEVICES class to control which users can allocate unit record devices, teleprocessing or communications devices, and graphics devices. For example, you can use the DEVICES class to ensure that only authorized users can allocate devices by name. You cannot use the DEVICES class to protect other kinds of devices, such as tape or DASD devices.

**Note:** To control who can log on to terminals, see “Protecting terminals” on page 225. To control who can log on to consoles, see “Protecting consoles” on page 228.

To control the allocation of devices, do the following:

1. Plan which devices to protect. You can, for example, protect specific devices with discrete profiles. You can also protect several devices with generic profiles.
2. Ask your MVS system programmer to supply the following information for each device to be protected:
  - The information that is necessary to specify the name of the profile that is to protect the device, such as the device class, unit name, and device number. These terms are described in more detail in Step “3” on page 232.
  - The RACF-defined users that can allocate the device that is protected by the profile.

**Note:** With this information, you can plan whether to use generic profiles, discrete profiles, or a combination, to protect the devices on your system.

If you decide to use generic profiles, you must activate generic processing for this class before you define the profiles.

```
SETOPTS GENERIC(DEVICES)
```

3. Create profiles in the DEVICES class:

```
RDEFINE DEVICES profile-name UACC(NONE)
```

where *profile-name* has the following format:

```
sysid.device-class.unit-name.device-number
```

where:

**sysid**

is the system identifier, which is defined on the SYSNAME value in the IEASYSxx member of SYS1.PARMLIB.

**Note:** The system identifier is necessary only if different devices with the same device class, unit name, and device address can be attached to multiple systems and have different security requirements. In most cases, you should specify an asterisk (\*) for this qualifier.



**device-class**

is one of the following UCB device classes:

**TP**

Teleprocessing or communication devices

**UR**

Unit record devices

**GRAPHIC**

Graphic devices

**Note:** These device classes are consistent with the class names used on the DISPLAY U operator command.

**unit-name**

is an esoteric device group (as defined by the installation) or a generic name (such as 3800) that identifies the device or devices.

**Note:** Because a user can allocate a device using either an esoteric or generic name, you must define profiles that would protect the device in either case.

For telecommunication devices, use the following list to determine what unit name should be used. The unit name that MVS uses for these devices is based on the transmission control unit (TCU) value of the IODEVICE statement.

TCU value	Unit name
TCU=2701	2701
TCU=2702	2702
TCU=2703	2703

For all other devices, see [z/OS HCD Planning](#) for unit name information.

**Note:** For any device for which MVS does not have a unit name, MVS uses eight character zeros (for example, 00000000). Use this as the unit name in the profile name to provide security for these devices.

**device-number**

is a 4-byte field that supplies the number of a specific device.

For information about I/O device numbers, see [z/OS HCD Planning](#).

**Note:** If *unit-name* identifies an esoteric device group, specify an asterisk (\*) in this qualifier.

Here are some sample profile names for the DEVICES class:

```
SYS01.GRAPHIC.3277-2.B40
SYS01.TP.3705.3FA
SYS01.UR.3800.00E
SYS01.UR.PRINTER1.*
```

Specifying UACC(NONE) means that only users who are specifically permitted to the profile can allocate the device.

#### 4. Give users the appropriate access authority:

```
PERMIT profile-name CLASS(DEVICES)
  ID(userid or groupname) ACCESS(access-authority)
```

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

**NONE**

Does not allow the allocation of the device

**READ**

Allows the allocation of the device.

5. When you are ready to start using the security provided by these profiles, activate both the DEVICES class and SETROPTS RACLIST processing for the class. SETROPTS RACLIST processing helps ensure high performance when access authorities are checked. You can do these two actions in the following command.

```
SETROPTS CLASSACT(DEVICES) RACLIST(DEVICES)
```

**Note:** Any time you make a change to a DEVICES profile, you must also refresh SETROPTS RACLIST processing for the DEVICES class for the change to take effect. For example:

```
SETROPTS RACLIST(DEVICES) REFRESH
```

### Example 1:

The following commands allow only USER1 to allocate PRINTER1.

```
RDEFINE DEVICES SYS01.UR.PRINTER1.* UACC(NONE)
RDEFINE DEVICES SYS01.UR.3800.00E UACC(NONE)
PERMIT SYS01.UR.PRINTER1.* CLASS(DEVICES) ID(USER1) ACCESS(READ)
PERMIT SYS01.UR.3800.00E CLASS(DEVICES) ID(USER1) ACCESS(READ)
```

### Example 2:

The following commands allow only group DESIGN1 to allocate graphics devices.

```
RDEFINE DEVICES SYS01.GRAPHIC.** UACC(NONE)
PERMIT SYS01.GRAPHIC.** CLASS(DEVICES) ID(DESIGN1) ACCESS(READ)
```

## Protecting LLA-managed data sets

When you use the START LLA or MODIFY LLA commands, library lookaside facility (LLA) invokes RACF to perform an authorization check. This check is done for each parameter library data set that LLA accesses, and for each LLA-managed data set.

You can control which users can issue the START LLA and MODIFY LLA commands. To do so, perform the following steps:

1. If data set profiles for each LLA parameter library data set do not currently exist, create them. These parameter library data sets are those that contain CSVLLAxx members that specify which libraries LLA is to manage and how it is to manage them. Ensure that each LLA command user (or group to which the user belongs) has READ access to the parameter library data sets that you protect.
2. Create profiles in the FACILITY class to protect the LLA-managed data sets. These data sets are the libraries that are specified in the CSVLLAxx and LNKSTxx members of a parameter library. For example:

```
RDEFINE FACILITY CSVLLA.data-set-name UACC(NONE)
```

Where *data-set-name* is the name of the LLA-managed data set.

Because of the CSVLLA prefix used on the resource names, and because the FACILITY class profiles cannot exceed 39 characters, the *data-set-name* portion of this profile is limited to 32 characters. For data set names that are longer than 32 characters, use generics so that the FACILITY class profile stays within the 39-character limit.

### Notes:

- a. Consider creating the same FACILITY profiles as you did the data set profiles in Step “1” on page [234](#).
- b. To have this protection, you must create profiles in the FACILITY class and the DATASET class, if you do not have access to the data set already.

- c. Except for the MODIFY LLA,UPDATE,LIBRARY=command, the LLA facility first checks the user's access through the FACILITY class profile and, unless this access is allowed, then checks for access through a data set profile.

For the MODIFY LLA,UPDATE,LIBRARY= command, RACF checking uses only the FACILITY class resource CSVLLA.dsn (up to 39 characters). No authority check is performed on the DATASET class. LLA rejects the command if no matching profile exists, or if the profile does not grant at least UPDATE access.

3. Give users and groups the appropriate access authority:

```
PERMIT CSVLLA.data-set-name CLASS(FACILITY)
      ID(userid or groupname) ACCESS(access-authority)
```

This PERMIT command allows users or groups to issue LLA commands for the specified LLA-managed library. This access authority (*access-authority*) can be one of the following:

**NONE**

Allows no access.

**UPDATE**

Allows users to work with the data sets by using the LLA START and LLA MODIFY commands.

**ALTER**

For discrete profiles, allows the same access as UPDATE, plus the ability to change the profile itself.<sup>7</sup> For generic profiles, equivalent to UPDATE.

4. If you have not already done so, activate the FACILITY class:

```
SETROPTS CLASSACT(FACILITY)
```

**Example:**

For example, to control all LLA-managed data sets with the high-level qualifier CICS, enter:

```
ADDSD 'CICS.*' UACC(NONE)
PERMIT 'CICS.*' ID(CICS) ACCESS(READ)
RDEFINE FACILITY CSVLLA.CICS.* UACC(NONE)
PERMIT CSVLLA.CICS.* CLASS(FACILITY) ID(CICS) ACCESS(UPDATE)
SETROPTS CLASSACT(FACILITY)
```

This command sequence allows CICS to issue the LLA MODIFY command for the LLA-managed data sets that have the high-level qualifier CICS.

## Controlling data lookaside facility (DLF) objects (Hiperbatch)

You can use profiles in the DLFCLASS class to control whether data in QSAM or VSAM data sets can be stored in a data lookaside facility (DLF) object and managed by Hiperbatch, an extension to QSAM and VSAM. Hiperbatch can improve the performance of batch jobs by minimizing I/O to the DASD device on which the data sets are stored. For more information about DLF objects, see *MVS Programming Hiperbatch Guide*, GC28-1470, available with the z/OS V1R13 publications in the [z/OS Internet library](http://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary) ([www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary](http://www.ibm.com/servers/resourcelink/svc00100.nsf/pages/zosInternetLibrary)).

Profiles in the DLFCLASS general resource class provide the following control information to DLF:

- The existence of a DLFCLASS profile for a QSAM or VSAM data set identifies the data set as one that is eligible to be processed as a DLF object.
- The RETAIN field in the DLFDATA segment of the profile allows you to indicate to DLF whether the object is to be a retained DLF object. The RETAIN field is an optional field.
- The access list for the profile identifies the users and groups who are allowed to access the DLF object.

<sup>7</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

- The JOBNAMES field in the DLFDATA segment of the profile allows you to further restrict access to the DLF object to specific job names:
  - When you specify a list of job names, access to the DLF object is allowed only when the user ID appears in the access list and the specific job name appears in the job names list.
  - When you do not specify a list of job names, all jobs that are submitted by authorized users (that is, users who are identified in the access list of the profile) can access the DLF object.

The JOBNAMES field is an optional field.

If you do not include this information in DLFCLASS profiles, your DLF installation exit must identify the eligible data sets and retained DLF objects, and also verify user or job name access to a DLF object.

For example, for a QSAM or VSAM data set named PAYROLL.SALARY.DATA, a DLFCLASS profile named PAYROLL.SALARY.DATA allows data from the data set to be stored in a DLF object that is used by jobs accessing the data set.

To use RACF to support DLF objects, perform the following steps:

1. Create a discrete profile in the DLFCLASS class:

```
RDEFINE DLFCLASS profile-name UACC(...) DLFDATA(...)
```

where *profile-name* is the fully qualified name of the data set. (Do not specify quotes.) The profile must be a discrete profile. For example, for a data set named PAYROLL.SALARY.DATA, the profile name would be:

```
PAYROLL.SALARY.DATA
```

UACC and DLFDATA are optional. Possible options to support a DLF object are:

### UACC

Controls access to the DLF object. Access authorities are as follows:

#### NONE

Allows no access to the DLF object.

#### READ

Allows the job to read from the DLF object.

#### UPDATE

Is equivalent to READ

#### CONTROL

Is equivalent to READ

#### ALTER

Allows READ access, and also allows users to change the profile (if it is a discrete profile).<sup>8</sup>

### DLFDATA

- If the DLF object is to be retained, specify RETAIN(YES) in the DLFDATA operand.

**Note:** If you do not specify DLFDATA, or if you do not specify the RETAIN value in the DLFDATA operand, RETAIN(NO) is defaulted.

- If access to the DLF object is to be limited to specific jobs, include the JOBNAMES value in the DLFDATA operand and list all of the applicable job names, as:

```
DLFDATA(JOBNAMES(jobname1...))
```

See [z/OS Security Server RACF Command Language Reference](#) for more details and other options.

2. Give users and groups the appropriate access authority. For example:

---

<sup>8</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

```
PERMIT profile-name CLASS(DLFCLASS) ID(userid or groupname)
ACCESS(access-authority)
```

3. When you are ready to start using the protection defined in the profiles, activate the DLFCLASS class as follows:

```
SETROPTS CLASSACT(DLFCLASS)
```

4. For enhanced performance related to the DLFCLASS profiles themselves, activate SETROPTS RACLIST processing as follows:

```
SETROPTS RACLIST(DLFCLASS)
```

### Example 1: Limiting access by job name

Users AHLEE and SMITH can both access a DLF object that corresponds to a data set named SALES.DATA, but they can do this only by submitting jobs whose job names begin with TAX, or jobs named TOTALS.

Create a profile for the DLF object:

```
RDEFINE DLFCLASS SALES.DATA DLFDATA(JOBNAMES(TOTALS TAX*)) UACC(NONE)
```

Give users and groups the appropriate access authority.

```
PERMIT SALES.DATA CLASS(DLFCLASS) ID(AHLEE SMITH) ACCESS(READ)
```

If the DLFCLASS class is not active:

```
SETROPTS CLASSACT(DLFCLASS)
```

### Example 2: Not retaining a DLF object

An interactive application is invoked many times during the day by many users. This application makes many I/O operations to data set PAY.DATA. At the end of the day, the application ends and the need for the DLF object ends. To improve performance when the application is in use, create a DLFCLASS profile for the data set with RETAIN(NO) specified.

Create a profile for the DLF object:

```
RDEFINE DLFCLASS PAY.DATA DLFDATA(RETAIN(NO)) UACC(NONE)
```

Give appropriate access:

```
PERMIT PAY.DATA CLASS(DLFCLASS) ID(PAYGRP) ACCESS(READ)
```

If the DLFCLASS class is not active:

```
SETROPTS CLASSACT(DLFCLASS)
```

## Using RACROUTE REQUEST=LIST,GLOBAL=YES support

The RACROUTE REQUEST=LIST,GLOBAL=YES option creates in-storage profiles in a data space rather than in private storage. It allows multiple address spaces to share the same set of RACLISTed profiles. Each additional application that issues RACROUTE REQUEST=LIST,GLOBAL=YES for the same class can access the data space already built. Profiles that are globally RACLISTed for a class with RACROUTE REQUEST=LIST,GLOBAL=YES can be refreshed simultaneously for all users by SETROPTS RACLIST(*classname*) REFRESH. The refresh occurs without requiring the applications to suspend work.

Classes that are RACLISTed solely by this means are listed in a line in the output of the SETR LIST command, with the following format:

```
GLOBAL=YES RACLIST ONLY =
```

After all applications have given up their access to the RACLIST data space by issuing `RACROUTE REQUEST=LIST,ENVIR=DELETE`, you can delete the data space by issuing `SETROPTS NORACLIST(classname)`. Remember that the `SETROPTS RACLIST REFRESH` and `SETROPTS NORACLIST` commands process both the class specified by the command and all valid classes sharing the same `POSIT` value as the specified class. Additionally, if the system is enabled for sysplex communication, the command is propagated to the other members of the sysplex data sharing group.

For a detailed comparison of the `RACROUTE` and `SETROPTS` processes, see [z/OS Security Server RACF Diagnosis Guide](#).

## The RACGLIST class

RACF uses the `RACGLIST` class to save the results of in-storage profiles RACLISTed to a data space from any of the following commands.

- `SETROPTS RACLIST`
- `SETROPTS RACLIST REFRESH`
- `RACROUTE REQUEST=LIST,GLOBAL=YES`

RACF uses the results from the data space to create or replace profiles named `classname_nnnnn` (where `nnnnn` begins at 00001) in the `RACGLIST` class. To enable RACF to use the `RACGLIST` profiles, the installation must activate the `RACGLIST` class and prime `RACGLIST` for a specific class by issuing the `RDEFINE RACGLIST classname` command, where `classname` is a valid class name. For example, if the `RACGLIST` class name profile is `TCICSTRN`, RACF creates additional profiles named `TCICSTRN_00001`, `TCICSTRN_00002`, and so forth.

RACF uses these `RACGLIST` profiles to build the `RACLIST` data space in any of the following cases:

- For a subsequent `RACROUTE REQUEST=LIST,GLOBAL=YES` request on a different system sharing the RACF database
- For `SETROPTS RACLIST` processing during a system IPL
- After a system IPL by `RACROUTE REQUEST=LIST,GLOBAL=YES` processing
- During the processing of a propagated `SETROPTS RACLIST` or `SETROPTS RACLIST classname REFRESH` command

The `RACGLIST` class provides a single-system image for security when you use `RACROUTE REQUEST=LIST,GLOBAL=YES` or `SETROPTS RACLIST` on multiple systems that are enabled for sysplex communication. All systems and regions using a class whose `RACLIST` results have been saved as `classname_nnnnn` profiles use the same data to make security decisions. Any system that performs a `RACROUTE REQUEST=LIST,GLOBAL=YES` retrieves the same profiles. When several changes are required for profiles in that class, other systems continue to access the stored profiles until the administrator completes the changes and tells RACF to refresh the profiles by issuing `SETROPTS RACLIST(classname) REFRESH`.

### Restrictions:

1. You should use the `RACGLIST` class only if all systems sharing the RACF database belong to the same global resource serialization (GRS) complex. See the appropriate level document of [z/OS MVS Planning: Global Resource Serialization](#) for information about defining a GRS complex.
2. Using `RACGLIST` class requires more space for the RACF database. However it ensures that results of any of the following requests are consistent across all systems sharing the database:
  - `RACROUTE REQUEST=LIST,GLOBAL=YES`
  - Propagated `SETROPTS RACLIST` command
  - Propagated `SETROPTS RACLIST REFRESH` command
3. Depending on how your installation's database is set up, it might take less processing time and I/O to read the stored `RACLIST` results from the `RACGLIST` profiles than to retrieve the original discrete and generic profiles for the class name. `RACGLIST` processing might improve startup time and system availability during restarts.

4. If you are using RACGLIST support and your database is being shared by two or more MVS systems, be sure that SYSZRAC2 is *not* in the SYSTEMS exclusion list in SYS1.PARMLIB.
5. The RACGLIST class cannot be used to create copies of profiles in the CDT class.

See [z/OS Security Server RACF Diagnosis Guide](#) for a detailed description of RACLIST processing when the RACGLIST class is active.

## Administering the use of operator commands

You can control who can issue MVS and JES operator commands regardless of their point of entry. This includes, for example, commands issued at MCS consoles, inline within batch JCL, through SVC 34, or through extended console support.

You can use RACF to authorize the following:

- For MCS consoles, you can authorize individual commands, as well as command groups, to individual operators, groups of operators, or to the consoles.
- For commands issued from NJE nodes and RJE workstations, you can authorize the node or workstation to individual commands or groups of commands.

In addition, the installation can use generic profiles to define groups of commands. If RACF is not used, the system defines the groups of commands. For more information on using MVS and JES to perform command authority checking, see one of the following documents:

- For MVS system commands, see [z/OS MVS Planning: Operations](#).
- For JES2 commands, see [z/OS JES2 Initialization and Tuning Guide](#).

“Authorizing the use of operator commands” on page 239 describes how you can use RACF to provide command authority checking.

**Note:** If SDSF is installed on your system, OPERCMDS profiles control which action characters and overtypable fields users can enter on SDSF panels. For complete information on creating OPERCMDS profiles for use with SDSF, see [z/OS SDSF Operation and Customization](#).

## Authorizing the use of operator commands

You can control which groups of users (system programmers and operators) can issue commands. You can use RACF to authorize or restrict users from entering some or all commands, or specific variations of commands, or the consoles from which commands can be entered.

To control the use of operator commands, create profiles in the appropriate RACF classes that enable RACF command authorization. The specific RACF classes that you must activate depend on the input source that you want to protect. Table 21 on page 239 lists the RACF classes that must be activated for each input source.

Table 21. RACF classes used to authorize operator commands

Source of commands	RACF security class to activate	See...
Operator commands (except when from remote workstations or NJE nodes)	OPERCMDS	<a href="#">“Controlling the use of operator commands” on page 240</a>
Commands from remote workstations (require additional setup)	OPERCMDS, FACILITY	<a href="#">“Commands from RJE work stations” on page 495</a>
Commands from NJE nodes (require additional setup)	OPERCMDS, FACILITY	<a href="#">“Commands from NJE nodes” on page 495</a>
Commands entered through, or generated by, SDSF	OPERCMDS, CONSOLE	<a href="#">“Administering the use of operator commands” on page 239</a>



## Command authorization in an MCS sysplex

If a critical system problem occurs in a multiple console support (MCS) sysplex, operators must issue commands to correct the problem. This problem might inhibit access to the RACF database, so MCS saves the security environment in the security object (ENVR) and uses it to perform authorization processing against the OPERCMDS profiles. This way, no access to the RACF database is required at the time.

In order to accomplish this, RACF must be able to successfully process the security object (ENVR) indicated by the ENVRIN value of the RACROUTE REQUEST=VERIFY macro used for OPERCMDS authorization processing. In a sysplex having mixed managers, this means that commands routed to systems that use RACF must be issued from systems that use RACF. Therefore, the installation must define MCS consoles so that at least one console attached to a system using RACF is available to issue commands to another system using RACF.

This also means that no refreshing of the list of groups is done. The user ID associated with the MCS console must be reinitialized whenever its user and group data or connections are changed. See [z/OS MVS Planning: Operations](#) for more details on MCS command authority checking and how to refresh the security environment.

The user and group names are not verified against the database when the security environment is used from another system. All systems in a sysplex should use the same RACF database. This will provide consistent user, group, and OPERCMD profiles and will ensure accurate authorization checking. In addition, definitions for security categories (members of the CATEGORY profile in the SECDATA general resource class) are likely to cause problems if all systems do not use the same RACF database.

## Controlling the use of operator commands

To control the use of operator commands, do the following:

1. Ask your system programmer for the following information:
  - a. The subsystem and resource name associated with the command to be authorized. The resource names are described in the following documents:
    - For MVS system commands, see [z/OS MVS System Commands](#).
    - For JES2 commands, see [z/OS JES2 Initialization and Tuning Guide](#).
    - For RACF operator commands, you need to define profiles in the OPERCMDS class to control authorization.

**Important:** If you do not define profiles for them, these commands cannot be protected. This means that anyone at a master console or a console with system authority would be able to use these commands. However, except for the DISPLAY command which does no additional authority checking, these commands check the console's attributes when no profile is found and can still fail the request. For the DISPLAY command, you should specify READ authority.

For examples of resource names used when defining profiles in the OPERCMDS class, see [Table 22 on page 242](#) and [Table 23 on page 244](#). Other examples are as follows:

```
RDEFINE OPERCMDS RACF.SIGNOFF.** UACC(NONE)
PERMIT RACF.SIGNOFF.** CLASS(OPERCMDS) ID(DJONES) ACCESS(UPDATE)

RDEFINE OPERCMDS RACF.DISPLAY.SIGNON.** UACC(NONE)
PERMIT RACF.DISPLAY.SIGNON.** CLASS(OPERCMDS) ID(DJONES) ACCESS(READ)
```

The base command or resource names are SIGNOFF and DISPLAY.SIGNON. Although SIGNON is not required at the console (because it is the default), it must be specified in the resource name to protect the DISPLAY command.

The RVARY command (except for the ACTIVE keyword) can be authorized by profiles in the OPERCMDS class, but only if the profile(s) granting authorization have previously been RACLISTed. If the profile data disallows access, or the profile itself cannot be accessed, the command is protected by an operator prompt. The RVARY command is protected the same way, regardless of whether it is entered from TSO or as an operator command.



- b. The UACC to be associated with the command.
- c. The user IDs of the operators or the group names of the groups of operators to whom you want to grant authority.
- d. For each operator or group of operators:
  - The access authority to be assigned to the operator or group of operators.
    - For RACF operator commands (shown in [Table 22 on page 242](#)), the required access authority is READ, except for the SIGNOFF command which requires UPDATE authority.
    - For RACF TSO commands entered as operator commands (shown in [Table 23 on page 244](#)). See Note “3” on page 245.
  - Any restrictions on which consoles the operators must be using when issuing certain commands. To do this, create profiles for consoles in the CONSOLE class and then specify the WHEN(CONSOLE) operand on the PERMIT command. See [“Conditional access lists for general resource profiles” on page 193](#).

**Note:** To authorize a console to a command or group of commands, create a RACF user profile for the console and place the console's user ID (or a RACF group to which the user ID is connected) in the access list of the OPERCMDS profile.

2. Use the RDEFINE command to create profiles for the commands:

```
RDEFINE OPERCMDS profile-name UACC(NONE)
```

where *profile-name* is:

```
subsystem-name.command[.qualifier]
```

where:

***subsystem-name***

is the name of the processing environment of the command (such as MVS, JES2, or RACF)

***command***

is the name of the command

***qualifier***

is the type of object the command specifies (JOB or SYS, for example) or an operand of the command (LIST, for example)

In these examples, you would first issue SETROPTS GENERIC(OPERCMDS) to turn generics on for the OPERCMDS class and then issue SETROPTS REFRESH:

```
RDEFINE OPERCMDS JES2.CALL.**      UACC(NONE)
RDEFINE OPERCMDS RACF.TARGET.LIST UACC(NONE)
```

**Note:**

- a. When an operator issues a command that the subsystem doesn't recognize, the subsystem checks for a profile named *subsystem-name*.UNKNOWN. To handle these commands, create a profile named:
  - MVS.UNKNOWN with UACC(READ) for MVS
  - JES2.UNKNOWN with UACC(NONE) for JES2
  - RACF.UNKNOWN with UACC(NONE) for RACF

Your security policy might require auditing of all commands issued, even if they are not valid on your system. You can audit these commands by specifying AUDIT(ALL) on these profiles.

3. For each controlled command, grant access to the users or groups who need to use it:

```
PERMIT profile-name CLASS(OPERCMDS) ID(user or group) ACCESS(access-authority)
```

For example:

```
PERMIT JES2.CALL.DSP.** CLASS(OPERCMD5) ID(OPER7 OPER24) ACCESS(UPDATE)
```

4. When you are ready to start controlling access to commands based on the profiles you have defined, activate the OPERCMD5 class:

```
SETROPTS CLASSACT(OPERCMD5) RACLIST(OPERCMD5)
```

**Example of Controlling Who Can Issue MVS Commands**

The following example shows how to use the OPERCMD5 class to control who can display and cancel jobs in your installation.

Suppose you want to let anyone display jobs but you want to restrict the task of cancelling jobs to a group of MVS operators. All of the MVS operators you want to authorize have RACF-defined user IDs connected to a group called OPERGRP.

According to *z/OS MVS Planning: Operations*, to authorize a user to issue the MVS DISPLAY JOBS command, you must give the user READ access to a resource named MVS.DISPLAY.JOB in the OPERCMD5 class. To authorize a user to issue the MVS CANCEL command for all jobs, you must give the user UPDATE access to a resource named MVS.CANCEL.JOB.\*\* in the OPERCMD5 class.

To grant these authorizations, enter:

```
SETROPTS GENERIC(OPERCMD5) REFRESH

RDEFINE  OPERCMD5 MVS.DISPLAY.JOB UACC(READ)
RDEFINE  OPERCMD5 MVS.CANCEL.JOB.** UACC(NONE)

PERMIT   MVS.CANCEL.JOB.** CLASS(OPERCMD5) ID(OPERGRP) ACCESS(UPDATE)

SETROPTS CLASSACT(OPERCMD5)
SETROPTS RACLIST(OPERCMD5) REFRESH
SETROPTS GENERIC(OPERCMD5) REFRESH
```

Now, anyone can display a job, but only the operators in OPERGRP can cancel a job.

Table 22. RACF operator command profiles: Naming conventions

Command	Resource name
DISPLAY	<p><i>subsystem-name.DISPLAY.SIGNON</i></p> <p>Any profile in the OPERCMD5 class covering this resource name protects the DISPLAY command, for example:</p> <pre>RDEFINE OPERCMD5 RACF.DISPLAY.SIGNON</pre> <p><b>Note:</b> No OPERCMD5 authority check is performed when the DISPLAY command is issued from a RACF parameter library member.</p>
RESTART	<p><i>subsystem-name.RESTART</i></p> <p>Any profile in the OPERCMD5 class covering this resource name protects the RESTART command, for example:</p> <pre>RDEFINE OPERCMD5 RACF.RESTART</pre> <p><b>Note:</b> The RESTART command cannot be issued from a RACF parameter library member.</p>

Table 22. RACF operator command profiles: Naming conventions (continued)

Command	Resource name
RVARY	<ul style="list-style-type: none"> <li>• IRR.RVARY.STATUS when the ACTIVE or INACTIVE keyword is used</li> <li>• IRR.RVARY.SWITCH when the SWITCH, DATASHARE, or NODATASHARE keyword is used.</li> </ul> <p>The resource names are the same regardless of whether RVARY is entered from TSO or as an operator command. Only successful accesses may be logged</p> <p><b>Note:</b> During recovery; RACF must not be allowed to attempt to access the database. Therefore, the ability to use OPERCMDS requires the profile(s) granting authorization have previously been RACLISTed. If the profile disallows access, or any other error occurs, the command is protected by an operator prompt. OPERCMDS protection is not used in the following situations:</p> <ul style="list-style-type: none"> <li>• The command is issued from a console that is not logged on.</li> <li>• The command is issued from a console when RACF is in failsoft mode. Note that OPERCMDS protection can work if a user is already logged on to TSO and was previously permitted.</li> </ul>
SET	<ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.SET.AUTOAPPL</li> <li>• <i>subsystem-name</i>.SET.AUTODIRECT</li> <li>• <i>subsystem-name</i>.SET.AUTOPWD</li> <li>• <i>subsystem-name</i>.SET.FULLRRSFCOMM</li> <li>• <i>subsystem-name</i>.SET.GENERICANCHOR</li> <li>• <i>subsystem-name</i>.SET.INCLUDE</li> <li>• <i>subsystem-name</i>.SET.JESNODE</li> <li>• <i>subsystem-name</i>.SET.LIST</li> <li>• <i>subsystem-name</i>.SET.PWSYNC</li> <li>• <i>subsystem-name</i>.SET.TRACE</li> </ul> <p>To protect the SET LIST command, for example:</p> <pre>RDEFINE OPERCMDS RACF.SET.LIST</pre> <p><b>Note:</b> No OPERCMDS authority check is performed when the SET command is issued from a RACF parameter library member.</p>
SIGNOFF	<p><i>subsystem-name</i>.SIGNOFF</p> <p>Any profile in the OPERCMDS class covering this resource name protects the SIGNOFF command, for example:</p> <pre>RDEFINE OPERCMDS RACF.SIGNOFF</pre> <p><b>Note:</b> No OPERCMDS authority check is performed when the SIGNOFF command is issued from a RACF parameter library member.</p>
STOP	<p><i>subsystem-name</i>.STOP</p> <p>Any profile in the OPERCMDS class covering this resource name protects the STOP command, for example:</p> <pre>RDEFINE OPERCMDS RACF.STOP</pre> <p><b>Note:</b> The STOP command cannot be issued from a RACF parameter library member.</p>

Table 22. RACF operator command profiles: Naming conventions (continued)

Command	Resource name
TARGET	<ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.TARGET.DESRIPTION</li> <li>• <i>subsystem-name</i>.TARGET.DENYINBOUND</li> </ul> <p><b>Note:</b> TARGET.DENYINBOUND also protects the ALLOWINBOUND and RESETDENYINBOUND COUNT operands.</p> <ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.TARGET.LIST</li> </ul> <p><b>Note:</b> TARGET.LIST also protects the LISTALL and LISTPROTOCOL operands.</p> <ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.TARGET.MAIN</li> <li>• <i>subsystem-name</i>.TARGET.NEWMAIN.<i>node-name</i>.<i>system-name</i></li> </ul> <p><b>Note:</b> TARGET.NEWMAIN.<i>node-name</i>.<i>system-name</i> also protects the PLEXNEWMAIN operand</p> <ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.TARGET.LOCAL</li> <li>• <i>subsystem-name</i>.TARGET.NODE</li> <li>• <i>subsystem-name</i>.TARGET.OPERATIVE</li> </ul> <p><b>Note:</b> TARGET.OPERATIVE also protects the DELETE and DORMANT operands.</p> <ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.TARGET.PREFIX</li> <li>• <i>subsystem-name</i>.TARGET.NEWPREFIX</li> </ul> <p><b>Note:</b> TARGET.NEWPREFIX also protects the NEWWORKSPACE operand.</p> <ul style="list-style-type: none"> <li>• <i>subsystem-name</i>.TARGET.PROTOCOL</li> <li>• <i>subsystem-name</i>.TARGET.PURGE</li> <li>• <i>subsystem-name</i>.TARGET.SYSNAME</li> <li>• <i>subsystem-name</i>.TARGET.WDSQUAL</li> <li>• <i>subsystem-name</i>.TARGET.WORKSPACE</li> </ul> <p>To protect the TARGET LIST command, for example:</p> <pre>RDEFINE OPERCMDS RACF.TARGET.LIST</pre> <p><b>Note:</b> No OPERCMDS authority check is performed when the TARGET command is issued from a RACF parameter library member.</p>

Table 23. RACF TSO commands entered as operator commands: Naming conventions

Command	Resource name
ADDGROUP or AG	<i>subsystem-name</i> .ADDGROUP
ADDSD or AD	<i>subsystem-name</i> .ADDSD
ADDUSER or AU	<i>subsystem-name</i> .ADDUSER
ALTDSD or ALD	<i>subsystem-name</i> .ALTDSD
ALTGROUP or AG	<i>subsystem-name</i> .ALTGROUP
ALTUSER or ALU	<i>subsystem-name</i> .ALTUSER
CONNECT or CO	<i>subsystem-name</i> .CONNECT
DELDSD or DD	<i>subsystem-name</i> .DELDSD
DELGROUP or DG	<i>subsystem-name</i> .DELGROUP

Table 23. RACF TSO commands entered as operator commands: Naming conventions (continued)

Command	Resource name
DELUSER or DU	<i>subsystem-name</i> .DELUSER
LISTDSD or LD	<i>subsystem-name</i> .LISTDSD
LISTGRP or LG	<i>subsystem-name</i> .LISTGRP
LISTUSER or LU	<i>subsystem-name</i> .LISTUSER
PASSWORD or PW or PHRASE	<i>subsystem-name</i> .PASSWORD
PERMIT or PE	<i>subsystem-name</i> .PERMIT
RACLINK	<i>subsystem-name</i> .RACLINK
RALTER or RALT	<i>subsystem-name</i> .RALTER
RDEFINE or RDEF	<i>subsystem-name</i> .RDEFINE
RDELETE or RDEL	<i>subsystem-name</i> .RDELETE
REMOVE or RE	<i>subsystem-name</i> .REMOVE
RLIST or RL	<i>subsystem-name</i> .RLIST
SEARCH or SR	<i>subsystem-name</i> .SEARCH
SETROPTS or SETR	<i>subsystem-name</i> .SETROPTS

**Note:**

1. RACF first checks that the operator issuing the TSO command is defined to RACF and if not, an error message is issued. If the operator is defined to RACF, a check is made to a profile in the OPERCMDS class to determine if the user ID has authority to issue the TSO command as an operator command. If the OPERCMDS class is not active, or if no OPERCMDS profile exists, the user will be allowed to issue the command as an operator command.
2. Existing command authorization is still enforced. For example, you must be a SPECIAL user to issue the SETROPTS INITSTATS command.
3. READ access is required to all the resource names shown in Table 23 on page 244 with the exception of SETROPTS. If SETROPTS LIST is issued with no other operands, READ access is sufficient. However, if any other SETROPTS option is issued, with or without also specifying LIST, UPDATE access is required.
4. If your installation renames any RACF TSO commands, they are still protected under the resource names shown in Table 23 on page 244. For example, if you renamed ADDGROUP as ADDBUNCH, RACF would still use *subsystem-name*.ADDGROUP as the resource name.

## Establishing security for the RACF parameter library

The RACF parameter library should be protected appropriately through the use of a DATASET profile. No OPERCMDS authority check is performed for commands issued from within a RACF parameter library member. Commands issued from within a RACF parameter library run with the authority of the RACF subsystem address space.

## Controlling message traffic

You can control whether users can receive messages sent with the TSO SEND command by defining resources in the SMESAGE class.

**Notes:**

- Use of the SMESAGE class is intended primarily as an audit mechanism for multilevel-secure environments. By itself, the SMESAGE class does not control *all* means of communication among users on the system.
- When the SMESAGE class is active and a profile does not exist for the specified user, the message request completes normally.
- When the SECLABEL class is active, the receiver of the message must pass the security label authorization check based on the receiver's current security label and the security label of the message (which was set by the sender's current security label at the time that the sender sent the message.)
- Security label checking for messages is available through the DIRAUTH class. See [z/OS Planning for Multilevel Security and the Common Criteria](#) for more information.

To control message traffic, do the following:

1. For each user for which you want to control message traffic, create a profile in the SMESAGE class:

```
RDEFINE SMESAGE receiving-userid UACC(NONE)
```

---

2. Give users the appropriate access authority:

```
PERMIT receiving-userid CLASS(SMESAGE) ID(sending-userid-or-group)  
ACCESS(access-authority)
```

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

### **NONE**

Prevents users and groups in the access list from sending messages to the user whose ID is the profile name

### **READ**

Allows users and groups in the access list to send messages to the user whose ID is the profile name.

---

3. When you are ready to start using the security provided by these profiles, activate both the SMESAGE class and SETROPTS RACLIST processing for the class. SETROPTS RACLIST processing helps ensure high performance when access authorities are checked. You can do these actions in the following command.

```
SETROPTS CLASSACT(SMESAGE) RACLIST(SMESAGE)
```

Any time you make a change to an SMESAGE profile, you must also refresh SETROPTS RACLIST processing for the SMESAGE class for the change to take effect. For example:

```
SETROPTS RACLIST(SMESAGE) REFRESH
```

---

4. Optionally, if you have implemented security labels, you can enable security label checking for messages by issuing the following command. You do not need to define any resources in the DIRAUTH class to implement security label checking.

```
SETROPTS CLASSACT(DIRAUTH)
```

---

## Controlling the opening of VTAM ACBs

You can use resources in the VTAMAPPL class to control which users can open the application control block (ACB) indicated by a VTAM application program when the user is not running an APF-authorized

program or command processor. (APF-authorized applications, such as APPC and TSO/E, do not need authorization in the VTAMAPPL class to open an ACB.)

To do this, perform the following steps:

1. Ask your VTAM system programmer for the following information:
  - The names of the VTAM application programs whose use is to be controlled
  - The names of RACF-defined users and groups who are to have access to those programs.
2. Create profiles in the VTAMAPPL class:

```
RDEFINE VTAMAPPL acb-name UACC(NONE)
```

where *acb-name* is the ACBNAME value on the APPL statement that applies to this ACB. (An ACB name is also called an LU name or a VTAM application name.)

If the ACBNAME is not specified on the APPL statement, use the name of the APPL definition statement (the ACBNAME default value). For details about ACBNAME, see [z/OS Communications Server: SNA Resource Definition Reference](#).

3. Give users and groups the appropriate access authority:

```
PERMIT acb-name CLASS(VTAMAPPL) ID(userid or group)
  ACCESS(access-authority)
```

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

**NONE**

Prevents users from opening the ACB

**READ**

Allows users to open the ACB

**UPDATE**

Is the same as READ

**CONTROL**

Is the same as READ

**ALTER**

Allows READ access, and also allows users to change the profile (if it is a discrete profile).<sup>9</sup>

4. When you are ready to start using the protection defined in the profiles, activate both the VTAMAPPL class and SETROPTS RACLIST processing for the class. You can do these two actions in one command:

```
SETROPTS CLASSACT(VTAMAPPL) RACLIST(VTAMAPPL)
```

**Note:** Any time you make a change to a VTAMAPPL profile, you must also refresh SETROPTS RACLIST processing for the VTAMAPPL class for the change to take effect.

## RACF and PSF (Print Services Facility)

If Print Services Facility for z/OS is installed, and the SECLABEL class is active, you can control how printed output is affected as follows:

- The separator pages (the job header and trailer) are always printed with the PSF identification label associated with the security label of the user data and cannot be falsified by a user. Printing of separator pages can be suppressed by the operator or the system programmer, but not by the print-job submitter.
- Data page labeling is in effect for the user data pages. This means that the PSF identification label associated with the security label of the user data is printed on all pages of printed output. By granting READ access to the PSF.DPAGELBL resource in the PSFMPL class, you can selectively allow users to stop the printing of PSF identification labels in printed output.

<sup>9</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

- On certain printers, end user data can be printed only in a system-defined user printable area. This function allows the PSF identification label to print end use data outside the system-defined user printable area.

**Note:** PSF print labeling support depends on the type of printer being used.

For specific information on PSF printers and information on using RACF to provide security for PSF, see *PSF for z/OS: Security Guide*.

## Auditing when users receive message traffic

---

You can audit when users receive data sent with the TSO SEND command or the TPUT macro, or received using the LISTBC command.

The following procedure sets up this auditing:

1. A user with the SPECIAL attribute activates the DIRAUTH class:

```
SETOPTS CLASSACT(DIRAUTH)
```

**Note:** No profiles are needed in the DIRAUTH class.

2. A user with the AUDITOR attribute requests that auditing be done each time a user receives data:

```
SETOPTS LOGOPTIONS(ALWAYS(DIRAUTH))
```

To stop auditing the DIRAUTH class, enter:

```
SETOPTS LOGOPTIONS(NEVER(DIRAUTH))
```

## RACF and APPC

---

RACF provides support to APPC in several ways:

- User verification during APPC transactions
- Support of persistent verification signed\_on\_from lists
- Protection of APPC transaction programs
- Protection of APPC server IDs (APPCSERV)
- LU security capabilities

For more information about APPC, see *z/OS MVS Planning: APPC/MVS Management*.

## User verification during APPC transactions

When APPC/MVS receives a request to allocate an APPC transaction program, it uses RACF to verify the user ID and password (if any) to be associated with that transaction.

The verification also includes checking the user's authority to use both the partner LU and the local LU to which the transaction request was routed.

### Partner LU as port of entry (POE)

You can use the APPCPORT general resource class to protect the port of entry (POE). There are two possible formats for the resource name in the APPCPORT class:

- If the APPC LU definition has enabled network-qualified names support (by specifying the NQN option on the LUADD statement), the format for the resource name is:

```
netid.luname
```

where:



**netid**

Is a network name consisting of 1 - 8 characters. The first character must be alphabetic.

**luname**

Is an LU name consisting of 1 - 8 characters.

- If APPC network-qualified names support is not enabled, the format for the resource name is:

```
luname
```

where:

**luname**

Is an LU name consisting of 1 - 8 characters. The first character must be alphabetic.

Failure to specify the correct LU name format could result in a security exposure. For more detailed information, see [z/OS MVS Planning: APPC/MVS Management](#).

## Local LU name as application (APPL)

RACF uses the APPL class to control the attach request. For more detailed information, see [z/OS MVS Planning: APPC/MVS Management](#).

## Protection of APPC/MVS transaction programs (TPs)

The security administrator can define profiles to the APPCTP class to protect APPC applications in which the outbound transaction program issues an allocate request for an inbound transaction program on MVS. For more detailed information, see [z/OS MVS Planning: APPC/MVS Management](#).

### Example:

The following example illustrates how you can use the RDEFINE command to define a transaction program profile named FINANC1.SMITH.ACCTPAYABLETP and specify a UACC of READ. A UACC of READ allows all users to access the transaction program and their keys.

```
RDEFINE APPCTP FINANC1.SMITH.ACCTPAYABLETP UACC(READ)
```

You can protect a transaction program by specifying a UACC of NONE. You can then create an access list that contains only those users who need access. The following example shows how you can define a transaction program profile named FINANC1.SMITH.ACCTPAYABLETP and give it a UACC of NONE:

```
RDEFINE APPCTP FINANC1.SMITH.ACCTPAYABLETP UACC(NONE)
```

After you protect the transaction program with a UACC of NONE, you can use the PERMIT command to define entries in the transaction program profile's access list. The following example shows how to use the PERMIT command to create entries in the access list of transaction program profile FINANC1.SMITH.ACCTPAYABLETP for users USERA and USERB, giving them each an access authority of READ:

```
PERMIT FINANC1.SMITH.ACCTPAYABLETP CLASS(APPCTP) ID(USERA USERB) ACCESS(READ)
```

The following example illustrates how you can use the RDEFINE command to define a CPI-C side information profile named TOOLS1.SYS1.SDLU1234 and specify a UACC of READ, which allows all users to read CPI-C side information.

```
RDEFINE APPCSI TOOLS1.SYS1.SDLU1234 UACC(READ)
```

You can protect CPI-C side information by specifying a UACC of NONE. You can then create an access list containing only users who need access. The following example shows how you can define a CPI-C side information profile named TOOLS1.SYS1.SDLU1234 and give it a UACC of NONE:

```
RDEFINE APPCSI TOOLS1.SYS1.SDLU1234 UACC(NONE)
```

After you protect CPI-C side information with a UACC of NONE, you can use the PERMIT command to define entries in the CPI-C side information profile's access list. The following example shows how to use the PERMIT command to create entries in the access list of CPI-C side information profile TOOLS1.SYS1.SDLU1234 for users USERA and USERB, giving them each an access authority of READ:

```
PERMIT TOOLS1.SYS1.SDLU1234 CLASS(APPCSI) ID(USERA USERB) ACCESS(READ)
```

## LU security capabilities

You can specify the conversation security you want to receive in the APPCLU profile that covers a session.

### Conversation security options

The conversation security options are stored in the CONVSEC field in the SESSION segment of the APPCLU general resource profile. CONVSEC specifies the information that is sent to the partner LU during BIND, and indicates what conversation security options are acceptable to this LU.

## Origin LU authorization

You can use the APPL general resource class to protect conversations between partner LUs. This support provides the ability to grant or deny access on the basis of the identity of both the user and the LU from which the user's request originated.

An example of how a security administrator would define origin LU authorization is as follows:

```
RDEFINE APPL local-luname UACC(NONE)
```

This command creates a RACF profile for the given LU. The specified UACC in this case would allow no user access to the LU named by *local-luname* without explicitly granted higher access authority.

Next, the security administrator could grant conditional access to a specific RACF-defined user or group whose request originates at a given partner LU with the following:

```
PERMIT local-luname CLASS(APPL) ID(userid)  
ACCESS(READ) ...  
WHEN(APPCPORT(partner-luname))
```

**Note:** There are two possible formats for the resource name in the APPCPORT class. See [“Partner LU as port of entry \(POE\)”](#) on page 248 for additional information.

In this example, you could specify ID(\*) to make LU *local-luname* accessible to anyone who is valid on the local system and whose request originates from LU *partner-luname*. Also, this example presupposes that the relevant classes have already been explicitly activated.

Using the WHEN() option puts an entry on the conditional access list of the RACF profile for *local-luname*, allowing *userid* READ access to this LU. This allows *userid* to use the local LUs services, but only when *partner-luname* is the port of entry from which the request originated.

## Protection of APPC server IDs (APPCSERV)

You can use the APPCSERV general resource class to allow authorization checking for a program running in an MVS address space that has identified itself as a server for a specific APPC/MVS transaction program. By protecting a resource in the APPCSERV class for each TP and giving READ access to authorized server IDs, you can ensure that only authorized server programs are allowed to serve a particular TP.

## RACF and CICS

If CICS is installed on your system, you can use RACF to provide security for CICS resources. For more information, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).

TXSeries® can use information from the RACF database to define user information on distributed CICS platforms. See the TXSeries document library for information.

## RACF and Db2

---

If Db2z/OS is installed on your system, you can use the DSNR general resource class for controlling access to Db2z/OS subsystems. RACF can control which users can use Db2z/OS. If you have multiple Db2z/OS subsystems running, RACF controls which users can use a specific Db2z/OS subsystem. Db2z/OS uses the user's RACF user ID in making its security decisions. Depending on the version of Db2z/OS installed, Db2z/OS can use a user's RACF group connections as secondary authorization IDs for Db2z/OS security decisions.

For information about implementing the Db2 RACF access control module to control access to Db2z/OS resources using RACF profiles with Db2z/OS for z/OS Version 8, and higher, visit [Db2 \(www.ibm.com/docs/en/db2\)](http://www.ibm.com/docs/en/db2).

For information about implementing Db2z/OS in a multilevel-secure environment, see [\*z/OS Planning for Multilevel Security and the Common Criteria\*](#).

## RACF and IMS

---

If IMS is installed on your system, you can use RACF to provide security for IMS resources. For complete information about using RACF with IMS, visit [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims).

## RACF and ICSF

---

If Integrated Cryptographic Service Facility (ICSF) is installed on your system, you can use RACF to control how ICSF cryptographic keys and services can be used.

This control is implemented by activating, RACLISTing, and defining appropriate resources in one or more RACF classes that support ICSF. For a brief description of the RACF classes that support ICSF, see [“Supplied resource classes for z/OS systems”](#) on page 683.

Profiles in RACF classes that support ICSF can contain an ICSF segment to provide enhanced export control of ICSF symmetric and asymmetric keys. For information about using the ICSF segment and using RACF classes to control ICSF keys and services, see [\*z/OS Cryptographic Services ICSF Administrator's Guide\*](#).

## RACF and z/OS UNIX

---

You can use RACF to provide additional security functions and administration for your z/OS UNIX environment. See [Chapter 21, “RACF and z/OS UNIX,”](#) on page 513.

For complete information on setting up and using RACF in the z/OS UNIX environment, see [\*z/OS UNIX System Services Planning\*](#).

## RACF support for NDS and Lotus Notes for z/OS

---

You can map Lotus Notes for z/OS short names and Novell Directory Services for OS/390 (NDS) user names to RACF user IDs. NDS and Lotus Notes for z/OS can determine the RACF user ID for a user who has been authenticated with an application user identity or a digital certificate. Once the application determines a user's RACF user ID, it might choose to use this identity for authorization checking when accessing traditional system resources, such as data sets. This allows your installation to maintain existing functions, resources, and security while consolidating application servers.

## Administering application user identities

You manage application user identities, such as Lotus Notes for z/OS short names and Novell Directory Services for OS/390 (NDS) user names, by administering user profiles. Through the ADDUSER, ALTUSER,

and DELUSER commands, you can associate a RACF user ID with an application user identity, and you can change and remove that association.

The following table shows the name of the identity segment of the user profile and the name of the user identity field in the identity segment that is used to associate application user identities to RACF user IDs.

Application	Identity segment in the user profile	User identity field
Lotus Notes for z/OS	LNOTES	SNAME
Novell Directory Services for OS/390	NDS	UNAME

Each RACF user ID can map to both a Lotus short name and an NDS user name, but no user ID can map to more than one Lotus short name or more than one NDS user name. In addition, each Lotus short name can map to only one user ID. Similarly, each NDS user name can map to only one user ID.

The application user identities that you specify in the identity segments of the user profiles must match the user identities defined by the administrators of each application. For example, the SNAME that you define for a RACF user ID must be the same short name that is defined for that user by the administrator of the Lotus Notes for z/OS application. For special considerations when selecting application user identities, see [“Considerations for application user names” on page 255](#).

## Adding application identity segments

The following example adds a new RACF user ID named CHEN, and associates it with an NDS user name for use with Novell Directory Services for OS/390.

```
ADDUSER CHEN NDS(UNAME('ChenMeiLing'))
```

ADDUSER command processing creates a new user profile named CHEN, adds an NDS segment to the user profile, and sets the UNAME field of the segment to ChenMeiLing.

## Modifying user identity segments

The following example adds an LNOTES segment to the existing user profile CHEN, and associates the RACF user ID with a short name for Lotus Notes for z/OS.

```
ALTUSER CHEN LNOTES(SNAME('ChenMeiLing'))
```

ALTUSER command processing adds an LNOTES segment to the USER profile CHEN and sets the SNAME field of the segment to ChenMeiLing.

## Removing user identity segments

The ALTUSER command can be issued to remove the association between a RACF user ID and an application user identity. The following example deletes the NDS segment from the RACF user profile named CHEN, and removes the user ID's association with the NDS user name ChenMeiLing.

```
ALTUSER CHEN NONDS
```

## System considerations

If your installation shares the RACF database with systems running releases prior to OS/390 Version 2 Release 10, your RACF support of Lotus Notes for z/OS and Novell Directory Services for OS/390 (NDS) is implemented using mapping profiles in the NOTELINK and NDSLINK classes. See [“Mapping profiles in the NOTELINK and NDSLINK classes” on page 253](#).

If your installation shares the RACF database with only systems running z/OS, or OS/390 Version 2 Release 10 or newer, you might or might not be using mapping profiles in the NOTELINK and NDSLINK class. You should see your system programmer to find out if your installation has been converted for stage 3 of application identity mapping. Stage 3 of application identity mapping uses an alias index that is

automatically maintained by RACF to map application user identities, such as Lotus short names and NDS user names, without using mapping profiles in the NOTELINK and NDSLINK classes. Once at stage 3, you can deactivate the NOTELINK and NDSLINK classes. See *z/OS Security Server RACF System Programmer's Guide* for information about running the IRRIRA00 conversion utility to convert to stage 3 of application identity mapping.

If your installation is new to RACF and you are not running any releases prior to OS/390 Version 2 Release 10, your system will automatically use application identity mapping at the stage 3 level without running the IRRIRA00 conversion utility, and there will be no mapping profiles in the NOTELINK or NDSLINK classes.

## Mapping profiles in the NOTELINK and NDSLINK classes

If your installation shares the RACF database with systems running releases prior to OS/390 Version 2 Release 10, or your installation shares the RACF database with only systems running z/OS, or OS/390 Version 2 Release 10 or newer, but has not been converted to stage 3 of application identity mapping, your RACF support of Lotus Notes for z/OS and Novell Directory Services for OS/390 may use mapping profiles.

Mapping profiles are automatically maintained through ADDUSER, ALTUSER and DELUSER command processing when NDS and LNOTES options are specified. Each mapping profile associates a RACF user ID with an application user identity, based on the information specified in the LNOTES and NDS segments of the user profile.

The profile name for mapping profiles in the NOTELINK class is the Lotus Notes for z/OS short name (SNAME). The profile name for mapping profiles in the NDSLINK class is the Novell Directory Services for OS/390 user name (UNAME). The APPLDATA field of each mapping profile contains the RACF user ID that corresponds to the application user identity. Each application identity segment of the user profile contains one user identity name. Note that when RACF creates a mapping profile as a result of an ADDUSER or ALTUSER command, the user ID of the command issuer becomes the owner of the profile.

The following examples illustrate how mapping profiles are automatically managed by RACF.

1. A mapping profile named ChenMeiLing is added in the NDSLINK class, with user ID CHEN in the APPLDATA field, as a result of executing the following command.

```
ADDUSER CHEN NDS(UNAME('ChenMeiLing'))
```

2. A mapping profile named ChenMeiLing is added in the NOTELINK class, with user ID CHEN in the APPLDATA field, as a result of executing the following command.

```
ALTUSER CHEN LNOTES(SNAME('ChenMeiLing'))
```

3. The mapping profile named ChenMeiLing is deleted from the NDSLINK class as a result of executing the following command.

```
ALTUSER CHEN NONDS
```

When ALTUSER command processing removes application identity segments from user profiles, it deletes the corresponding mapping profiles in the appropriate general resource class. Using the DELUSER command to delete a user profile that contains application identity segments will also delete the corresponding mapping profiles.

### Important:

If your installation uses mapping profiles, do not execute the DELUSER command for a user profile that contains identity segments from RACF systems that do not support identity mapping profiles. These systems do not automatically manage mapping profiles. You will inadvertently leave residual mapping profiles in a general resource class when the user profile is deleted. See information about recovery procedures in *z/OS Security Server RACF System Programmer's Guide*.

In general, you should not administer mapping profiles using the RDEFINE, RALTER, RDELETE or RLIST commands. For information on correcting mapping profiles that are inadvertently deleted or damaged, see [z/OS Security Server RACF System Programmer's Guide](#).

## Authorizing applications to use identity mapping

Applications that do not run in system key or in supervisor state require RACF authorization to be able to use the identity mapping service (IRRSIM00). Applications that run in system key or in supervisor state do not require RACF authorization.

To authorize Novell Directory Services for OS/390 (NDS) and Lotus Notes for z/OS applications, which do not run in system key or supervisor state, you must define RACF user IDs for the applications. The RACF user IDs must be given READ access to a RACF general resource called IRR.RUSERMAP in the FACILITY class.

## Defining applications as RACF users

Each NDS and Lotus Notes for z/OS server must be defined as a RACF user, if not already defined. It can run as a job or a started procedure.

The following example shows RACF user IDs (LOTUS09 and NDS14, respectively) being defined for a Lotus Notes for z/OS server and a Novell Directory Services for OS/390 server. The user IDs are members of a RACF user group called MAPGRP, and the owner for all profiles is MAPADM.

```
ADDGROUP MAPGRP OWNER(MAPADM)
ADDUSER LOTUS09 GROUP(MAPGRP) OWNER(MAPADM)
ADDUSER NDS14 GROUP(MAPGRP) OWNER(MAPADM)
```

If the application server executes as a batch job, the RACF user ID that is added is the user ID associated with the batch job. If the server executes as a started procedure, you must assign a RACF user ID using one of the following methods:

- Add the procedure name as an entry in the STARTED class. (This is the preferred method.)
- Add the procedure name in the RACF started procedure table (ICHRIN03), unless this table has already been modified by your installation to contain a generic entry.

In addition, you should assign the PROTECTED attribute to the user IDs that you associate with application servers. For more information, see [“Assigning RACF user IDs to started procedures”](#) on page 141.

## Permitting access to the IRR.RUSERMAP resource

Authorization to use the identity mapping service (IRRSIM00) is controlled through a RACF general resource called IRR.RUSERMAP in the FACILITY class. You must define a profile to protect this resource and permit application user IDs to access the resource with READ authority.

**Important:** Make sure an existing generic profile in the FACILITY class does not inadvertently grant this authority by default. Create a profile to protect the IRR.RUSERMAP resource with UACC(NONE) until you determine which applications require identity mapping.

The following example protects the IRR.RUSERMAP resource in the FACILITY class with UACC(NONE) and authorizes the group of application servers called MAPGRP to use identity mapping.

```
RDEFINE FACILITY IRR.RUSERMAP UACC(NONE)
PERMIT IRR.RUSERMAP CLASS(FACILITY) ID(MAPGRP) ACCESS(READ)
```

## Activating identity mapping

The FACILITY class must be active to enable identity mapping. If it is not already active at your installation, you must activate the FACILITY class using the SETROPTS command.

```
SETROPTS CLASSACT(FACILITY)
```



If your installation maintains FACILITY class profiles in storage through SETROPTS RACLIST processing, you must issue the following command to refresh the FACILITY class after you define or alter any profiles protecting the IRR.RUSERMAP resource.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

## Considerations for application user names

Certain application user identities, such as the Lotus Notes for z/OS short name (SNAME) and the Novell Directory Services for OS/390 (NDS) user name (UNAME), can contain blanks or lowercase letters.

Blanks are not permitted as part of a RACF profile name. Therefore, when building the profile name, ADDUSER and ALTUSER command processing will replace blanks with the X'4A' character (which often resolves to the ¢ symbol). RACF command processing also prevents the X'4A' character (¢) from being specified as part of an actual application user name.

You should use caution when specifying lowercase letters in application user names if:

- Your installation shares the RACF database among systems that support identity mapping and systems that do not.
- Administrators can issue DELUSER commands from systems that do not support identity mapping.

Residual mapping profiles might be left if DELUSER commands are issued from systems that do not support identity mapping. If the profile name contains blanks or lowercase letters, you might be unable to remove these profiles using RACF commands. See information about recovery procedures in [z/OS Security Server RACF System Programmer's Guide](#).

For information about the set of characters supported for application user names, see [z/OS Security Server RACF Command Language Reference](#).

## Storing encryption keys using the KEYSMSTR class

You can define and store encryption keys that can be used to encrypt and decrypt data stored in profiles in the RACF database. These keys are stored in the SSIGNON segment of profiles in the KEYSMSTR class. The following profiles in the KEYSMSTR class are used to hold the keys used to encrypt and decrypt the following types of passwords:

Table 24. KEYSMSTR class profiles

Profile	Purpose
DCE.PASSWORD.KEY	Contains the key used to encrypt DCE user passwords or Distributed File Service (DFS) Server Message Block (SMB) user passwords that are stored in the DCE segment of a user profile.
LDAP.BINDPW.KEY	Contains the key used to encrypt LDAP BIND passwords in the PROXY segments of USER or FACILITY class profiles for use by the z/OS LDAP server when acting as a proxy for a requester.

### Rules:

1. Each profile must be defined using a discrete profile named exactly as shown.
2. You must define an encryption key in the LDAP.BINDPW.KEY profile before you can store an LDAP BIND password in the PROXY segment of either of the following profile types:
  - a. User profiles, by using the PROXY BINDPW operands of the ADDUSER or ALTUSER commands
  - b. Resource profiles, by using the PROXY BINDPW operands of the RDEFINE or RALTER commands

Similarly, you must define an encryption key in the DCE.PASSWORD.KEY profile before users can store DCE or DFS SMB user passwords in the RACF database, or before the DCE single signon feature can be used.

**Note:** The SSIGNON segment is also used to protect PassTicket keys. Many of the considerations documented for protecting PassTicket keys also apply to the KEYSMSTR class. For details, see [“Protecting PassTicket keys”](#) on page 286 in the chapter [Chapter 11, “Using PassTickets,”](#) on page 281.

Steps for storing a key in a KEYSMSTR profile

Perform the following steps to define a KEYSMSTR profile and store an encryption key.

- 1. Choose a type of key encryption. Base your choice of encryption type on whether your system has cryptographic software, such as ICSF, installed. Two algorithm options (DES and AES) are available for encryption. AES requires that you use the KEYLABEL option and it must refer to an existing AES key label in the ICSF CKDS.

**Note:** When using a secure AES key label, make sure that the covering CSFKEYS class profile specifies SYMCPACFWRAP(YES) in the ICSF segment. This enables the key to be used as a protected key.

If you have...	Then use...	Notes
Cryptographic software installed	Key encryption (KEYENCRYPTED operand)	Cryptographic software must be active on the system when you define the KEYSMSTR profile.
No cryptographic software installed	Key masking (KEYMASKED operand)	

- 2. Create a profile in the KEYSMSTR class to define and store your encryption key, using your choice of encryption type as the operand of the SSIGNON segment.

**Example:**

```
RDEFINE KEYSMSTR LDAP.BINDPW.KEY SSIGNON(KEYENCRYPTED(0023428875DECFA))
```

In this example, LDAP BIND passwords will be encrypted using the key stored in the LDAP.BINDPW.KEY profile in the KEYSMSTR class. The value of the key is 0023428875DECFA.

**Guideline:** For security reasons, choose a key that is known only to you, the security administrator.

- 3. Display the profile you created using the RLIST command to verify that the key is protected.

```
RLIST KEYSMSTR LDAP.BINDPW.KEY SSIGNON NORACF
```

**Result:** The value of your key should not be displayed, but the information shown indicates whether the key value is masked or encrypted. If encrypted, the ICSF key label is displayed.

**Example:**

```
CLASS      NAME
-----
KEYSMSTR   LDAP.BINDPW.KEY

SSIGNON INFORMATION
-----
KEYENCRYPTED DATA NOT DISPLAYABLE
```

- 4. Activate the KEYSMSTR class.

**Example:**

```
SETROPTS CLASSACT(KEYSMSTR)
```

**Rule:** You must activate the KEYSMSTR class before RACF will use the keys stored in the KEYSMSTR profiles.



---

When you are done, the key that you stored in the SIGNON segment of the KEYSMSTR profile will be used to encrypt and decrypt LDAP passwords.

## Defining delegated resources

Some applications and daemons initiate requests that require access to resources to which the client who invoked the daemon might not otherwise need access. For example, the FTP daemon (FTPD) shipped with z/OS Communication Server requires access to sensitive ICSF resources that the FTP client does not. Generally, you must authorize the client user IDs to access resources that are needed by the daemon. However, if instructed by the documentation for the application or daemon, such as the FTP daemon, you can define a particular resource as a *delegated* resource and authorize it for use by the daemon's user ID rather than by the client user IDs.

Delegated resources are general resources that are eligible to be accessed by specially programmed applications that request RACF to check the application, or daemon's, authority for a resource when the client's authority is insufficient. Applications programmed in this way, such as the FTP daemon, are said to contain support for *nested* ACEEs because the identity of the application or daemon is said to be nested beneath the identity of the client for authorization purposes.

You indicate that a resource is a delegated resource by adding the text string RACF-DELEGATED to the APPLDATA field of the profile protecting the resource. The RACF-DELEGATED text string will always be translated to upper case by the RALTER or RDEFINE commands.

The RACF-DELEGATED text string can appear anywhere within the APPLDATA field, allowing for the existence of other information already in the field, or for new information that might be added in the future.

The following examples are commands that define a resource as a delegated resource:

```
RDEFINE CSFSERV CSFENC APPLDATA('RACF-DELEGATED')
RALTER CSFSERV CSFENC APPLDATA('RACF-DELEGATED')
RALTER CSFSERV CSFENC APPLDATA('THIS RESOURCE IS A RACF-DELEGATED RESOURCE')
```

## Steps for authorizing daemons to use delegated resources

To avoid authorizing clients to certain resources, define the resources as *delegated* and authorize the daemon rather than the end users. The following sample procedure authorizes the z/OS Communications Server FTP daemon to access the ICSF resource in the CSFSERV class.

**Before you begin:** Consult your application documentation to determine the name of the daemon and the names of the resources to be delegated. Be sure the application is written to exploit delegated resources and nested ACEEs.

1. Mark the resource as delegated by defining APPLDATA using any one of the following command examples.

- RALTER CSFSERV CSFENC APPLDATA('RACF-DELEGATED')
- If APPLDATA is already defined for this profile (this is unlikely), then enter the existing application data along with the delegated string. For example:

```
RALTER CSFSERV CSFENC APPLDATA('existing-text RACF-DELEGATED')
```

- To define all profiles within a given class as delegated, use the SEARCH command. For example:

```
SEARCH CLASS(CSFSERV) CLIST('RALTER CSFSERV ' ' APPLDATA('RACF-DELEGATED')')
EX EXEC.RACF.CLIST
```

**Restriction:** Only users with the system-SPECIAL attribute are authorized to mark a resource as delegated when SETROPTS SECLABELCONTROL is in effect and the resource has an assigned security label.

2. Authorize the daemon user ID to access the delegated resource.

```
PERMIT CSFSENC CLASS(CSFSESV) ID(FTPD) ACCESS(READ)
```

3. Optionally, if you previously authorized end users to access the delegated resource, remove their access authorities. For example:

```
PERMIT CSFSENC CLASS(CSFSESV) ID(FTPUGRP) DELETE
```

4. Refresh the CSFSERV class to activate your access changes.

```
SETROPTS RACLIST(CSFSESV) REFRESH
```

---

## Chapter 9. Limiting ALTER access authority in discrete profiles

Access to z/OS resources is controlled by DATASET and general resource profiles. These profiles contain the default access level and access lists that define a user's access to resources protected by the profile. There are several levels of access that can be granted to the protected resources. See, [Chapter 7, “Protecting data sets on DASD and tape,” on page 147](#) and [Chapter 8, “Protecting general resources,” on page 185](#) for more information.

When the profile is a discrete profile, the ALTER level of access not only grants access to the resources protected by the profile, but also allows changes to most fields in the base segment of the profile itself, including the default access and the access lists.

The security administrator can separate these capabilities by disallowing discrete profile management for users with ALTER access. This chapter describes how to do that.

---

### Summary of ALTER access

The following are ways in which a user can have ALTER access to a discrete profile:

1. A user ID access list entry with ALTER
2. A group access list entry with ALTER
3. An access list for ID(\*) with ALTER
4. A universal access of ALTER
5. ALTER access to a matching GLOBAL class member

Having ALTER access to a discrete general resource or DATASET profile allows the following administrative actions against the profile:

1. Ability to modify (RALTER, ALTDSD) the base segment of the profile, excluding the OWNER field.
2. Ability to modify the access list (PERMIT)
3. Ability to list the access list (RLIST AUTHUSER, LISTDSD AUTHUSER, R\_admin profile extract)
4. Ability to copy the access list from another profile (RDEFINE FROM, ADDSD FROM, PERMIT FROM) to which you have ALTER access
5. Ability to delete the profile (RDELETE, DELDSD)

In addition, ALTER access provides the ability to introduce a ‘conflict’ in the following situations:

- For member/grouping classes, assuming you also have class authority (CLAUTH) to the member class:
  - Ability to define a discrete member class profile (RDEFINE) when you have ALTER access to an existing entity which currently covers the discrete name. The existing entity could be in the form of:
    - a covering generic profile in the member class
    - a covering generic grouping class member
    - a matching discrete grouping class member
  - Ability to define a discrete grouping class member (RDEFINE ADDMEM, RALTER ADDMEM) when you have ALTER access to an existing entity which currently covers the discrete name. The existing entity could be in the form of:
    - a covering generic profile in the member class
    - a matching discrete profile in the member class
    - a covering generic grouping class member

- a matching discrete grouping class member
- For the Global Access Table, assuming you have authority to create or alter the GLOBAL profile itself (see below):
  - For both RDEFINE and RALTER, ability to define a discrete or generic GLOBAL profile member (ADDMEM) when you have ALTER access to an existing entity which currently covers the name. The existing entity could be in the form of:
    - a covering generic profile in the 'base' class (the class name represented by the GLOBAL class profile name) when the member is a discrete name
    - a matching profile in the base class
  - For RALTER, ability to delete a discrete GLOBAL profile member (DELMEM), as described above for defining a member.

Authority to the GLOBAL profile itself can, at a minimum, come from:

- RDEFINE: class authority (CLAUTH) to the base class
- RALTER: ALTER access to the GLOBAL profile

## Restricting ALTER access

The abilities listed above can be restricted using the `IRR.ALTER.class-name` resource in the FACILITY class. UPDATE access to `IRR.ALTER.class-name` allows these abilities. Thus, you can exclude these abilities from the general population while allowing individual users or groups to retain these abilities for specific RACF classes.

**Note:** RACF ships a sample query named "ALDS" in the IRRICE member of SYS1.SAMPLIB. This query searches for instances of ALTER access in discrete DATASET profiles. This could provide a starting point for an investigation to determine if such users or groups require the ability to manage the profile (or whether they require ALTER access at all). The query could also be modified as appropriate to search general resource profiles.

### Examples:

To completely remove the administrative capabilities conferred by ALTER access, simply define a generic profile with a universal access of NONE and no users or groups permitted:

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY) GENERIC(FACILITY)
RDEFINE FACILITY IRR.ALTER.* UACC(NONE)
SETROPTS RACLIST(FACILITY) REFRESH
```

To allow ALTER access to continue working as before for a group of application administrators, for a class specific to that application:

```
RDEFINE FACILITY IRR.ALTER.APPCLASS UACC(NONE)
PERMIT IRR.ALTER.APPCLASS CLASS(FACILITY) ID(APPGROUP) ACCESS(UPDATE)
SETROPTS RACLIST(FACILITY) REFRESH
```

### Note:

- The profile(s) controlling this behavior must start with "IRR.ALTER.". That is, a backstop FACILITY profile such as "\*", "\*\*\*", or "IRR.\*", will not be used when determining if ALTER access is honored for profile management.
- The `IRR.ALTER.class-name` profile is like any other RACF profile. If you define a discrete profile and grant ALTER authority to it, the user can change or delete the profile, and thus subvert the function. Since ALTER access is not used when checking this resource, you should avoid granting ALTER access to it for any reason.
- When adding a member to a grouping profile, the check is made as though the member were a profile in the member class. For example, a command such as:

```
RDEFINE GTERMINL GRPTERM ADDMEM(TM1 TM2)
```

results in a check for each member being added. Note that, while this results in checks that seem redundant (that is, two checks to IRR.ALTER.TERMINAL), the log string for each check will uniquely identify the resource name being checked. See, “[Logging the use of ALTER access for profile management](#)” on page 261 for more information on logging.

Additionally, standard checks apply when modifying or listing the grouping class profile itself (“GRPTERM” in this example), including when adding members. That is, in the absence of any other privilege, ALTER access to the GTERMINL profile is required to update the member list (just as it would to update the installation data field, for example).

- When adding a member to a GLOBAL class profile, the check is made as though the member were a profile in the class named by the GLOBAL profile name. For example, a command such as:

```
RDEFINE GLOBAL APPL ADDMEM(MYAPPL/READ)
```

results in a check for the “IRR.ALTER.APPL” resource.

There is no check for the GLOBAL class profile itself (in this example, “APPL”) when authorizing the specified members. However, as described above for grouping class profiles, standard checks apply when modifying or listing the GLOBAL class profile (“APPL” in this example).

## Logging the use of ALTER access for profile management

To get an idea of whether ALTER access is currently being used for profile management, you can define a generic profile to continue allowing the current behavior while creating an audit trail of the use of the privilege:

```
RDEFINE FACILITY IRR.ALTER.* UACC(UPDATE) AUDIT(SUCCESS(READ))  
SETROPTS RACLIST(FACILITY) REFRESH
```

RACF only logs successful accesses to this resource. The log string of this SMF 80 Type 2 (“access”) record contains the 8-character class name (padded with blanks) of the profile being listed, modified, or deleted, followed by a reserved byte, followed by the name of the profile being listed, modified, or deleted.

While the access record does not indicate the command that resulted in the check, the command’s Type 80 SMF record will follow the access record, assuming the command is audited.

### Exceptions:

- RACF does not log RLIST and LISTDSD commands, so in these cases, only the access record will exist (and only if the user requests that the access list be displayed).
- When the R\_admin callable service (IRRSEQ00) is used to extract a general resource or DATASET profile, and the access list is returned, an access record for the controlling FACILITY resource (IRR.RADMIN.RLIST or IRR.RADMIN.LISTDSD) will precede the access record for IRR.ALTER.\*, assuming those accesses are being logged and the caller isn’t in supervisor state and bypassing the check.

The fact that the resource check was audited does not always mean that ALTER access was the only reason the user is authorized to list or modify the profile. For example:

- When non-BASE segment keywords are specified on ALTDSD and RALTER, ALTER access will get checked for a group-SPECIAL user, or an otherwise unprivileged user, even though no BASE segment keywords are specified.
- ALTER access is checked before group authority when a discrete member class profile or discrete grouping class member conflicts with an existing name. Thus, if a user has both group-SPECIAL over the entity being checked and ALTER access to the same entity, you will see an access record for IRR.ALTER.class-name.



## Chapter 10. Administering the dynamic class descriptor table (CDT)

This topic describes how to administer class names for general resources in the RACF class descriptor table (CDT).

### Overview of the class descriptor table

The RACF class descriptor table (CDT) contains the names and attributes of the resource classes that can be used on your RACF system. There are up to three sets of class descriptor entries that comprise the CDT.

1. A required set of CDT entries supplied by IBM in assembler module ICHRRCDX
2. An optional set of entries coded by your system programmer in assembler module ICHRRCDE
3. An optional set of entries defined by you, the security administrator, by administering RACF profiles in the CDT general resource class.

Together, the supplied CDT entries in module ICHRRCDX and the installation-defined CDT entries in module ICHRRCDE are known as the *static* CDT. They are considered static entries because changes to these RACF modules are not effective until the next system IPL.

The *dynamic* CDT consists of the set of entries that you administer using RACF commands. These entries are effective without an IPL. Dynamic CDT entries are created from profiles that you define in the CDT general resource class. (The names of the profiles you define in the CDT class become new classes in the dynamic CDT.) RACF authorization checking processes the dynamic CDT as a logical extension of the static CDT.

This topic describes how to administer dynamic CDT entries using general resource profiles in the CDT class.

For information about...	See...
Syntax of RACF commands to administer profiles in the CDT class	<a href="#">z/OS Security Server RACF Command Language Reference</a>
Supplied CDT entries in ICHRRCDX	Appendix A, "Supplied RACF resource classes," on page 683
Adding and changing CDT entries in ICHRRCDE	<a href="#">z/OS Security Server RACF System Programmer's Guide</a> and your system programmer

Your installation should not change or delete entries in ICHRRCDX. You can update ICHRRCDE but use of this module is not the preferred method for adding installation-defined resources classes. If your installation has already installed and updated ICHRRCDE, you are not required to remove or update this module. However, consider migrating your static CDT entries from ICHRRCDE to the dynamic CDT. See ["Migrating to the dynamic CDT"](#) on page 276.

### Restrictions for applications and vendor products

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

## Using the dynamic CDT

Entries in the dynamic CDT are used to add, change, or delete installation-defined classes. These are optional CDT entries that are created when you define profiles in the CDT general resource class. The names of the profiles in the CDT class become the names of your new classes in the dynamic CDT.

Sample procedures for administering (adding, changing, and deleting) dynamic classes are included in this topic. The tasks of adding and changing dynamic classes use the RDEFINE and RALTER commands to define and modify attributes of CDT class profiles. You use the SETROPTS RACLIST(CDT) and SETROPTS RACLIST(CDT) REFRESH commands to build entries in the dynamic CDT. These commands effectively transform CDT profiles into RACF classes. The names of RACF classes created in this way (dynamic classes) can be used in RACF commands and the RACROUTE macro, just as you would use any other RACF class name.

Once you create the dynamic CDT by executing the SETROPTS RACLIST(CDT) command, it remains active until you disable it. (See [“Disabling the dynamic CDT”](#) on page 275.) When you restart your system, RACF automatically rebuilds the dynamic CDT using attributes from CDT class profiles in the RACF database. As with other RACF classes, if you activate SETROPTS class options for a dynamic class before a system restart, RACF automatically activates those SETROPTS class options after a restart.

**Restriction:** The number of classes you can define in the dynamic CDT is limited by the total number of entries in the class descriptor table. The maximum total number of entries is 1024 and includes entries for the following classes:

- Classes supplied by IBM in ICHRRCDX
- Classes your installation defines in ICHRRCDE
- Classes you define in the dynamic CDT.

To list all RACF classes defined on your system, including dynamic classes, you can use the Data Security Monitor (DSMON) to produce the *Class Descriptor Table Report*. See [z/OS Security Server RACF Auditor's Guide](#) for more information about DSMON.

To list all CDT class profiles on your system, execute the SEARCH CLASS(CDT) command. This list of profiles might differ from the list of dynamic classes generated by the DSMON *Class Descriptor Table Report* for one of the following reasons:

1. Some profiles in the CDT class might have been added after the most recent SETROPTS RACLIST(CDT) REFRESH command was issued. Profiles added in this way are defined on your system but are not active classes.
2. Profiles in the CDT class might have been defined with errors that prevented the classes from being added to the dynamic CDT.

## Profiles in the CDT class

The task of administering profiles in the CDT class can be done by you, the security administrator, but because changes in the dynamic CDT can affect resource classes in the static CDT and impact your entire RACF system, you are advised to consult with your system programmer before making changes. Then, you can delegate the task of administering dynamic CDT entries to the system programmer by granting class authority (CLAUTH) and field-level access to the CDTINFO profile segments.

### Example:

```
ALTUSER SYSPROG CLAUTH(CDT)
RDEFINE FIELD CDT.CDTINFO.* UACC(NONE)
PERMIT CDT.CDTINFO.* CLASS(FIELD) ID(SYSPROG) ACCESS(UPDATE)
```

See [“Field-level access checking”](#) on page 200 for the steps to enable field authority.

When you define class name entries in the dynamic CDT, you create profiles in the CDT class. These profiles define the dynamic CDT entries themselves, rather than protect resources in the classes they define. In other words, granting READ access to the profiles in the CDT class allows the user, for example, a system programmer, to view the dynamic CDT class entries. Granting ALTER access to these profiles



allows the user to change the access list or delete class name entries (Note that ALTER access can be restricted. See, Chapter 9, “Limiting ALTER access authority in discrete profiles,” on page 259 for more information). Therefore, you should control ALTER access carefully. In addition, having access to a CDT profile does not grant access to profiles in the resource class defined by that CDT entry.

The name of the profile you create in the CDT class is the name of your new class in the dynamic CDT. The syntax rules for the CDT profile name are as follows:

**Rules:**

1. The profile name must be 1 - 8 characters, consisting of the following:

**A - Z**

**0 - 9**

**#**  
(X'7B')

**@**  
(X'7C')

**\$**  
(X'5B')

2. You must include at least one character from the following:

**0 - 9**

**#**  
(X'7B')

**@**  
(X'7C')

**\$**  
(X'5B')

Rule “2” on page 265 ensures that class names you define in the dynamic CDT do not conflict with class names supplied by IBM in ICHRRCDX. If you do not follow this rule, a warning message is issued for the RDEFINE or RALTER command when you define or update a profile in the CDT class; however, the profile is still defined or updated. You can use the RDELETE command to delete the profile before you issue the SETROPTS RACLIST(CDT) command to build or refresh the dynamic CDT. If you do not delete the profile and subsequently issue the SETROPTS RACLIST(CDT) command, another warning message is issued but the class can be defined and activated if there are no other errors.

**Restriction:** If you do not follow Rule “2” on page 265 and if, in the future, IBM supplies a class using the same name you defined, the IBM class will be used instead of your class, and the results will be unpredictable.

## Adding a dynamic class with a unique POSIT value

To add a new class to the dynamic CDT, use the RDEFINE command to add a profile to the CDT class. An important attribute of every CDT entry is its POSIT value. There are 1024 possible numeric POSIT values. You can specify POSIT values 19 - 56 and 128 - 527. However, certain POSIT values are reserved for IBM use.

**Restriction:** POSIT values 0 - 18, 57 - 127, and 528 - 1023 are reserved for IBM use and should not be used for your dynamic class entries unless you intend to share SETROPTS options with an IBM supplied class. (For example, you might choose to have an installation-defined CICS class share SETROPTS options with an IBM supplied CICS class.) If you use a reserved POSIT number that is not currently used for an IBM supplied class, be aware that in the future IBM might create a supplied class with this POSIT number. If this conflict occurs, processing results for your class will be unpredictable.

Classes with the same POSIT value are administered as a single class when you specify a class option, such as CLASSACT or RACLIST, on the RACF SETROPTS command or grant CLAUTH authority to one of them. You would add a new class with a unique POSIT value when you want to administer it separately from any other class.

## Steps for adding a dynamic class with a unique POSIT value

Perform the following steps in this example to define a new class called PIX2004 that you will administer separately.

1. Determine a unique POSIT value for the new profile. Evaluate the class entries in the dynamic CDT. Consult your system programmer to evaluate the class entries in the static CDT (modules ICHRRCDX and ICHRRCDX).

- 
2. Define the new class.

**Example:**

```
RDEFINE CDT PIX2004 UACC(NONE)
  CDTINFO(DEFAULTUACC(NONE) FIRST(ALPHA) MAXLENGTH(42) OTHER(ALPHA,SPECIAL)
  POSIT(303)
  RACLIST(REQUIRED))
```

Investigate any error messages issued by the RDEFINE command; some errors can prevent the class from being added to the dynamic CDT. Use the RALTER command to correct any errors in the profile.

**Tip:** If you miss the error messages from the RDEFINE command, you can use the CDTINFO keyword, with no suboperands, on the RALTER command to initiate validation checking of the fields again. For example, to initiate validation for the command in this step, you can execute the following command.

```
RALTER CDT PIX2004 CDTINFO
```

Validation checking will be performed again, and any error messages will be issued again.

- 
3. Create the dynamic CDT.

```
SETROPTS CLASSACT(CDT) RACLIST(CDT)
```

Or, if the dynamic CDT was already active, refresh the dynamic CDT.

```
SETROPTS RACLIST(CDT) REFRESH
```

Again, investigate any error messages issued by the SETROPTS command because some errors can prevent the class from being added to the dynamic CDT.

If you do not complete this step before proceeding, you will receive the following message when you execute the RDEFINE commands in Step [“4”](#) on page 266.

```
IKJ56702I INVALID CLASS, PIX2004
```

- 
4. Define profiles in the new class, as needed.

**Example:**

```
RDEFINE PIX2004 JANUARY.CATLG UACC(NONE)
RDEFINE PIX2004 FEBRUARY.CATLG UACC(NONE)
```

- 
5. Activate and RACLIST the new class.

SETROPTS CLASSACT (PIX2004) RACLIST (PIX2004)

## Adding a dynamic class that shares a POSIT value

To add a new class to the dynamic CDT that you will administer together with another class, add the new class with the same POSIT value as the other class. (See [“Processing options that are controlled by a shared POSIT value”](#) on page 267 for details about the RACF processing options that are controlled together for classes that share a POSIT value.)

For example, if you have an existing class called PONIES8 (either in the dynamic CDT or in ICHRRCDE) with a unique POSIT number (301, for example), you might add a new class called HORSES8, a class related to PONIES8, and logically requiring the same RACF processing options.

Assume that you have already activated the following SETROPTS options for the existing PONIES8 class:

- CLASSACT
- RACLIST
- AUDIT

When you execute the RDEFINE CDT command to add the new HORSES8 class to the CDT, specify the POSIT number as 301 (the same as for PONIES8). When you refresh the dynamic CDT, all of the same RACF processing options that are in effect for class PONIES8 will automatically be in effect for the new class HORSES8, except SETROPTS RACLIST. The SETROPTS RACLIST(HORSES8) command must be issued separately for the HORSES8 class because a new dataspace must be built.

### Rules:

1. If you want SETROPTS RACLIST active for a new class, you must execute the SETROPTS RACLIST command after you define the new class to build its new associated dataspace.
2. If SETROPTS GENLIST is active for a new class, you must execute the SETROPTS GENLIST command after you define the new class to build its associated in-storage profiles.

After the dataspace has been built initially, you can issue either one of the following commands to refresh RACLISTed profiles in both the HORSES8 and PONIES8 classes.

- SETROPTS RACLIST(HORSES8) REFRESH or
- SETROPTS RACLIST(PONIES8) REFRESH

Further, by issuing either one of the following commands, you activate global access checking for both the PONIES8 and the HORSES8 classes.

- SETROPTS GLOBAL (HORSES8) or
- SETROPTS GLOBAL (PONIES8)

Similarly, by issuing either one of the following commands, you activate STATISTICS for both the PONIES8 and the HORSES8 classes.

- SETROPTS STATISTICS(PONIES8) or
- SETROPTS STATISTICS(HORSES8)

Any number of classes can share the same POSIT number. For example, a third class called MARES8 could be added and could also share POSIT number 301 with PONIES8 and HORSES8.

## Processing options that are controlled by a shared POSIT value

When a POSIT value is shared between two or more classes, certain RACF processing options are controlled in the same manner (simultaneously) for all classes with the shared POSIT value. The following options affect the processing of resources or profiles associated with classes that share a POSIT value:

Table 25. Processing options controlled simultaneously for classes sharing a POSIT value

Type of processing	Corresponding option
Authorization checking	SETROPTS CLASSACT
Auditing	SETROPTS AUDIT
Statistics	SETROPTS STATISTICS
Generic profile access checking	SETROPTS GENERIC
Generic command processing	SETROPTS GENCMD
Global access checking	SETROPTS GLOBAL
Special resource access auditing	SETROPTS LOGOPTIONS
Authorization checking in a dataspace	SETROPTS RACLIST
Class authority (whether a user has CLAUTH)	ALTUSER <i>userid</i> CLAUTH

## Rules about disallowing generics when sharing a POSIT value

1. All classes with a shared POSIT value must be defined with the same GENERIC setting. This is because the SETROPTS GENERIC and SETROPTS GENCMD commands process all classes that share a POSIT number.
2. If your new dynamic class does not have the same GENERIC setting as the rest of the classes sharing the POSIT value, RACF will issue a warning message during SETROPTS RACLIST(CDT) processing and dynamically change the GENERIC setting of one or more classes sharing the POSIT value.
  - If your new class shares a POSIT number with a supplied class, RACF changes the GENERIC setting of your new class to match the supplied class. (The class attribute in the supplied class takes precedence.)
  - If your new class shares a POSIT number with installation-defined classes (static or dynamic), RACF determines the least restrictive attribute (GENERIC(ALLOWED) is less restrictive than GENERIC(DISALLOWED)) and changes the GENERIC(DISALLOWED) class attributes to GENERIC(ALLOWED).

**Exception:** A grouping class and member class can share a POSIT number although their GENERIC keyword values need not match. You must specify GENERIC(DISALLOWED) for grouping classes. However, you can specify either ALLOWED or DISALLOWED for member classes.

## Steps for adding a dynamic class with a shared POSIT value

Perform the steps in this example to define a new class called HORSES8 that you will administer together with the class called PONIES8 and that will share its POSIT value.

1. Define the new class:

```
RDEFINE CDT HORSES8 UACC(NONE)
CDTINFO(DEFAULTUACC(NONE) FIRST(ALPHA) MAXLENGTH(200) OTHER(ALPHA,SPECIAL)
POSIT(301)
RACLIST(REQUIRED))
```

2. Refresh the dynamic CDT:

```
SETROPTS RACLIST(CDT) REFRESH
```

**Result:** The same SETROPTS options that were previously active for the PONIES8 class are now active for the HORSES8 class because the classes share a POSIT value. The exception is SETROPTS RACLIST.

3. Define some profiles in the new class:

```
RDEFINE HORSES8 PALOMINO.FRANKE UACC(NONE)
RDEFINE HORSES8 APPALOOSA.ANDRE UACC(NONE)
```

4. RACLIST the profiles in the HORSES8 class:

```
SETROPTS RACLIST(HORSES8)
```

## Changing a POSIT value for a dynamic class

Before changing the current POSIT value of an existing class in the dynamic CDT, plan carefully because changing a POSIT value could cause unintended results. For example, you could inadvertently deactivate a class if you change its POSIT value to a POSIT value shared with a class that is not active. If you change a POSIT value, perform the following procedure.

### Steps for changing a POSIT value of an existing dynamic class

Perform the following steps to change the current POSIT value of an existing class:

1. Execute the following command to list the SETROPTS options for all classes.

```
SETROPTS LIST
```

Record all active system options for the class that you want to change.

2. Examine all dynamic and static CDT entries to see if any other existing class shares the current POSIT value or the new POSIT value.

- If no other existing class shares the current POSIT value, use the SETROPTS command to deactivate all options associated with the class you want to change to ensure that any new class will not have unexpected options if you add a new class using that POSIT value in the future. For example, if the SETROPTS options CLASSACT, STATISTICS, GENERIC and GENCMD are active for the class, you would issue the following command to deactivate those options.

```
SETROPTS NOCLASSACT(classname) NOSTATISTICS(classname)
NOGENERIC(classname) NOGENCMD(classname)
```

- If another existing class shares the current POSIT value, do not use the SETROPTS command to deactivate any SETROPTS option associated with the class you want to change because this would turn off the SETROPTS option for all classes sharing the POSIT value. In this case, proceed to Step “3” on page 269 without making any changes.

3. Change the POSIT value.

```
RALTER CDT classname CDTINFO(POSIT(new-posit-value))
```

4. Refresh the CDT class on all systems that will use the changed class.

```
SETROPTS RACLIST(CDT) REFRESH
```

5. Activate the desired SETROPTS options, using the SETROPTS LIST output from Step “1” on page 269 as reference, assuming for this example that SETROPTS options CLASSACT, STATISTICS, GENERIC, and GENCMD were previously active for the class you changed.

- If an existing class does *not* share the new POSIT value, issue the following command to reactivate those options.

```
SETOPTS CLASSACT(classname) STATISTICS(classname)
GENERIC(classname) GENCMD(classname)
```

- If an existing class shares the new POSIT value, all of the same RACF processing options that were in effect for the shared class are automatically in effect for the class you changed, with the exception of the SETROPTS RACLIST option.

**Rules:**

- a. If you want SETROPTS RACLIST active for a changed class, you must execute the SETROPTS RACLIST command after you change the class to build its new associated dataspace. If the class was RACLISTed before the POSIT change, add the REFRESH option to rebuild the dataspace.

```
SETOPTS RACLIST(classname) REFRESH
```

- b. If SETROPTS GENLIST is active for the changed class, you must execute the SETROPTS GENERIC REFRESH command after you change the class to refresh its associated in-storage profiles.

```
SETOPTS GENERIC(classname) REFRESH
```

---

## Guidelines for changing dynamic CDT entries

---

If you change attributes for an existing class in the dynamic CDT, plan carefully to avoid unintended results. Before making changes to the following particular class attributes, follow these guidelines to help you consider the effect of your changes on existing applications and RACF processing.

After you change an attribute for a class, refresh the CDT class on all systems that will use the changed class:

```
SETOPTS RACLIST(CDT) REFRESH
```

### Guidelines for changing particular class attributes:

1. **CASE:** Before you change CASE (ASIS) to CASE (UPPER), review all profiles in the class and delete any profiles that contain lowercase letters in the profile name.
2. **FIRST or OTHER:** Before you change the FIRST and OTHER values to make them more restrictive, review all profiles in the class and delete those that contain the less restrictive characters. **Example:** Before you change FIRST (ALPHA, NUMERIC) to FIRST (ALPHA), delete any profiles that contain a number in the first character of the profile name.
3. **GENERIC:** Before you change GENERIC (ALLOWED) to GENERIC (DISALLOWED), review [“Rules about disallowing generics when sharing a POSIT value” on page 268](#) and follow [“Steps for changing a dynamic class to disallow generic profiles” on page 272](#).
4. **GENLIST:** Before you change GENLIST (ALLOWED) to GENLIST (DISALLOWED), remove any in-storage generic profiles in the class by issuing the following command.

```
SETOPTS NOGENLIST(classname)
```

Do not issue SETROPTS NOGENLIST for an existing class when it shares a POSIT value with other classes that are active. This action might impact authorization processing for the other classes.

5. **GROUP and MEMBER:** Before you change the GROUP or MEMBER attributes for a class, delete any member classes by issuing the following command.

```
RALTER grouping-classname profile-name DELMEM(member-list)
```

Then, be sure to update both the grouping class definition and the member class definition in compatible ways. For example, if you change a member class to a non-member class by changing

GROUP(*grouping-classname*) to NOGROUP, then you must change the corresponding grouping class to a non-grouping class by changing MEMBER(*member-classname*) to NOMEMBER.

After you change the GROUP or MEMBER attribute for your class and refresh the CDT class, refresh the in-storage profiles for your class. If you do not refresh the in-storage profiles, RACF authorization checking for resources in your class might return unpredictable results.

- If your class is RACLISTed, refresh using the SETROPTS RACLIST command.

```
SETROPTS RACLIST(classname) REFRESH
```

- If your class is GENLISTed, refresh using the SETROPTS GENERIC command.

```
SETROPTS GENERIC(classname) REFRESH
```

6. **MAXLENGTH** and **MAXLENX**: Before you change the length of profile names, review the following guidelines.

- Before you *increase* the length of profile names in a class, examine existing applications to see if modifications are necessary. When you increase the length attribute in a class, update only the MAXLENX operand to specify the new length, leaving the existing MAXLENGTH value. This might allow some applications using the RACROUTE macro with the ENTITY operand to work properly with the previous maximum length. These applications include those that use RACROUTE REQUEST=AUTH, REQUEST=FASTAUTH, and REQUEST=EXTRACT with TYPE (other than EXTRACTN). However, applications that use other RACF macros, such as ICHEINTY and RACROUTE REQUEST=EXTRACT,TYPE=EXTRACTN, might likely need modifications to process the new profile name length correctly. Modify applications that use the RACROUTE macro to include the ENTITYX operand to support the new maximum length.
- Before you *decrease* the length of profile names in a class, examine existing applications that use the RACROUTE macro with ENTITY or ENTITYX option, or the ICHEINTY macro with the ENTRY or ENTRYX option, to see if modifications are necessary. Then, be sure to delete any profiles in that class that have names longer than the new MAXLENGTH or MAXLENX limits. If you do not delete the profiles with the longer names before you decrease the MAXLENGTH or MAXLENX values for the class, authorization checking for resources in the class might give unpredictable results.

After you change the MAXLENGTH or MAXLENX attribute for your class and refresh the CDT class, refresh the in-storage profiles for your class. If you do not refresh the in-storage profiles, RACF authorization checking for resources in your class might return unpredictable results.

- If your class is RACLISTed, refresh using the SETROPTS RACLIST command.

```
SETROPTS RACLIST(classname) REFRESH
```

- If your class is GENLISTed, refresh using the SETROPTS GENERIC command.

```
SETROPTS GENERIC(classname) REFRESH
```

7. **PROFILESALLOWED**: Before you change PROFILESALLOWED(YES) to PROFILESALLOWED(NO), delete all profiles from the class using the RDELETE command.

8. **RACLIST**: Before you change RACLIST(REQUIRED) or RACLIST(ALLOWED) to RACLIST(DISALLOWED), do both of the following.

- Remove any RACLISTed profiles in the class by issuing the following command.

```
SETROPTS NORACLIST(classname)
```

Do not issue SETROPTS NORACLIST for an existing class when it shares a POSIT value with other classes that are active. This action might impact authorization processing for the other classes.

- Modify any applications that process RACLISTed profiles in that class.

**Testing consideration:** Consider how you can test your changes. If you have a test system that does not share a RACF database with your production system, you might be able to test the change with existing applications and RACF commands on your test system before activating the change on a production

system. If your test system does share a RACF database with your production system, you might need to create a class in the dynamic CDT specifically for testing, and modify your applications to use the test class.

**RRSF consideration:** If you are using RACF remote sharing facility (RRSF) and change a dynamic CDT entry, consider updating the dynamic CDT entry on all systems that use the dynamic class. Also, see [“RRSF considerations for the dynamic CDT” on page 279.](#)

## Defining a dynamic class with generics disallowed

You can add a dynamic class with GENERIC(DISALLOWED) to prevent generic profiles for resources in this class.

### Example:

```
RDEFINE CDT PIX2006 CDTINFO(POSIT(334) GENERIC(DISALLOWED))
```

Before you add a dynamic class with GENERIC(DISALLOWED), determine whether the new class shares a POSIT value with other classes. If so, ensure that all other classes sharing the POSIT value also have GENERIC(DISALLOWED). See [“Rules about disallowing generics when sharing a POSIT value” on page 268.](#)

## Steps for changing a dynamic class to disallow generic profiles

### Before you begin:

- Determine if the dynamic class that you want to change to GENERIC(DISALLOWED) shares a POSIT value with other classes. If so, determine if the other classes sharing the POSIT value also have GENERIC(DISALLOWED). (See [“Rules about disallowing generics when sharing a POSIT value” on page 268.](#))
- Do not perform these steps if other classes sharing the POSIT value have GENERIC(ALLOWED) and you do *not* want to change those classes. Instead, first change this class to a unique POSIT value or to a POSIT value shared with classes that have GENERIC(DISALLOWED).

Perform the following steps to change an existing dynamic class called HORSES8 from GENERIC(ALLOWED) to GENERIC(DISALLOWED).

1. Delete all generic profiles in the HORSES8 class. To do this:

- a. Execute the SEARCH command to create a CLIST containing a command to delete each generic profile in the class.

```
SEARCH CLASS(HORSES8) GENERIC CLIST('RDELETE HORSES8 ')
```

- b. Execute the CLIST.

```
EXEC EXEC.RACF.CLIST LIST
```

- c. Verify that no generic profiles remain in the class.

```
SEARCH CLASS(HORSES8) GENERIC
```

2. If the class shares a POSIT value, repeat Step [“1” on page 272](#) for each class sharing the POSIT value.

3. Deactivate generic processing for the HORSES8 class.

```
SETROPTS NOGENERIC(HORSES8) NOGENCMD(HORSES8)
```

If your class shares a POSIT value with other active classes, this command deactivates generic processing for those classes as well.



- 
4. Alter the HORSES8 class to prevent generic profiles.

```
RALTER CDT HORSES8 CDTINFO(GENERIC(DISALLOWED))
```

- 
5. If the class shares a POSIT value, repeat Step “4” on page 273 for each class sharing the POSIT value.

- 
6. Refresh the in-storage profiles for the CDT class.

```
SETROPTS RACLIST(CDT) REFRESH
```

---

When you finish, you have changed an existing dynamic class, and all classes sharing its POSIT value, from GENERIC(ALLOWED) to GENERIC(DISALLOWED). You have also deleted all generic profiles from all classes sharing the POSIT value.

## Deleting a class from the dynamic CDT

You can delete a class entry from the dynamic class descriptor table by deleting the class profile from the CDT class and refreshing the dynamic CDT. If the class is active, you should deactivate the class before deleting the CDT profile and refreshing the dynamic CDT. If you do not deactivate the class, then if you create a class with the same POSIT value in the future, the class will automatically be active. The same consideration applies to each SETROPTS option controlled by a shared POSIT value. (See the table in “Processing options that are controlled by a shared POSIT value” on page 267.)

### Steps for deleting a dynamic CDT class

**Restriction:** The following procedure cannot be used to delete classes from the static CDT (modules ICHRRCDX or ICHRRCDE). To modify the static CDT, consult your system programmer and see [z/OS Security Server RACF System Programmer's Guide](#).

#### Before you begin:

- If you have applications that use resources in the dynamic class, those applications, such as those issuing RACROUTE REQUEST=LIST,GLOBAL=YES for the class, should be changed or removed. Otherwise, the applications could fail after you remove the class from the dynamic CDT.
- Evaluate the uniqueness of the POSIT value of the class to be deleted.
  - If the POSIT value is *unique*, follow the steps below to deactivate all SETROPTS options.
  - If the POSIT value is *shared*, some of the steps below should *not* be executed and they are so noted. If those steps were executed, the SETROPTS options for all classes that share the POSIT value with the deleted class would be deactivated. This would have unintended effects on those classes.

Perform the following steps to delete an existing class from the dynamic CDT.

1. Delete all profiles in the class to be deleted.
  - a. Execute a SEARCH command to create a CLIST with a command to delete each profile in the class.

#### Example:

```
SEARCH CLASS(HORSES8) CLIST('RDELETE HORSES8 ')
```

- 
- b. Execute the CLIST created in Step “1.a” on page 273.

**Example:**

```
EXEC EXEC.RACF.CLIST LIST
```

- 
- c. Verify no profiles remain in the class.

**Example:**

```
SEARCH CLASS(HORSES8)
```

- 
2. Issue the following command and note every occurrence of the class you want to delete.

```
SETOPTS LIST
```

- 
3. If the class to be deleted does *not* share a POSIT value with other existing classes, deactivate the class.

**Example:**

```
SETOPTS NOCLASSACT(HORSES8)
```

Do not deactivate this class when it shares a POSIT value with other classes that are active. (See the **"Before you begin"** topic of this procedure.)

- 
4. If you are using global access checking for the class and the class to be deleted does *not* share a POSIT value with other existing classes, deactivate the GLOBAL option for the class.

**Example:**

```
SETOPTS NOGLOBAL(HORSES8)
```

Do not deactivate the GLOBAL option for this class when it shares a POSIT value with other classes that are active. (See the **"Before you begin"** topic of this procedure.)

- 
5. If you have a GLOBAL profile for the class, delete it.

**Example:**

```
RDELETE GLOBAL HORSES8
```

- 
6. If you have a RACGLIST profile for the class, delete it.

**Example:**

```
RDELETE RACGLIST HORSES8
```

- 
7. If the class to be deleted does *not* share a POSIT value with other existing classes, deactivate the other active system options for your class, using the SETOPTS LIST command output from Step [“2”](#) on [page 274](#).

**Example:**

```
SETOPTS NOAUDIT(HORSES8) LOGOPTIONS(DEFAULT(HORSES8)) NORACLIST(HORSES8)
      NOGENERIC(HORSES8) NOGENCMD(HORSES8) NOSTATISTICS(HORSES8)
```

Do not deactivate the active system options for this class when it shares a POSIT value with other classes that are active. (See the **"Before you begin"** topic of this procedure.)

- 
8. If you are using GENLIST processing for the class to be deleted and the class does *not* share a POSIT value with other existing classes, deactivate GENLIST processing.

**Example:**

```
SETROPTS NOGENLIST(HORSES8)
```

Do not deactivate GENLIST processing for this class when it shares a POSIT value with other classes that are active. (See the **"Before you begin"** topic of this procedure.)

- 
9. Delete the class from the CDT class.

**Example:**

```
RDELETE CDT HORSES8
```

If you receive message ICH12304I indicating that the class cannot be deleted because there are profiles in the class, your RACF database might contain generic profiles in this class that are hidden from the SEARCH and RLIST commands. This can happen when a generic profile is defined in a class that is subsequently disabled for generics with the SETROPTS NOGENCMD or NOGENERIC command. To resolve this, schedule an appropriate time to issue the SETROPTS GENCMD command and then repeat Step “1” on page 273 to find and delete such profiles. After you successfully delete the profiles, issue the SETROPTS NOGENCMD command. Be sure to carefully plan when to enable the GENCMD option because it will affect other classes that share the same POSIT value.

- 
10. Refresh the dynamic CDT.

```
SETROPTS RACLIST(CDT) REFRESH
```

- 
11. If you have users with class authority (CLAUTH) for the deleted class, remove their authorities.

**Example:**

```
ALTUSER userid NOCLAUTH(HORSES8)
```

---

## Disabling the dynamic CDT

You can disable the dynamic CDT only after you have determined that no applications are using dynamic classes and after you have deleted each dynamic class by executing the procedure defined in [“Deleting a class from the dynamic CDT”](#) on page 273. After you have deleted all dynamic classes, deactivate and disable the use of the dynamic CDT by issuing the following command.

```
SETROPTS NORACLIST(CDT) NOCLASSACT(CDT)
```

**Restriction:** If you do not delete a dynamic class before you issue the SETROPTS NORACLIST(CDT) command, RACF no longer recognizes that dynamic class. If you subsequently enable the dynamic CDT using the SETROPTS RACLIST(CDT) command, use of the previously defined dynamic class might cause unpredictable results. Any profiles in the previously defined dynamic class will remain and some SETROPTS options might still remain active, but RACLIST options might no longer be active. To re-enable a previously defined dynamic class, see [“Re-enabling a previously defined dynamic class”](#) on page 276.

## Re-enabling a previously defined dynamic class

---

If you erroneously disable the CDT class (by issuing the SETROPTS NORACLIST(CDT) command) before you delete all dynamic classes, you can restore use of the deleted dynamic classes by executing the following procedure.

### Steps to re-enable a previously defined dynamic class

1. If SETROPTS RACLIST was active for the dynamic class (before you issued the SETROPTS NORACLIST(CDT) command), rebuild the profiles in storage for the class.

**Example:** SETROPTS RACLIST(HORSES8)

---

2. If GLOBAL=YES RACLIST ONLY was active for the dynamic class (before you issued the SETROPTS NORACLIST(CDT) command), the application that issued the RACROUTE REQUEST=LIST,GLOBAL=YES macro must reissue the macro to rebuild the profiles in storage for the class.
- 

3. Issue the SETROPTS LIST command and carefully review the output, noting whether the dynamic class you want to restore appears in any of the following lists.

- GLOBAL CHECKING CLASSES
- GENERIC PROFILE CLASSES
- GENLIST CLASSES

- a. If SETROPTS GLOBAL is active for your class, refresh the global access checking table for the class.

**Example:** SETROPTS GLOBAL(HORSES8) REFRESH

---

- b. If SETROPTS GENERIC or SETROPTS GENLIST is active for your class, refresh the generic profiles in storage for the class.

**Example:** SETROPTS GENERIC(HORSES8) REFRESH

---

You have now re-enabled the use of a previously defined dynamic class that inadvertently remained when the dynamic CDT was disabled. The options that were previously active for the class should again be active.

## Migrating to the dynamic CDT

---

**Restriction:** You cannot move *supplied* classes (ICHRRCDX) to the dynamic CDT. You can only migrate classes from the *installation*-defined CDT (ICHRRCDE) to the dynamic CDT.

Because dynamic CDT entries can be changed without an IPL, you should consider migrating your static installation-defined classes to the dynamic CDT. The RACF Web site provides a REXX exec to translate your installation-defined CDT into a series of RDEFINE CDT commands to facilitate this migration. Look for this download at the [RACF home page \(www.ibm.com/products/resource-access-control-facility/resources\)](http://www.ibm.com/products/resource-access-control-facility/resources).

If you do not use the CDT migration exec, define classes in the dynamic CDT to replace the same-named class in ICHRRCDE. Use class attributes on the RDEFINE CDT command that match the equivalent class attributes for each class on the ICHERCDE macro invocation (used to create ICHRRCDE). Use [Table 26 on page 277](#) to determine the equivalent class attributes. (If you use the REXX EXEC to create the RDEFINE CDT commands, this translation is done for you.) If you choose class attributes on the RDEFINE CDT command that do not match the equivalent class attributes on your ICHERCDE macro invocation for a class, a warning message is issued to note the attribute differences.

When you issue an RDEFINE CDT command to define a class that already exists in ICHRRCDE, a warning message is issued to remind you that a duplicate entry exists in ICHRRCDE. When you add the class to the dynamic CDT during SETROPTS RACLIST(CDT) or SETROPTS RACLIST(CDT) REFRESH command processing, another warning message is issued to indicate the class definition in the dynamic CDT overrides the definition in the static CDT. If you subsequently delete the entry in the dynamic CDT, the class definition in the static CDT will again be in effect, and another message will indicate this.

#### Rules:

- If you are replacing a grouping or member class from the installation-defined CDT with a dynamic class, you must specify the equivalent GROUP or MEMBER operand on the definition of the dynamic class. If the grouping or member class definition does not match, an error message is issued. For example, if your installation-defined class HORSES8 is a grouping class that specifies the member class PONIES8 (MEMBER=PONIES8 is specified on the ICHERCDE macro), then your dynamic class definition for HORSES8 must include the CDTINFO(MEMBER(PONIES8)) operand.
- When you move a grouping or member class from ICHRRCDE to the dynamic CDT, you must define both the grouping and member class to the CDT class before issuing SETROPTS RACLIST(CDT) to build or refresh the dynamic CDT. A grouping class in the dynamic CDT cannot reference a member class in the static CDT. Similarly, a member class in the dynamic CDT cannot reference a grouping class in the static CDT.

See [Table 26 on page 277](#) for a comparison of the class attributes of the ICHERCDE macro and the corresponding class attributes to specify when you define dynamic class entries in the CDT class.

Table 26. ICHERCDE macro operands and the corresponding operands for the RDEFINE and RALTER commands

ICHERCDE macro operand	Corresponding RDEFINE/RALTER operand
CLASS=	<i>profile-name</i>
CASE=UPPER   ASIS	CDTINFO(CASE(UPPER   ASIS))
DFTRETC=0   4   8	CDTINFO(DEFAULTTRC(0   4   8))
DFTUACC=ALTER   CONTROL   UPDATE   READ   NONE	CDTINFO(DEFAULTTUACC(ACEE   ALTER   CONTROL   UPDATE   READ   NONE)) <a href="#">“1” on page 278</a>
EQUALMAC=YES   NO	CDTINFO(MACPROCESSING(NORMAL   EQUAL))
FIRST=ALPHA	CDTINFO(FIRST(ALPHA,NATIONAL))
FIRST=NUMERIC	CDTINFO(FIRST(NUMERIC))
FIRST=ALPHANUM	CDTINFO(FIRST(ALPHA,NUMERIC,NATIONAL))
FIRST=ANY	CDTINFO(FIRST(ALPHA,NUMERIC,NATIONAL,SPECIAL))
FIRST=NONATABC	CDTINFO(FIRST(ALPHA))
FIRST=NONATNUM	CDTINFO(FIRST(ALPHA,NUMERIC))
GENERIC=ALLOWED   DISALLOWED	CDTINFO(GENERIC(ALLOWED   DISALLOWED))
GENLIST=ALLOWED   DISALLOWED	CDTINFO(GENLIST(ALLOWED   DISALLOWED))
GROUP= <i>grouping-classname</i>	CDTINFO(GROUP( <i>grouping-classname</i> ))
ID= <i>number</i>	None. <a href="#">“2” on page 278</a>
KEYQUAL= <i>nnn</i>	CDTINFO(KEYQUALIFIERS( <i>nnn</i> ))
MAXLENX= <i>nnn</i>	CDTINFO(MAXLENX( <i>nnn</i> ))
MAXLNTH= <i>nnn</i>	CDTINFO(MAXLENGTH( <i>nnn</i> ))
MEMBER= <i>member-classname</i>	CDTINFO(MEMBER( <i>member-classname</i> ))
OPER=YES   NO	CDTINFO(OPERATIONS(YES   NO)) <a href="#">“3” on page 278</a>
OTHER=ALPHA	CDTINFO(OTHER(ALPHA,NATIONAL))

Table 26. ICHERCDE macro operands and the corresponding operands for the RDEFINE and RALTER commands (continued)

ICHERCDE macro operand	Corresponding RDEFINE/RALTER operand
OTHER=NUMERIC	CDTINFO(OTHER(NUMERIC))
OTHER=ALPHANUM	CDTINFO(OTHER(ALPHA,NUMERIC,NATIONAL))
OTHER=ANY	CDTINFO(OTHER(ALPHA,NUMERIC,NATIONAL,SPECIAL))
OTHER=NONATABC	CDTINFO(OTHER(ALPHA))
OTHER=NONATNUM	CDTINFO(OTHER(ALPHA,NUMERIC))
POSIT= <i>nnn</i>	CDTINFO(POSIT( <i>nnn</i> ))
PROFDEF=YES   NO	CDTINFO(PROFILESALLOWED(YES   NO))
RACLIST=ALLOWED   DISALLOWED	CDTINFO(RACLIST(ALLOWED   DISALLOWED))
RACLREQ=YES   NO	CDTINFO(RACLIST(REQUIRED))
RVRSMAC=YES   NO	CDTINFO(MACPROCESSING(NORMAL   REVERSE))
SIGNAL=YES   NO	CDTINFO(SIGNAL(YES   NO))
SLBLREQ=YES   NO	CDTINFO(SECLABELSREQUIRED(YES   NO))

**Note:**

1. If you do not specify the DEFAULTUACC operand, the default is DEFAULTUACC(NONE) which is different from the ICHERCDE default of using the ACEE value.
2. The ID operand is not applicable for use with dynamic CDT.
3. If you do not specify the OPERATIONS operand, the default is OPERATIONS(NO) which is different from the ICHERCDE default of OPER=YES.

## Sysplex considerations for the dynamic CDT

You can use the dynamic CDT in a sysplex environment where some systems are not using the dynamic CDT.

When RACF is enabled for sysplex communications, RACF propagates the following commands to the rest of the sysplex.

- SETROPTS RACLIST(CDT)
- SETROPTS RACLIST(CDT) REFRESH
- SETROPTS NORACLIST(CDT)

This propagation simplifies security management for resource classes in the dynamic CDT.

When RACF is enabled for sysplex communications, RACF propagates the preceding listed commands to the members of the data-sharing group even when the command *fails* on the system where it was issued. If the command fails on any of the member systems, RACF does not back out or undo the command execution from the member systems where the command did not fail. This allows you to use the dynamic CDT in a sysplex environment where some systems are downlevel or are not using the dynamic CDT.

When you move a static class to the dynamic CDT in a sysplex environment and the static class is defined with different options on various systems, you will receive different warning messages on each system. Examine the message log on each peer system when you execute the SETROPTS RACLIST(CDT) REFRESH command to ensure that each execution completed as expected.

## Shared system considerations for the dynamic CDT

You can use the dynamic CDT on systems running z/OS Version 1 Release 6, and later, that share the RACF database with systems that are downlevel or not using the dynamic CDT.

When RACF is *not* enabled for sysplex communications but the RACF data base is shared between two or more system, the following commands are not propagated to shared systems.

- SETROPTS RACLIST(CDT)
- SETROPTS RACLIST(CDT) REFRESH
- SETROPTS NORACLIST(CDT)

These commands do not take effect on shared systems until you issue them on each of those systems, or until the systems are IPLed.

## RRSF considerations for the dynamic CDT

---

When automatic direction is implemented across two or more systems, the class descriptor tables, including the dynamic classes, should be compatible on all participating systems.





---

## Chapter 11. Using PassTickets

If your installation includes workstations and client machines that are operating in a client/server environment, you might want to use RACF PassTickets to provide enhanced security across a network. A PassTicket provides an alternative to the RACF password and password phrase which allows workstations and client machines to communicate with a host without using a RACF password or password phrase.

Use of a PassTicket removes the need to send RACF passwords and password phrases across the network and allows you to move the user authentication part of signing on to a host from RACF to another product or function. End users of an application can use the PassTicket to authenticate their user IDs and log on to computer systems that contain RACF.

This chapter describes the PassTicket and how to set up the PassTicket environment. It includes information about:

- Activating the PTKTDATA class
- Defining profiles in the PTKTDATA class
- How RACF processes the PassTicket
- Enabling the use of PassTickets
- Auditing the use of PassTickets

For information about the programming that is needed for an application to generate a PassTicket, see *z/OS Security Server RACF System Programmer's Guide*.

---

### The RACF PassTicket

The RACF PassTicket is a *one-time-only*<sup>10</sup> password that is generated by a requesting product or function. It is an alternative to the RACF password and password phrase that removes the need to send RACF passwords and password phrases across the network in clear text. It makes it possible to move the authentication of a mainframe application user ID from RACF to another authorized function executing on the host system or to the workstation local area network (LAN) environment.

#### Legacy PassTickets and enhanced PassTickets

RACF PassTickets can be configured with two different algorithms:

- The legacy PassTicket algorithm
- The enhanced PassTicket algorithm

The legacy PassTicket algorithm is the original PassTicket implementation and the enhanced PassTicket algorithm is an updated version of the PassTicket algorithm. enhanced PassTickets function similarly as legacy PassTickets but contain a number of usability and security enhancements.

RACF supports generation and evaluation of PassTickets with either the legacy PassTicket algorithm or the enhanced PassTicket algorithm, per application based on PTKTDATA class profile configuration. Both legacy PassTickets and enhanced PassTickets can be generated by appropriately authorized applications to authenticate z/OS users to other z/OS applications. In either case, the generated PassTicket value is supplied to z/OS applications as an 8-character value in the password field. Both legacy PassTickets and enhanced PassTickets are generated and evaluated using a shared secret key. Both legacy PassTickets and enhanced PassTickets may be generated on z/OS or on other platforms and both PassTicket generation algorithms are documented in *z/OS Security Server RACF Macros and Interfaces*.

While the legacy PassTicket algorithm uses a secret 64-bit DES key, the enhanced PassTicket algorithm uses a 256-2048 bit HMAC secret key. While legacy PassTicket key material may be optionally masked in the RACF database, enhanced PassTickets keys must be stored encrypted in ICSF. For more information on PassTicket keys, see [“Protecting PassTicket keys” on page 286](#).

While the legacy PassTickets character set uses only uppercase characters A-Z and digits 0-9, enhanced PassTickets can optionally use an expanded character set which also includes the lowercase characters a-z and two special symbols. By supporting a much larger set of possible valid values, enhanced PassTickets have more variability and are therefore more secure against certain attack vectors than legacy PassTickets. The enhanced PassTicket character set can be configured in the PTKTDATA class profile with the TYPE(MIXED) or TYPE(UPPER) keywords in the SSIGNON segment.

While legacy PassTickets are valid 10 minutes before or after they are generated, enhanced PassTickets provides a configurable validity period which can be set between 1 second and 10 minutes. By configuring a shorter validity period, installations can limit the amount of time that enhanced PassTickets are valid. The enhanced PassTicket validity period can be configured in the PTKTDATA class profile with the TIMEOUT keyword in the SSIGNON segment.

For more information on configuring legacy and enhanced PassTickets, refer to the SSIGNON segment for the RDEFINE and RALTER commands in the *z/OS Security Server RACF Command Language Reference*.

**Note:** IBM strongly recommends using the enhanced PassTicket algorithm as it provides the same capabilities as the legacy PassTicket algorithm but also provides increased security.

## Activating the PTKTDATA class

Profiles that contain PassTicket information must be defined to the PTKTDATA class. This class must be active before RACF can generate or validate PassTickets.

Activate the PTKTDATA class only after at least one PassTicket key is defined. Otherwise, activating the PTKTDATA class when no PassTicket keys are defined can negatively impact authentication performance.

To activate the PTKTDATA class, enter the SETROPTS command as follows:

```
SETROPTS CLASSACT(PTKTDATA) RACLIST(PTKTDATA)
```

After you define or change profiles in the PTKTDATA class, refresh the class, as follows:

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

## Defining profiles in the PTKTDATA class

For each application that users can gain access to with the PassTicket, you must create at least one profile in the PTKTDATA class. The profile associates a PassTicket key with a particular application on a particular system. The profiles can be created so they apply to:

- All users
- Users who belong to a specific RACF group
- A specific RACF user, when connected to a specific RACF group
- A specific RACF user

To define the profile, use the RDEFINE command:

```
RDEFINE PTKTDATA profile-name SSIGNON(key-description) UACC(NONE)
```

where:

<sup>10</sup> Because it can only be used to authenticate to a specific application for a limited time interval, a RACF PassTicket is resistant to reuse. For most applications, once a particular PassTicket is used, the same user cannot use it again for the same application. For performance reasons, RACF uses main memory for this storage. If an application can run on more than one computer with individual memory at the same time, this level of reuse protection might not be available.

**PTKTDATA**

specifies the PassTicket key class.

**profile-name**

is the name of the profile (see [“Determining PTKTDATA profile names”](#) on page 283).

For the PTKTDATA class, the profile must be a discrete profile. Because each application must be uniquely defined, you cannot specify a generic profile in the PTKTDATA class. If you specify a generic profile, it is ignored during PassTicket processing for the application, and PassTickets cannot be used to authenticate users for that application.

**key-description**

defines the PassTicket keys and related configuration settings.

For legacy PassTickets:

- A subset of these keywords specify the method RACF is to use to protect the legacy PassTicket key in the RACF database on the host. You can specify either masking or encryption for the method (see [“Protecting PassTicket keys”](#) on page 286).
- legacy PassTicket keys are 64-bit Data Encryption Standard (DES) keys. With DES, eight of the 64 bits are reserved for use as parity bits, so those eight bits are not part of the 56-bit key. In hexadecimal notation, the DES parity bits are: X'0101 0101 0101 0101'. Any two 64-bit keys are equivalent DES keys if their only difference is in one or more of these parity bits.

For enhanced PassTickets:

- A subset of these keywords identify the enhanced PassTicket keys and related configuration settings to be used to generate and evaluate an enhanced PassTicket. enhanced PassTicket keys are 256-2048 bit HMAC keys.

## Determining PTKTDATA profile names

A PTKTDATA class profile name can consist of *one* of the following:

- An application name only
- An application name appended (or qualified) by a RACF connect group name
- An application name qualified by a RACF user ID
- An application name qualified by both a RACF connect group name and a RACF user ID

When the profile name consists of the application name and one or two qualifiers, the qualifiers are separated by a period. When a RACF connect group name and a RACF user ID are used as qualifiers, the group name must be appended to the application name and the user ID must be appended to the group name.

According to this rule, the name structures in the following list can be used as profile names in the PTKTDATA class. Any other name structures will be ignored. In this example, the application name is TSO1234, the user's current connect group name is SYS1, and the user ID is IBMUSER:

1. An application name concatenated with a RACF group name and user ID: TSO1234.SYS1.IBMUSER
2. An application name concatenated with a RACF user ID: TSO1234.IBMUSER
3. An application name concatenated with a RACF group name: TSO1234.SYS1
4. An application name: TSO1234

You might consider using a qualified profile when PassTicket generation is done on a different system whose security controls you trust less than z/OS and RACF, or if the other system is not under your control (for example, a system owned by a business partner). By scoping the use of a key, you limit the identities that can authenticate to the application using a PassTicket, should the key become compromised.

When PassTicket generation is done by the RACF PassTicket generation service, only profiles with name structure 4, the unqualified application name, are used. All other name structures are ignored.

When PassTicket evaluation occurs, multiple profiles can exist that fit the particular application, user, and group specification. When multiple profiles exist, RACF processing is as follows:

1. Assuming there is at least one qualified profile, RACF selects one qualified profile name according to the precedence shown in the previous list (items 1, 2, and 3).  
The first qualified profile found using this search precedence is selected and RACF evaluates the PassTicket using this key. Any other profiles with qualified names are ignored.
2. If no qualified name is found, or the evaluation using the key within the qualified profile is not successful because the key is not correct, RACF searches for a profile using only the application name. If such a profile exists, RACF evaluates the PassTicket using the key contained within this profile.

Depending on the application (APPC, CICS, IMS, MVS batch, TSO, or z/VM), the PassTicket function uses a specific method for determining profile names in the PTKTDATA class. If your application is other than those listed, see [“Other applications”](#) on page 286.

**Note:**

1. Check with your system programmer to see if your installation is using RACF exit ICHRIX01 to modify the application name that RACF uses during user verification processing. If so, the application name used to determine the PTKTDATA class profile name for APPC, CICS, IMS, MVS batch, TSO or VM applications *must* match the application name ICHRIX01 selects.  
For example, if the ICHRIX01 exit places the character string TS01234 in the application name position of the exit parameter list, the application name position of the PTKTDATA class profile must also be TS01234.
2. When a PassTicket is evaluated by RACROUTE REQUEST=VERIFY, an SMF Type 80 event code 1 record is always created. Relocate section 443 contains the application name that was used in the evaluation process.

## APPC, CICS, or IMS

To define a profile for an APPC, CICS, or IMS application, define the profile in the PTKTDATA class with a high-level qualifier that matches the standard naming conventions you use to define these applications to the APPL class.

- For general information on the APPL class, see [“Protecting applications”](#) on page 223.
- For APPC information, see [“RACF and APPC”](#) on page 248.
- For information about using IMS with RACF, visit [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims).
- For information about using CICS with RACF, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).

## MVS batch jobs

For MVS batch jobs that include RACF passwords in the job control language (JCL), you can replace the password with a PassTicket. To define the PTKTDATA profile for an MVS batch job, do the following:

1. Ask the system programmer for the SMF system identifier of the MVS system where the application runs. The SMF identifier is located in SMFPRMxx member of SYS1.PARMLIB and is specified by the SID value.
2. Determine the high-level qualifier of the PTKTDATA class profile for the MVS batch job by using the characters MVS as a prefix to the system's SMF identifier.

**Example of a profile name for an MVS batch job:** If the SMF identifier of the system where the application runs is SYS2, the resulting profile name is MVSSYS2. If the profile applies only to RACF users connected to the RACF group PROD, the resulting profile name is MVSSYS2.PROD.

**Note:**

1. If the SMF identifier contains blanks or characters that are not alphanumeric, omit them before specifying the profile name. For example, if the SMF identifier is SY\*6, specify the high-level qualifier of the PTKTDATA class profile as MVSSY6.
2. Certain batch jobs that are submitted from online TSO sessions can be defined in the PTKTDATA class profile using either the characters MVS or TSO as the prefix to the system's SMF identifier.

**Guideline:** Use MVS as the prefix to help administrators distinguish between a batch job and a non-batch job.

## TSO

Ask the system programmer if a VTAM generic resource name for TSO is being used, then reference the appropriate information below.

### ***Creating a TSO profile name (when a VTAM generic resource name for TSO is used)***

If VTAM generic resource naming is used for TSO application names, ask the system programmer for the generic name for TSO, and use it as the high-level qualifier of the PTKTDATA profile name.

The VTAM generic resource name for TSO is located in:

- The GNAME value in the TSOKEYxx member of SYS1.PARMLIB.
- The GNAME= operand of the START command issued to start TSO.

**Example:** The VTAM generic resource name for TSO on your system is TSOGR, and a PTKTDATA profile needs to be created for the TSO application. Use the VTAM generic resource name as the profile name. The resulting profile name is TSOGR. If the profile applies only to RACF users connected to the RACF group PROD, the resulting profile name is TSOGR . PROD.

#### **Note:**

1. A VTAM generic resource name allow the name by which an application is known to its end users to be different from the actual application name on a given execution system. This allows multiple real application servers to be used by large numbers of users who request the services of the application name by its generic name, while the requested service is actually provided by multiple backend application servers. With VTAM generic resources, the real backend application name does not need to be exposed to end users, since they refer to the application only by its generic name.
2. During TSO signon PassTicket evaluation, RACF checks the VTAM terminal address space control table (TCAST) for a VTAM generic resource name for each TSO application environment. If a VTAM generic resource name exists for this particular TSO application, it is used as the application name by RACF for the evaluation process. This results in consistency of application names between PassTicket generation time and evaluation time.

### ***Creating a TSO profile name (when a VTAM generic resource name for TSO is not used)***

If VTAM generic resource naming is not used for TSO, you should:

1. Ask the system programmer for the SMF system identifier of the MVS system where the application is running.

The SMF system identifier is located in the SMFPRMxx member of SYS1.PARMLIB and is specified by the SID value.

2. Determine the high-level qualifier of the profile name to represent the TSO application in the PTKTDATA class using the characters TSO as a prefix to the system's SMF identifier.

**Example:** The SMF identifier of the system the TSO application is running on is SYS2. To create the profile name, use TSO as the prefix. The resulting profile name is TSOSYS2. If the profile applies only to RACF users connected to the RACF group PROD, the resulting profile name is TSOSYS2 . PROD.

**Note:** If the SMF identifier contains blanks or characters that are not alphanumeric, they cannot be specified in the profile name. For example, if the SMF identifier is SY-5, you must specify the profile defined in the PTKTDATA class as TS0SY5.

## z/VM logon

If you want to use a PassTicket to access a z/VM system, you can define a profile for z/VM:

1. Ask the system programmer for the system ID of the z/VM system. It can be located by examining the CPU-ID (system ID) field in the RACF SMF CONTROL file.
2. Determine the high-level qualifier name of the profile to represent z/VM in the PTKTDATA class using the characters VM as a prefix to the CPU-ID field.

**Example of a z/VM profile name:** The system ID of the z/VM system is ISGR8. To create the profile name, use VM as the prefix. The resulting profile name is VMISGR8. If the profile applied only to RACF users connected to the RACF group PROD, the resulting profile name would be VMISGR8.PROD.

**Note:** If the CPU-ID field contains blanks or characters that are not alphanumeric, they cannot be specified in the profile name. For example, if the CPU-ID field contains VM7, you must specify the PTKTDATA class profile as VMVM7.

## z/OS UNIX applications

For the z/OS LDAP server, use the job name associated with the LDAP server's started procedure as the name of your PTKTDATA profile.

For other UNIX-based applications, such as the z/OS HTTP Server and the FTP server, that authenticate using the z/OS UNIX `__passwd()` service, the default application name is OMVSAPPL. Therefore, in most cases, use OMVSAPPL as the name of your PTKTDATA profile. However, because an application might change the default value or use the `__passwd_applid()` service to specify an explicit value, see your programmer for the correct application name to specify as the name of your PTKTDATA profile.

## Other applications

If your application is other than APPC, CICS, IMS, MVS batch, TSO or z/VM, define the application name that your application passes to RACF in the APPL parameter of the RACROUTE REQUEST=VERIFY as the name of your PTKTDATA profile. If your application does not pass an application name, follow the instructions for creating an MVS batch job profile name. See [“MVS batch jobs” on page 284](#).

## Protecting PassTicket keys

PassTicket keys are sensitive and must be protected from unauthorized disclosure. Entities with access to the configured application PassTicket keys can generate valid PassTickets for that application.

When you define legacy PassTicket keys, RACF either masks or encrypts each key. If the system has ICSF installed and available, you can store PassTicket keys in ICSF for added protection. When you define enhanced PassTicket keys they must be stored in ICSF. For more information, see [“Storing legacy PassTicket Keys Masked in RACF” on page 286](#) and [“Storing PassTicket keys encrypted in ICSF” on page 287](#).

## Storing legacy PassTicket Keys Masked in RACF

Legacy PassTicket keys can be stored encrypted in ICSF or masked in RACF with a proprietary masking algorithm when you define or alter it.

The masking algorithm is designed to provide protection against casual viewing of the PassTicket masked keys. The algorithm is **not** a cryptographic algorithm and cannot provide the level of security for the PassTicket keys that the use of cryptography can provide.

**Note:** IBM **STRONGLY** recommends that masked PassTicket keys are not used outside of a test environment.

To mask a legacy PassTicket key when you define or alter it, use the SSIGNON operand and KEYMASKED value with the RDEFINE or RALTER command.

You can use the ENCRYPTKEY keyword to encrypt a masked key and move it into the CKDS. See [“Converting legacy PassTicket masked keys to encrypted keys”](#) on page 288.

**Note:** To prevent unauthorized users from looking at the PassTicket keys that are stored in the RACF database, make sure the universal access authority (UACC) of the RACF database is NONE. This prevents unauthorized users from listing or copying the RACF data set that contains these sensitive keys.

## Storing PassTicket keys encrypted in ICSF

ICSF can be used to store PassTicket keys in the CKDS, encrypted under the master key. Using ICSF ensures the maximum possible security for the PassTicket keys.

For legacy PassTickets, there are two options for defining the key to ICSF:

- Use the SSIGNON operand and the KEYLABEL keyword to identify the CKDS key label to use for the particular PTKTDATA profile being added or altered. The key must refer to a DES key with a type of DATA and a length of 8 bytes. KEYLABEL is the recommended option as it allows for secure key entry and the use of your own naming convention for keys. You are responsible for adding the appropriate key to the CKDS, with the specified label, before it is used in a PassTicket operation.
- Use the SSIGNON operand and KEYENCRYPTED keyword to enter the key value to use for the particular PTKTDATA profile being added or altered. RACF will generate a key label value in the form *IRR.SSIGNON.sysname.mmddyyyy.hhmmss.nnnnnn* and add the key to the CKDS. The key label name is not user configurable.

For enhanced PassTickets, the key must be defined in ICSF:

- Use the SSIGNON operand and the EPTKEYLABEL keyword to identify the CKDS key label to use for the particular PTKTDATA profile being added or altered.
- The key label must refer to an ICSF HMAC key with a key algorithm of HMAC, a key type of MAC and the key usage fields must indicate GENERATE. The supported HMAC key size range is from 32 to 256 bytes. The recommended minimum key size is 64 bytes.
- You are responsible for adding the appropriate key to the CKDS with the specified label, before it is used in a PassTicket operation.
- The RACF enhanced PassTicket support uses ICSF HMAC keys which require that the ICSF CKDS is defined in either the variable length record format or common record format (KDSR). For more information on ICSF CKDS formats, please refer to [Introduction to z/OS ICSF in z/OS Cryptographic Services ICSF System Programmer's Guide](#).

The RLIST command displays the key label used for an encrypted key.

When the SSIGNON segment is displayed, the output will indicate which fields apply to legacy PassTickets and which apply to enhanced PassTickets.

When the RACF database is shared, the use of ICSF is simplest when the CKDS and RACF database are shared across a common set of systems. RACF always uses the local system's CKDS when generating or evaluating a PassTicket. If the PassTicket is generated on one system, and then evaluated on a different system, the evaluation will fail if RACF is unable to retrieve the key from the local CKDS. If the ICSF CKDS is not shared across systems which share the RACF database, ICSF services must be used to export the key label from the system on which the PassTicket key was defined. The key must then be imported to the ICSF CKDS of all other systems which share the RACF database. The ICSF CSNDSYX and CSNDSYI services can be used to export and import PassTicket keys from the ICSF CKDS. The ICSF CSNBKEX and CSNBKIM services can also be used to export and import PassTicket keys from the ICSF CKDS. There is a similar consideration if you are using the remote sharing facility to propagate commands that update PTKTDATA class profiles. If the target of the propagation is a multisystem node, the CKDS in use on the remote node's MAIN system will be the only CKDS updated with the new PassTicket key.

Note that older versions of RACF might have stored a legacy PassTicket key token in the profile instead of a key label. The creation of a legacy PassTicket key token is also possible if the user entering the RACF



command lacks authorization to the CSFKEYS profile protecting the key label name, or to the CSFKRC or CSFKRW service. Like a normal KEYENCRYPTED key, a key token is also encrypted under the CKDS master key, but it is stored in RACF instead of the CKDS, and thus there is no key label. RACF updates the key token when a master key change is detected. RACF only updates a key token when it is used in a PassTicket operation. If the master key is changed twice between use of a specific key token, the key token is rendered unusable. When the RACF database is shared, and the CKDS is not shared across the generating and evaluating systems, the CKDS master keys must be the same.

The RLIST command will indicate the presence of a legacy PassTicket key token. You can use the ENCRYPTKEY keyword of the RALTER command to move this token into the CKDS using a RACF-generated key label name. See [“Converting legacy PassTicket masked keys to encrypted keys”](#) on page 288.

**Important:** RACF does not delete keys from the CKDS. Before deleting or changing an encrypted key, take note of the current key label value so that it can be deleted from the CKDS, using ICSF interfaces.

## Converting legacy PassTicket masked keys to encrypted keys

A masked key can be converted to an encrypted key by specifying SSIGNON(ENCRYPTKEY) on an RALTER PTKTDATA command. The value of the key itself is not changed, and the user does not need to know the current value. This option is helpful when the key value is unknown, or cannot be immediately changed because it is being used to generate PassTickets on a different system. Keep in mind that the conversion cannot be reversed.

**Note:** If a key has been compromised, using ENCRYPTKEY provides no protection. The key value should be changed immediately.

ENCRYPTKEY can also be used to move a key token into the ICSF CKDS.

For example, to start encrypting the masked key associated with the MYAPPL application:

```
RALTER PTKTDATA MYAPPL SSIGNON(ENCRYPTKEY)
```

If the key is already encrypted, message IRR52254I is issued and the profile is unchanged.

To encrypt all of your masked keys at the same time, use the SEARCH command with the CLIST option:

```
SEARCH CLASS(PTKTDATA) CLIST('RALTER PTKTDATA ' ' SSIGNON(ENCRYPTKEY)')
```

This creates a CLIST in the data set named *issuing-userID.EXEC.RACF.CLIST*. After reviewing and editing this CLIST as appropriate, execute it. For example, from TSO:

```
EX 'JDAYKA.EXEC.RACF.CLIST'
```

## Authorization requirements for managing PassTicket keys

If SSIGNON(KEYENCRYPTED) is specified for legacy PassTickets on an RDEFINE PTKTDATA or RALTER PTKTDATA command, access to the following ICSF services needs to be defined:

- CSFCKI
- CSFKRC
- CSFKRW
- CSFKRD

If SSIGNON(KEYENCRYPTED) or SSIGNON(ENCRYPTKEY) is specified for legacy PassTickets on an RDEFINE PTKTDATA or RALTER PTKTDATA command, the user requires READ access to keys in the form of *IRR.SSIGNON.sysname.\** using profiles in the CSFKEYS class. *Sysname* is the name of the system where the keyword was specified. For information on protecting ICSF resources, see *z/OS Cryptographic Services ICSF Administrator's Guide*.

If RACF field level access checking is enabled, the user issuing the RDEFINE or RALTER command specifying the SSIGNON segment must have UPDATE access to the appropriate fields. For legacy PassTickets, UPDATE access is required to the PTKTDATA.SSIGNON.SSKEY resource in the FIELD class.



Note that the ENCRYPTKEY, KEYENCRYPTED, KEYMASKED, KEYLABEL and NOLEGACYKEY keywords all store data into the SSKEY field, and thus they are all protected by the same resource profile. For details on field level access control, see [“Field-level access checking”](#) on page 200.

## Examples of defining PTKTDATA class profiles

Suppose you want to define a profile for TSO in the PTKTDATA class. The system programmer has told you that a VTAM generic resource name for TSO is not being used, and that the SMF identifier of the system on which the TSO application is to run is R001.

### For legacy PassTickets:

You want to encrypt the legacy PassTicket key and specify a key value of X'E001193519561977'.

To define the profile for legacy PassTickets, enter:

```
RDEFINE PTKTDATA TSOR001 SSIGNON(KEYENCRYPTED(E001193519561977))
```

### For enhanced PassTickets:

You want to set the enhanced PassTicket ICSF key label to the value of 'TSOR001.EPTKEY01' and the character set type to MIXED.

To define the profile for enhanced PassTickets, enter:

```
RDEFINE PTKTDATA TSOR001 SSIGNON(EPTKEYLABEL(TSOR001.EPTKEY01) TYPE(MIXED))
```

## When the profile definitions are complete

After you define the PTKTDATA class profile for the application program that is to generate a PassTicket, the program can be installed and used.

RACF provides several services by which a z/OS application can request the generation of a PassTicket. For details on the R\_GenSec and R\_ticketerv services, see [R\\_GenSec \(IRRS00 or IRRSGS64\): Generic security API interface](#) and [R\\_ticketerv \(IRRSPK00\): Parse or extract in z/OS Security Server RACF Callable Services](#). For details on the RCVTPTGN service, see [Using the RCVTPTGN service to generate a PassTicket in z/OS Security Server RACF Macros and Interfaces](#).

## How RACF processes the PassTicket

To validate a PassTicket, RACF does the following:

1. Determines whether a profile has been defined for the application in the PTKTDATA class.
  - If a profile has not been defined and the value does not match the user's password, the user receives a message from the application indicating that the password is not valid.<sup>11</sup>
  - If the application is defined in the PTKTDATA class, processing continues.
2. Evaluates the value entered in the password field. The evaluation determines whether:
  - The value is a PassTicket consistent with this user ID, application, and time range.
  - For enhanced PassTickets, the PassTicket value also must have been generated with the same character set type (UPPER or MIXED) as the evaluator.

### Time Considerations:

- A PassTicket is considered to be within the valid time range when the time of generation, with respect to the clock on the generating computer, is within the acceptable validity period of the time of evaluation, with respect to the clock on the evaluating computer. For legacy PassTickets,

<sup>11</sup> RACF sends a message to the SYSLOG and to the security console. The application rejects the logon request the same way it rejects an incorrect password. The text of the message the user receives depends on the application.

the acceptable validity period is 10 minutes before or after the generation time. For enhanced PassTickets, the acceptable validity period is configurable between 1 second and 10 minutes before or after the generation time.

- Be sure that your MVS system and the evaluating computer use clock values that are within that time range. RACF uses the value stored for coordinated universal time (UTC), formerly called Greenwich mean time (GMT), in the algorithms that process PassTickets.
- One way to ensure that reasonably synchronized values are used is to set UTC in the GMT value of the MVS time of day (TOD) clock and to set a similar value in each of the other systems with which RACF shares PassTicket information. You can still use the MVS local time for local timestamp information, and resetting the local time does not affect the GMT value kept in the TOD clock.

**Important:** Before setting the TOD clock's GMT value to UTC, make sure that the subsystems and applications you use are not affected.

- To be sure the MVS system clock is set properly, the system console operator should issue:

```
DISPLAY T
```

- The system displays the time with information similar to the following:

```
IEE136I LOCAL: TIME=14.06.18 DATE=1997.309
          GMT: TIME=19.06.18 DATE=1997.309
```

**Important:** If the MVS DISPLAY T command indicates that your system clock is not set correctly for GMT, you need to analyze the consequences of resetting the clock. It is possible that other programs that execute on the system have been adjusted to tolerate an incorrect GMT setting. You might need to readjust those programs before resetting the system clock.

- See *z/OS MVS Initialization and Tuning Reference* and *z/OS MVS System Commands* for more information on setting clocks. See *z/OS Security Server RACF Macros and Interfaces* for more information on the algorithms.

3. Determines if the PassTicket has been used previously on this computer system for this user ID, application and time range.

- If the value was used before, and if PassTicket replay protection has not been bypassed, the user receives a message from the application<sup>12</sup> indicating that the password is not valid.
- If the value was not used before or PassTicket replay protection has been bypassed, the PassTicket is considered valid and processing continues.

4. Allows or denies access to the target application.

- If the PassTicket is valid, RACF gives the user access to the desired application.
- If the value is not valid, the host application sends a message to the user indicating that the password is not valid (assuming the value also did not evaluate correctly as the user's RACF password).

**Note:** If the PassTicket key is stored in ICSF with the KEYENCRYPTED, ENCRYPTKEY, KEYLABEL or EPTKEYLABEL keywords, ICSF must be active when RACF tries to authenticate the PassTicket. If it is not active, RACF cannot validate the PassTicket.

## Bypassing PassTicket replay protection

You might use the option to bypass PassTicket replay protection when the threat of PassTicket replay is not a security concern, such as in the following cases:

<sup>12</sup> RACF sends a message to the SYSLOG and to the security console. The application rejects the logon request the same way it rejects an incorrect password. The text of the message the user receives depends on the application.

- Applications which save the password and use it for multiple logons on a user's behalf. (Note that the multiple logons must occur within the PassTicket validity window for this type of application to work with PassTickets.)
- Trusted registry domains that exchange PassTickets as a method of establishing trust.
- Applications that request PassTickets for a particular USERID/APPLID combination more than once during a one-second time interval.

The option to bypass PassTicket replay protection allows the PassTicket replay protection to be bypassed for selected applications or combinations of selected applications, users, or groups.

**Note:** The option to bypass PassTicket replay protection should only be used in secure environments where access to generated PassTickets is limited within a secure or internal network.

## Bypassing legacy PassTicket replay protection

You indicate that replay protection is to be bypassed for legacy PassTickets for a particular application by adding the text string NO REPLAY PROTECTION to the APPLDATA field of the PTKTDATA profile for that application. You must separate each word in the string with a single blank space, alphanumeric character, or keyboard symbol. The NO REPLAY PROTECTION text string will always be translated to upper case by the RALTER or RDEFINE commands.

The NO REPLAY PROTECTION text string can appear anywhere within the APPLDATA field, allowing for the existence of other information already in the field, or for new information that might be added in the future.

The following are examples of commands that will cause legacy PassTicket replay protection to be bypassed.

### Examples:

```
RALTER PTKTDATA profile-name APPLDATA('NO REPLAY PROTECTION')
RDEFINE PTKTDATA profile-name APPLDATA('NO REPLAY PROTECTION')
RDEFINE PTKTDATA profile-name
    APPLDATA('FOR THIS APPLICATION NO REPLAY PROTECTION IS IN EFFECT')
```

### Note:

1. Other than the APPLDATA (application data) field of the application profile containing the text string, NO REPLAY PROTECTION, there is no other external indication that replay protection is bypassed.
2. The APPLDATA field replay protection only applies to legacy PassTickets and does not affect the replay behavior of enhanced PassTickets.

## Bypassing enhanced PassTicket replay protection

You indicate that replay protection is to be bypassed for enhanced PassTickets for a particular application by setting the SSIGNON segment keyword REPLAY(YES) in the PTKTDATA profile for that application.

### Example:

```
RALTER PTKTDATA profile-name SSIGNON(REPLAY(YES))
```

**Note:** The SSIGNON segment REPLAY keyword replay protection only applies to enhanced PassTickets and does not affect the replay behavior of legacy PassTickets.

## Enabling the use of PassTickets

To enable RACF to validate PassTickets, the RACF administrator must have:

- Activated and RACLISTed the PTKTDATA class.
- Defined a PassTicket key for each application in a profile in the PTKTDATA class.
- Issued the SETROPTS RACLIST(PTKTDATA) command.

- Ensured that encrypted PassTicket keys have been distributed to the CKDS of all systems which share the RACF database (as necessary).

**Note:** If you make any additions or changes to profiles in the PTKTDATA class after you issue the SETROPTS RACLIST(PTKTDATA) command, be sure to reissue it using the REFRESH option in order to activate your changes.

As a result, the RACF database and ICSF contain all the information necessary to validate PassTickets for each application that has a PTKTDATA class profile defined.

## Verifying the PassTicket environment

After activating the PassTicket environment for each application, you should verify the environment.

- Generate a PassTicket for a user of the application using RCVTPTGN, R\_ticketserv, R\_GenSec, or the Java™ interface. For information about these services see *z/OS Security Server RACF Macros and Interfaces*.
- Access the application using the userid and PassTicket.

## Migrating from legacy PassTickets to enhanced PassTickets

Enhanced PassTickets provide the same capabilities as legacy PassTickets but with improved security. Migration from legacy PassTickets to enhanced PassTickets will take planning and effort. RACF allows for an installation to have both legacy PassTickets and enhanced PassTickets configured for the same application in the same PTKTDATA class profile.

When a PTKTDATA class profile contains both a legacy PassTicket key and enhanced PassTicket key:

- PassTicket generation requests through RACF services will result in an enhanced PassTicket.
- PassTicket evaluation requests through RACF will evaluate the PassTicket with both the legacy PassTicket algorithm and enhanced PassTicket algorithm.

Installations that wish to migrate from legacy PassTickets to enhanced PassTickets can use the following steps as a guide:

### 1. Determine the desired enhanced PassTicket character type:

This setting determines the possible characters that represent the enhanced PassTicket. TYPE(UPPER) will only use uppercase A-Z and digits 0-9. TYPE(MIXED) also includes lowercase a-z and the symbols underscore ( \_ ) and dash ( - ).

In general, installations that have RACF mixed case password support enabled should use TYPE(MIXED) and installations that do not have RACF mixed case password support enabled should use TYPE(UPPER). The default value is TYPE(MIXED). The TYPE setting of the enhanced PassTicket evaluator must match the TYPE setting of the PassTicket generator.

### 2. Determine the desired enhanced PassTicket REPLAY allowed setting:

In some cases an installation may require PassTickets to be able to be replayed for a particular application. To allow replay of enhanced PassTickets for the application set REPLAY(YES) in the PTKTDATA class application profile. The default value is REPLAY(NO). See [“Bypassing PassTicket replay protection”](#) on page 290 for more information on PassTicket replay considerations.

### 3. Define the enhanced PassTicket HMAC key in the ICSF CKDS:

The key must be defined with a key label that refers to an ICSF HMAC key with a key algorithm of HMAC, a key type of MAC and the key usage fields must indicate GENERATE. The supported HMAC key size range is from 32 to 256 bytes. The recommended minimum key size is 64 bytes.

Refer to *z/OS Cryptographic Services ICSF Application Programmer's Guide* for details on managing ICSF keys.

### 4. Add the enhanced PassTicket Key Label to the PTKTDATA class profile:

Update the PTKTDATA class application profile to add the key label of the HMAC key in ICSF using the SSIGNON segment EPTKEYLABEL keyword.

When the enhanced PassTicket key label is added to the PTKTDATA class application profile RACF will begin to evaluate user specified passwords with the enhanced PassTicket algorithm.

Systems which do not share the same RACF database and/or ICSF datasets will need to provision the same enhanced PassTicket HMAC secret key in order to generate and evaluate compatible enhanced PassTickets.

#### 5. Update applications to generate enhanced PassTickets:

Applications that generate PassTickets on-platform using RACF services will begin to generate enhanced PassTickets when an enhanced PassTicket key is added to the PTKTDATA class application profile.

Applications that generate PassTickets outside of RACF with the PassTicket algorithm need to be updated to support the enhanced PassTicket algorithm. Once the application supports generation of enhanced PassTickets, it must be configured with the same HMAC secret key and character set as RACF for evaluation to be successful. Refer to the *z/OS Security Server RACF Macros and Interfaces* for details on implementing the enhanced PassTicket algorithm in your own application.

#### 6. Test enhanced PassTicket generation and evaluation:

Use the application to generate an enhanced PassTicket and attempt to use it to authenticate to the configured target application.

#### 7. Remove the legacy PassTicket key:

Once it has been confirmed that enhanced PassTicket evaluation is successful and all applications are no longer generating legacy PassTickets for the target application, the legacy PassTicket key should be removed from the PTKTDATA class profile with the NOLEGACYKEY keyword.

## Preventing errors

The following checklist describes the errors that might cause a PassTicket to fail. To prevent these errors from occurring:

1. Read the list before you use the PassTicket.
2. Review your process to ensure that you have entered all of the information correctly.
3. Verify the information by using the procedures described in [“Verifying the PassTicket environment” on page 292](#).

Use this checklist to prevent or correct errors:

- The PTKTDATA class is activated.
- You issued the SETROPTS RACLIST(PTKTDATA) command.
- You issued the SETROPTS RACLIST(PTKTDATA) REFRESH command after defining the profile.
- A PTKTDATA class profile exists for the application.
- The application name used by RACROUTE REQUEST=VERIFY during evaluation matches the name in the PTKTDATA profile that you expect to be used. The SMF Type 80 event code 1 record includes relocate section 443, which contains the application name that was used in the evaluation process. If a z/OS application is using the R\_Gensec, R\_Ticketserv or RCVTPTGN service to generate or evaluate a PassTicket and these requests are being logged (see [“Auditing the use of PassTickets” on page 294](#)), SMF Type 80 event code 81 (Evaluate) and event code 82 (Generate) will contain the application name in relocate section 67.
- You issued the RDEFINE command correctly.
- A protected user ID may not be used for PassTicket authentication.
- The PassTicket key must be the same on the system which generated the PassTicket and the system on which the PassTicket is being evaluated.
- The application name used to generate the PassTicket must match the application name used to log on with the PassTicket. Ensure the application name is not altered by a user exit during logon.

- PassTickets can be generated with the legacy PassTicket algorithm or enhanced PassTicket algorithm. Enhanced PassTickets can use either a character set type of UPPER or MIXED. The PassTicket must be evaluated with the same algorithm and character set type as it was generated.

Even if you have followed the proper procedures, it is still possible to receive a message stating that a password is incorrect and be denied access to the application. This can occur if:

- PassTicket replay protection is not being bypassed, and the PassTicket was used previously for this user, application, and time range.

In this case, RACF generates an SMF record that logs an attempt to replay a PassTicket.

- The GMT clock on the evaluating computer is outside the valid time range for the PassTicket.

This can be caused by one of the following:

- The GMT clock on the generating computer and the clock on the evaluating computer are not reasonably synchronized.
- The PassTicket was not used within approximately 10 minutes of being generated.
- The system clock on the evaluating computer might not be set correctly in relation to GMT. See the information about time considerations in [“How RACF processes the PassTicket” on page 289](#).
- An encrypted key is being used, but the key is not preset in the local ICSF CKDS.

PassTicket diagnostic reason codes are provided when generation or evaluation of a PassTicket fails in the following locations:

- SMF Type 80 records:
  - The event codes and relocate sections documented above (as containing the application name used) also contain failure return and reason codes.
- Service return and reason codes:
  - The services used to generate and evaluate PassTickets (also listed above) can provide useful diagnostic information. Note that for R\_Gensec and R\_Ticketserv, the application must have requested the additional diagnostics by using the 'extended' versions of the functions. Check if the application provides a trace log or other diagnostic medium containing the return and reason codes from these services.

## Auditing the use of PassTickets

---

Generation and evaluation of PassTickets can be audited. The SETR LOGOPTIONS settings of the PTKTDATA class, as well as the AUDIT and GLOBALAUDIT setting of the PTKTDATA profiles which contain PassTicket keys can be used to determine how the use of PassTickets is audited.

SMF type 80, event code 82 (PassTicket generate) records are created in the following circumstances:

- The RCVTPTGN service is used to generate a PassTicket.
- The R\_ticketserv or R\_GenSec service is used to generate a PassTicket.
- The Java service, described in *z/OS Security Server RACF Macros and Interfaces* is used to generate a PassTicket.

SMF type 80, event code 81 (PassTicket evaluate) records are created in the following circumstances:

- The R\_ticketserv or R\_GenSec service is used to evaluate a PassTicket.
- The Java service, described in *z/OS Security Server RACF Macros and Interfaces* is used to evaluate a PassTicket.

When a user provides a PassTicket to a standard z/OS authentication service, logging is performed for the PassTicket evaluation in the SMF type 80 event code 1 record created to record the logon event. This SMF record indicates when a user authenticates with PassTicket and whether the PassTicket was evaluated with the legacy PassTicket algorithm or enhanced PassTicket algorithm.

For more details on PassTicket audit records, please refer to *z/OS Security Server RACF Macros and Interfaces*.

## Allowing password changes when authenticating with a PassTicket

By default, RACROUTE REQUEST=VERIFY and REQUEST=VERIFYX do not allow a new password or password phrase to be specified when the value specified as the current password is a PassTicket. This function could be useful, for example, by a password reset application when the current password is unknown.

You can restrict this capability to only those applications that are known to need this capability. For example, the kpasswd command provided by Network Authentication Service (Kerberos) uses this capability to change a user's password.

When a user is authenticated using a PassTicket and a new password or phrase is specified, RACROUTE REQUEST=VERIFY/X checks a resource in the PTKTDATA class named IRRPTAUTH.PWCHANGE.APPL.appl-name for UPDATE access, on behalf of the user who is being verified. The value of appl-name is the exact same value used during PassTicket evaluation.

If no profile covers the resource, the password change is not allowed.

For example, suppose your Kerberos users belong to a group named KRBUSERS and the application name used by Kerberos is SKRBKDC. To allow Kerberos users to change their passwords using the kpasswd command, issue the following commands:

```
RDEFINE PTKTDATA IRRPTAUTH.PWCHANGE.APPL.SKRBKDC UACC(NONE)
```

```
PERMIT IRRPTAUTH.PWCHANGE.APPL.SKRBKDC CLASS(PTKTDATA)
      ID(KRBUSERS) ACCESS(UPDATE)
```

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

When a failure occurs, VERIFY/X creates an SMF 80 Event Code 1 record with either an Event Code Qualifier value of x'1A' (invalid new password) or x'25' (new password phrase is not valid). Depending on the logging options in effect for the PTKTDATA class or the PTKTDATA profile, an Event Code 2 (ACCESS) record may also be created and an ICH408I message issued to the security console.

You can create a generic profile that denies access and set it in warning mode to discover what applications are using this function, without impacting their ability to do so:

```
RDEFINE PTKTDATA IRRPTAUTH.PWCHANGE.APPL.*
      UACC(NONE) WARNING
```

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

Alternatively, you can create a generic profile that grants UPDATE access and logs successes over time. Be sure you are auditing the USER class so that password changes can be identified in the SMF record:

```
RDEFINE PTKTDATA IRRPTAUTH.PWCHANGE.APPL.*
      UACC(UPDATE) AUDIT(ALL(READ))
```

```
SETROPTS RACLIST(PTKTDATA) REFRESH
```

```
SETROPTS AUDIT(USER)
```

You can also review your existing SMF data to see if this function has been used in the past. All PassTicket evaluations are logged by RACROUTE REQUEST=VERIFY. When SETROPTS AUDIT(USER) is in effect, password and phrase changes are identified by SETROPTS AUDIT(USER) appearing as a reason for logging. Relocate Section 443 indicates when a PassTicket was successfully evaluated.





---

## Chapter 12. Detecting ACEE modifications

The ACEE (accessor environment element) is a control block that contains a description of a user's security environment, including user ID, current connect group, user attributes, and group authorities. An ACEE is constructed during user identification and verification, using information in the user's RACF profile, and is anchored in the user's address space. It is referenced by RACF during command and resource authorization processing to determine if the user is allowed to perform a given action or access a given resource.

An application running in an authorized state can make modifications directly to the ACEE in storage after its creation. Such a modification would be outside the involvement of the security product, and the application must take caution to ensure that such updates do not interfere with normal system processing. Some applications might make an ACEE change to temporarily or permanently elevate the privilege of the user. This use is not necessarily malicious, and may be beneficial. For example, consider a reporting application that must access a large number of protected resources on a user's behalf. It could be infeasible to explicitly grant permission to all of the resources, and it might result in excess auditing that is unnecessary.

A security administrator might want to be aware when an application modifies an ACEE, to make sure that any increase in the user's authority complies with the organization's security policy. This can be done using the ACEECHK class.

When the ACEECHK class is active and RACLSTed, RACF command and authorization processing will detect certain changes made to the ACEE that affect authorization. When a modification is detected, RACF issues message IRR421I to the security console. IRR421I contains information to possibly help you determine whether the modification is expected. See *z/OS Security Server RACF Messages and Codes* for more information about message IRR421I.

If the modification is expected, a program can be added to an exception list in the ACEECHK class such that future IRR421I messages are suppressed. When RACF detects an ACEE modification, it will look in the ACEECHK class for the existence of a 'bypass' profile, indicating that the IRR421I message should be suppressed. Only the existence of the profile matters; the access list, universal access, and so forth are ignored.

To understand the bypass profiles, you must understand the ACEEs and the user IDs that are possibly involved. Every address space has an ACEE that represents the identity of the user or application associated with the address space. This ACEE is pointed to by the address space control block extension (ASXB), which can be located using well-defined system pointers. An address space might be for a single-user. For example, a TSO logon or a batch job.

An authorized application, such as a server, can work on its own behalf, or can run work under the authority of its clients. It can do the latter in one of three ways:

1. By attaching a task (or "thread") and associating an ACEE with it. Such an ACEE can likewise be located using well-defined system pointers to access the task control block (TCB).
2. By creating an ACEE for the end user and explicitly providing it to RACROUTE REQUEST=AUTH. This ACEE may be created for the life of an authorization request, or for a longer interval. The application knows the location of such ACEEs, but RACF cannot locate them using system pointers. Such an ACEE is called a 2nd party ACEE.
3. By specifying a user ID on RACROUTE REQUEST=AUTH. RACF creates the ACEE and anchors it in the caller's ACEE. Such an ACEE is called a 3rd party ACEE.

RACF command and authorization checking will check the task level ACEE if one exists and the address space level ACEE if not. In addition, authorization checking will check a 2nd and 3rd party ACEE. In this situation, it is possible for two IRR421 messages to be issued.

IRR421I displays a "call chain". This describes the flow of control of programs leading up to the currently running program, which is the program that triggered the IRR421I. This information might help in identifying the application responsible for modifying the ACEE, and so it is important to understand the

possible program configurations that might exist in the environment. The job step task is typically the "top" program in an address space. It might call other programs, and these programs might call other programs and so forth. In a simple case where a batch job runs a single program that modifies its ACEE and then accesses a resource for which a system authorization facility (SAF) check is made, then that program is the one for which you would consider adding a bypass profile.

However, if this program calls another program, and it is the second program driving a SAF check, this second program is the one that IRR421I identifies as the currently running program. If it is a utility program from some system library, you would not want to create an exception for it because the exception would apply to the utility's use by all programs that call it. Now consider the case of a program or command invoked by a TSO user. As in the previous scenario, it calls a utility program that drives a SAF check. Again, you would not want to add an exception for the utility program. However, in this case, you would also not want to add an exception for the job step program since that would belong to TSO. Adding such an exception would suppress IRR421I messages for all TSO users.

**Note:** It is entirely possible that the program that modified the ACEE is not in the call chain. For example, imagine if someone implemented an SVC in an authorized library to turn on a bit in the ACEE and also wrote a TSO command processor to call it. If someone issued the TSO command, then ran a different program which triggered an IRR421I message, the TSO command program name would not be identified in the call chain. As another example, the program might actually have been in the call chain, but was not the first program of a parent task with respect to the currently running program (only the first program of ancestor tasks is displayed in the call chain in IRR421I).

RACF exception checking is performed for every program in the displayed call chain, starting at the currently running program and ending with the job step program, until a matching exception is located. Therefore, it is important to understand the application environment before adding an exception.

A bypass profile name is of the format:

```
IRR.EXCLUDE.program-name[.user1[.user2]]
```

Where:

- "IRR.EXCLUDE." is a constant prefix
- *program-name* is the name of a program in the chain of execution
- *user1* is the user ID from the address space ACEE
- If *user1* is a server running work on behalf of a different user, then *user2* is the user ID from the end user ACEE that is being checked (the task-level, 2nd or 3rd party ACEE). Note that 2nd and 3rd party ACEES apply only to RACROUTE REQUEST=AUTH processing (ACEEs cannot be passed to RACF commands. The command issuer's ACEE will always be located environmentally; from either the TCB or ASXB).

RACF will look for a matching profile in the following order.

If the ACEE being checked is the address space ACEE:

1. IRR.EXCLUDE.*program-name.user1*
2. IRR.EXCLUDE.*program-name*

If the ACEE being checked is not the address space ACEE:

1. IRR.EXCLUDE.*program-name.user1.user2*
2. IRR.EXCLUDE.*program-name.user1*
3. IRR.EXCLUDE.*program-name.user2*
4. IRR.EXCLUDE.*program-name*

**Note:** For authorization checking, if both a task level and 2nd or 3rd party ACEE are found to have been modified, the list is checked for each user ID separately.

**Examples:**

1. The installation knows that program ABCXYZ modifies ACEEs in a safe manner and wants to suppress IRR421I messages when any user is running the program. Defining a profile named IRR.EXCLUDE.ABCXYZ accomplishes this.
2. A started task is designed to safely alter its own ACEE. It runs a program named STCPROG1. The administrator has set up the started task to run under user ID STCUSR1. The message is to be suppressed when STCUSR1 is running STCPROG1, but not when any other user is running STCPROG1. Defining a profile named IRR.EXCLUDE.STCPROG1.STCUSR1 accomplishes this.

**Notes:**

- It is recommended to audit changes to the ACEECHK class using SETROPTS AUDIT(ACEECHK)
- ACEECHK supports generic profile names. However, profiles named "\*" and "\*\*\*" are ignored since they would cause every program to be exempted from privilege escalation detection.
- When RACF program control is active, the exception list will not be checked if the environment is uncontrolled ("dirty"). IBM recommends activating program control when activating the ACEECHK class so that any program added to the exception list is known to come from a library defined to program control. See [Protecting programs](#) for more information.
- If the modification is detected in the user ID field of either of the possible ACEEs involved, the profile qualifier corresponding to the modified ACEE is not considered when looking for a match. For example, if the user ID was modified in a 2nd-party ACEE, RACF will not check for IRR.EXCLUDE.*program-name.user1.user2* or IRR.EXCLUDE.*program-name.user2*.
- A unix program cannot be added to the exception list. If you wish to suppress IRR421I messages for a unix file, it will first need to be moved into an MVS library.

**Failure option**

You can request that RACF abend the running program when a modified ACEE is detected. This will result in a 4C6 abend with reason code X'ACE'. The abend is issued only if an exception was not located for a program in the call chain.

To request this failure option, define the IRR.ABEND.ON.FAILURE resource profile in the ACEECHK class and REFRESH the class. As with the exception profiles described above:

- only the existence of the profile matters.
- "backstop" profiles (\* or \*\*) are ignored.

Careful thought and planning should precede the decision to enable the failure option. You should only do so after running in production under normal workloads for a long enough interval that you are confident that you have identified all programs that are known to modify ACEEs. The results of an abend on an application can be unpredictable and largely depend on the recovery mechanisms in place for the programs running in the environment.

The RACF SETROPTS command will always fail when a modified ACEE is detected. The RACF RDEFINE command will only fail when it is asked to define a profile in the ACEECHK class and a modified ACEE is detected. All other RACF commands, including RDEFINE commands that do not involve the ACEECHK class, will only issue warning messages when an ACEE modification is detected by the ACEECHK class.



## Chapter 13. Protecting programs

This topic provides in-depth information on controlling programs, program libraries, and program access to data.

### Related information:

- For more information about using RACF to control access to program dumps, see [“Controlling access to program dumps”](#) on page 230.
- For information about enabling users to digitally sign program modules and enabling signature verification of program modules, see [Chapter 14, “Program signing and verification,”](#) on page 327.

## Overview of protecting programs

Program control provides the following functions:

- Simple controls to restrict the ability to execute specified programs by granting users either READ or NONE access through the PROGRAM class, and (when necessary) READ access to the DATASET profile that protects the load library that contains the program.
- More complex controls that can prevent users from copying sensitive programs or viewing the contents of such programs by granting the users either EXECUTE or NONE access through the PROGRAM class, or (in some cases) EXECUTE to the DATASET profile that protects the library that contains the program. Programs controlled in this way are referred to as *execute-controlled* programs.
- Improved resistance to attacks by malicious users or programs implementing malicious functions (such as Trojan horses) in a z/OS UNIX environment when you define the BPX.DAEMON profile in the FACILITY class and require that the program execution environments for UNIX daemons and servers remain *clean*.
- Program access to data sets (PADS) to allow users to have more access to data sets than they would otherwise have while running specified programs that provide restricted access to the data.
- Program access to SERVAUTH resources to allow access to IP addresses only when executing certain programs.

By defining programs in the PROGRAM class you indicate that you place some amount of trust in their behavior. Although the level of trust can vary, these programs are trusted more than programs created by general users of the system. An environment in which someone has run a program not defined in the PROGRAM class is considered a *dirty*, *unsafe*, or *uncontrolled* environment.

RACF requires a clean environment in functions 2 through 5 above because allowing use of those functions in an uncontrolled environment would make it relatively simple for malicious users with some specific knowledge to bypass the program-related security controls and gain inappropriate access to the data.

### Terms to know:

1. When used in this discussion, an *environment* is one of the following:
  - TSO session
  - TSO command invoked by **TSOEXEC** or the IKJEFTSR service
  - Job step in a batch job, started procedure, or started job
  - UNIX address space
2. A *clean* environment is one in which only programs defined in the PROGRAM class have run.
3. A *program* refers to a load module residing in a partitioned data set (PDS) or a program object residing in a program library (PDS/E).

### Restrictions:

1. Programs that reside in the UNIX file system are excluded from this discussion. Execution of programs in the UNIX file system is controlled using UNIX security controls (as opposed to RACF PROGRAM profiles), and programs resident in the UNIX file system cannot be used for PADS or program access to SERVAUTH resources.
2. RACF and z/OS cannot protect programs written in the TSO/E CLIST language, PERL, Java, or other interpreted languages.
3. RACF and z/OS can protect programs written in REXX only if they are compiled and link-edited as load modules or program objects.

In making use of program control, you must decide:

- Whether to operate in BASIC (default) or ENHANCED (more secure) program security mode.
- Which programs to define in the PROGRAM class and how to define them (which to some extent depends on the program security mode chosen).
- How to protect the libraries that contain the programs.

See the [“Migrating from BASIC to ENHANCED program security mode” on page 308](#) for migration and other planning considerations.

## Program security modes

---

RACF can operate in:

- BASIC program security mode (default)
- ENHANCED program security mode
- ENHANCED-WARNING program security mode

Compared with BASIC mode, ENHANCED mode offers extra protection from hackers and other malicious users, but requires more work in setting up the PROGRAM profiles that control program protection. It also further restricts the environment in which users can make use of program access to data sets (PADS), program access to SERVAUTH resources, and execute-controlled programs. It optionally provides additional restrictions on the execution of UNIX servers and daemons.

ENHANCED-WARNING mode provides a migration path from BASIC mode to ENHANCED mode. It operates like ENHANCED mode, but when a request occurs that would fail because it does not meet the restrictions for ENHANCED mode RACF checks to see if it would have granted the request if running in BASIC mode. If so, RACF allows the request but issues warning messages and creates SMF records to warn you of the problem. This allows you to fix the problem before completing the migration. See [“Migrating from BASIC to ENHANCED program security mode” on page 308](#) for a procedure to migrate from BASIC to ENHANCED program security mode.

You can specify the mode through the IRR.PGMSECURITY profile in the FACILITY class. Define the profile and specify the APPLDATA operand as:

- 'BASIC' for RACF to operate in BASIC program security mode
- 'ENHANCED' for RACF to operate in ENHANCED program security mode
- Empty, or any value other than 'BASIC' or 'ENHANCED', for RACF to operate in ENHANCED-WARNING program security mode.

If you do not define this profile, RACF operates in BASIC program security mode.

**Guideline:** If you make use of the program control functions, use ENHANCED program security mode for the extra protection that it provides.

After choosing a program control mode and defining IRR.PGMSECURITY to specify that mode, PROGRAM profiles should be defined. Then, program control functions can be enabled by issuing SETROPTS WHEN(PROGRAM). If you make changes to the PROGRAM profiles, you can make those changes effective by issuing SETROPTS WHEN(PROGRAM) REFRESH. RACF activates your chosen program security mode based on the presence of the IRR.PGMSECURITY profile and the contents of the APPLDATA field in the profile. This is done when you issue either of these commands or during subsequent system

initialization (IPL). RACF does not inspect the APPLDATA field for the IRR.PGMSECURITY except during this processing, and does not issue an error message if the profile has an unexpected APPLDATA value. Instead, it runs in ENHANCED-WARNING program security mode.

**Note:** Any job steps, started procedure steps, or TSO sessions that start after you switch to ENHANCED or ENHANCED-WARNING program security mode run in that mode. Any that started before you switched, continue to run in BASIC program security mode until they finish.

You can display the program security mode for the system at any time by issuing SETROPTS LIST. The first line of output indicates whether you have activated WHEN(PROGRAM) processing and displays the program security mode.

## Simple program protection in BASIC or ENHANCED mode

If you have a need to prevent a user or group from executing a particular program, you can define a profile for that program in the PROGRAM class and issue the PERMIT command to specify an access level of NONE for the program. This simple usage of program control does not depend on making any other PROGRAM definitions or on keeping the environment clean. You can also use this simple form of program control to audit usage of a program. If you only need to audit usage of the program and do not need to control which users can execute it, grant all users READ access and set the AUDIT operand appropriately.

A program protected by a PROGRAM profile is sometimes called a *controlled* program. To protect a program with a PROGRAM profile, use the RDEFINE command and specify the name of the program as the *profile-name*. You must also specify the ADDMEM operand and indicate:

- Name of the library that contains the program (required)
- Volume serial of the DASD volume that contains the library (optional)
- PADCHK or NOPADCHK option (optional; see [“Choosing between the PADCHK and NOPADCHK operands” on page 314](#) for more information.)

The PROGRAM profile can also contain other information, such as the following:

- UACC
- Standard access list
- Conditional access list

**Specific and nonspecific profile names:** The name of the PROGRAM profile can be completely specified, in which case the profile protects only one program name. This type of PROGRAM profile is known as a *specific* PROGRAM profile because it protects one specific program name. The name of the PROGRAM profile can also end with an asterisk (\*), in which case the profile can protect more than one program name. This type of PROGRAM profile is known as a *nonspecific* PROGRAM profile.

**Example:** A PROGRAM profile named ABC\* protects programs whose names begin with ABC (including a program named ABC) and reside in the library specified using the ADDMEM operand unless another profile name matches more characters of the program name.

If you have two PROGRAM profiles named ABC\* and ABC, and both profiles specify the name of the library where the ABC program resides, RACF uses the ABC\* profile for authorization checking of program ABC, *not* the ABC profile.

- If you want to control the ABC program with a *specific* profile named ABC, you can use one of the following methods:
  - Move the ABC program to a separate library and alter the ABC profile using the ADDMEM operand of the RALTER command to specify the new library and the DELMEM operand to remove the old library. (You will also need to change the way you invoke the ABC program to ensure that the new copy is used.)
  - Delete the ABC\* profile and define a set of new profiles named ABCx\* profiles where x is the next character that matches your program names that begin with the characters ABC. For example, if you have programs named ABCJA, ABCJB, ABCLA, and ABCLB, define profiles named ABCJ\* and ABCL\* to protect them.

- If you want to control the ABC program with a *nonspecific* profile, delete the profile named ABC\* and define a profile named AB\*. (Before doing this, examine all program names beginning with the characters AB to ensure that this new profile does not authorize unintended access to any additional programs.)

When defining the PROGRAM profile, supply the ADDMEM operand in the following format:

```
ADDMEM('library_name'/optional_volume_serial/optional_PADCHK_or_NOPADCHK)
```

For 'library\_name', specify the data set name of the library that contains the program, such as 'SYS1.LINKLIB'.

You can optionally specify a volume serial, such as 123456, SYSRES, or VOLAAA. If you specify a volume serial in this format, the PROGRAM profile protects the program only when the specified library exists on that named volume.

If you do not want to specify a specific volume serial, you have two choices:

1. Omit the volume serial completely. In this case, RACF ignores the volume serial when examining the PROGRAM profile, and considers it a match if the program resides in that library, regardless of the volume serial where the library resides. For this, you could specify:

```
ADDMEM 'SYS1.LINKLIB' //
```

2. Specify a volume serial of '\*\*\*\*\*' for a special case where the library exists on the IPL volume (not on an extension of the IPL volume, however). For this, you could specify:

```
ADDMEM 'SYS1.LINKLIB' /*****'
```

**Guideline:** In general, specify NOPADCHK to simplify your other setup. Refer to [“Choosing between the PADCHK and NOPADCHK operands” on page 314](#) for more information.

For example, a complete ADDMEM specification for a PROGRAM profile might be:

```
ADDMEM('SYS1.LINKLIB'//NOPADCHK)
```

A PROGRAM profile can contain multiple ADDMEM operands, such as:

```
RDEFINE PROGRAM ABC ADDMEM('AAA.LIBRARY1'//NOPADCHK)
RALTER PROGRAM ABC ADDMEM('BBB.LIBRARY1'//NOPADCHK 'CCC.LIBRARY1'//NOPADCHK)
```

You can specify a UACC and an access list for your PROGRAM profile, as you would for other profiles. For the purposes of this discussion, remember that you should be using access values of NONE or READ, rather than EXECUTE. See [“More complex controls: Using EXECUTE access for programs or libraries \(BASIC mode\)” on page 307](#) for more information.

Programs reside in program libraries that can be for public use (those in the system link list) or for limited private use (accessed through JOBLIB or STEPLIB, or through the TSO/E CALL command specifying the data set name). To restrict user's ability to run programs, you might need to protect the program library so the user cannot read from it. In some cases you do not need to provide special protection for the program library, other than ensuring that general users cannot update it.

**Guideline:** Restrict UPDATE access to libraries containing controlled programs, just as you restrict UPDATE access to APF-authorized libraries.

[“Protecting program libraries” on page 309](#) discusses the ways to protect the two types of libraries since there might be cases where you need to do this.

When defining your PROGRAM profiles, also consider the following guidelines:

### Guidelines:

1. If the program you are protecting runs APF-authorized or if you specify the program in the conditional access list to use PADS, you generally do not need to prevent the user from reading the library that contains the program. If the user has READ access to the library, he can copy the program to a different library. However, the copy will not execute successfully because it does not come from an



APF library and, therefore, will not run with APF authority. Additionally, PADS control will not consider the copy a controlled program.

2. If the program you are protecting does not run APF-authorized and you do not intend to use it for PADS, you might want to prevent the user from copying it to another library and executing it from there. However, this is probably only important if the program itself contains sensitive data or algorithms. If a program does not contain sensitive data or algorithms, does not run APF-authorized, and is not used for PADS, do not control its use at all. Instead, consider controlling access to the data that the program uses.

You also should consider the following rules when defining your PROGRAM profiles:

#### Rules:

1. The profiles protect a program only if it resides in the library specified in the ADDMEM operand.
2. If you have multiple libraries that contain the program, and the libraries have different data set names, multiple ADDMEM operand specifications are necessary if you want to protect all copies of the program.
3. You cannot restrict access to programs that reside in the system link pack area (PLPA, MLPA, FLPA, dynamic LPA).
4. With a multiple-user address space (such as CICS Transaction Server), if one user loads a program then another user in the same address space can also execute the program while it remains resident.

#### Restrictions:

1. Some system functions bypass normal MVS contents supervision processing. IMS has some functions that operate this way, for example. RACF program control does not work for programs accessed by such functions, because the system invokes the RACF program control functions only when processing a request to LINK, LOAD, XCTL, or ATTACH a program. RACF program control also will not work for programs loaded from z/OS UNIX file systems, but you can still control such programs using UNIX functions such as permission bits and access control lists (ACLs), that work with RACF.
2. All profiles in the PROGRAM class are discrete profiles. GENERICOWNER is not supported for the PROGRAM class. Even though profiles ending in an asterisk (\*) are allowed in this class, they are not generic profiles, but a special form of discrete profile. That's why we use the terms *specific* and *nonspecific* when discussing PROGRAM profiles, as we mentioned previously.
3. WARNING mode is not supported for PROGRAM profiles.
4. You can specify NOTIFY and auditing for profiles in the PROGRAM class, and for the PROGRAM class itself. However, RACF does not maintain access statistics for PROGRAM profiles.

## When a controlled program has an alias name

When a controlled program has an alias (an alternate name that can be used to execute it), define both the real name and the alias name. This might require additional PROGRAM profiles. For example, to control the use of the DELUSER command and its associated alias DU, and also authorize only the SECADMIN group to use them, issue the following commands.

```
RDEFINE PROGRAM DELUSER UACC(NONE) ADDMEM('SYS1.LINKLIB'//NOPADCHK)
RDEFINE PROGRAM DU      UACC(NONE) ADDMEM('SYS1.LINKLIB'//NOPADCHK)

PERMIT DELUSER ID(SECADMIN) ACCESS(READ) CLASS(PROGRAM)
PERMIT DU      ID(SECADMIN) ACCESS(READ) CLASS(PROGRAM)
```

## Program control by SMFID in BASIC or ENHANCED mode

Program control by system identifier provides a way to restrict access to a program based on system identifier (SMFID). The SMFID is the SID value specified in the active SMFPRMxx member of your system parameter library, such as SYS1.PARMLIB.

To set up program control by system ID, create a conditional access list for the PROGRAM profile that protects the program. To ensure no access to the profile in general, specify UACC(NONE), or

specify ID(\*) ACCESS(NONE) on the PROGRAM profile. Then, permit users on selected systems using WHEN(SYSID(*system-id*)) with the ID and ACCESS operands on the PERMIT command:

```
PERMIT profile-name CLASS(PROGRAM) ID(user or group or *) ACCESS(READ)
      WHEN(SYSID(system-identifier))
```

This restricts the specified users or groups to executing the program only on a system that has a matching system identifier.

## Maintaining a clean environment in BASIC or ENHANCED mode

As previously mentioned, several functions require a clean or controlled program environment:

- Use of PADS
- Use of program access to SERVAUTH resources
- Use of EXECUTE access for programs or libraries
- Improved UNIX security through the definition of FACILITY profile BPX.DAEMON

A *program environment* consists of a job step in a batch job, a started procedure or job, a user's TSO session, or a UNIX address space. In TSO, you can also create a separate program environment by invoking a TSO command using the TSO/E **TSOEXEC** command or the underlying IKJEFTSR programming service.

A *clean* or *controlled* environment indicates that the user has run only programs defined as controlled programs. You define programs through the PROGRAM class in RACF, as shown earlier. You can also define programs that reside in the UNIX file system as controlled programs, by using the UNIX extattr command with the +p option. Refer to [z/OS UNIX System Services Planning](#) for more information.

### Guidelines:

1. To simplify the work of keeping a user's environment clean, define certain libraries using the PROGRAM profile \* or \*\*, rather than trying to define each of the programs in the libraries individually. Also, use PROGRAM \*\* rather than PROGRAM \*. Just as with generic profiles in other classes, using \*\* enables you to list just that profile when you want it with the RLIST command. If you define PROGRAM \*, you will list all profiles in the PROGRAM class when you issue the RLIST command. This might provide several listings and make it more difficult for you to find the profiles you are most interested in.
2. As a starting point for defining a clean environment, examine your system link list and define PROGRAM \*\* entries for the IBM supplied libraries in the link list to ensure that RACF considers all the programs in those libraries controlled. You might also want to add other supplied libraries.

**Examples:** (Note that the following data sets names are only examples and might be different in your installation.)

```
RDEFINE PROGRAM ** ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(READ)
RALTER PROGRAM ** ADDMEM('SYS1.MIGLIB'//NOPADCHK)
RALTER PROGRAM ** ADDMEM('SYS1.CMDLIB'//NOPADCHK)
RALTER PROGRAM ** ADDMEM('SYS1.CSSLIB'//NOPADCHK)
RALTER PROGRAM ** ADDMEM('cee.version.SCEERUN'//NOPADCHK)
RALTER PROGRAM ** ADDMEM( +
  'TCPIP.SEZALOAD'//NOPADCHK +
  'TCPIP.SEZATCP'//NOPADCHK +
  'db2.DSNLOAD'//NOPADCHK +
  'db2.DSNEXIT'//NOPADCHK +
  'ftp.userexits'//NOPADCHK)
```

3. If you include SYS1.LINKLIB in PROGRAM \*\* or PROGRAM \*, you should define specific profiles for programs ICHDSM00 and IRRDPI00, two programs that RACF ships in SYS1.LINKLIB and that you probably do not want available to all users. Refer to *z/OS Security Server RACF System Programmer's Guide* for a description of IRRDPI00, and to *z/OS Security Server RACF Auditor's Guide* for a description of ICHDSM00, which should help you decide which users to authorize for these programs.

```
RDEFINE PROGRAM ICHDSM00 ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(NONE)
RDEFINE PROGRAM IRRDPI00 ADDMEM('SYS1.LINKLIB'//NOPADCHK) UACC(NONE)
```

```
PERMIT ICHDSM00 CLASS(PROGRAM) ID(authorized IDs) ACCESS(READ)
PERMIT IRRDPI00 CLASS(PROGRAM) ID(authorized IDs) ACCESS(READ)
```

4. Specify UACC(READ) for PROGRAM \*\* or PROGRAM \*. With these definitions, you are only trying to keep the environment clean, and are not actually trying to control which users can run programs. Using a value other than READ for UACC on this profile can cause system problems. To avoid some of these problems, RACF grants READ authority when all of the following conditions are true:
  - A user loads a program from SYS1.LINKLIB
  - The program is protected by PROGRAM \*\* or PROGRAM \*
  - The access request is processed using the profile values UACC(NONE) or ID(\*) with ACCESS(NONE)
5. If you have users with the RESTRICTED attribute and want to allow them to execute programs in the clean or controlled program environment, you must specifically authorize them for protected resources in the PROGRAM class. (A UACC of READ does not allow a restricted user to access a RACF-protected resource. See [“Defining restricted user IDs”](#) on page 75.)

#### Tips:

1. For the purposes of keeping the environment clean, you do not need to worry about defining programs in the system link pack area (LPA, PLPA, FLPA, MLPA, dynamic LPA) because RACF always considers those programs controlled.
2. If a user tries to do something that requires a clean environment and RACF disallows that action because the user has a dirty environment, RACF issues messages to the job log or system log describing why the problem occurred.

## More complex controls: Using EXECUTE access for programs or libraries (BASIC mode)

As discussed above using access levels of READ or NONE to allow or restrict access to programs is a simple form of program control. However, in some cases you might have programs that contain sensitive data (such as passwords or PIN numbers) or algorithms. While you might want to let some users execute those programs, you might not want them to examine the data or algorithms contained within the programs.

In these cases, consider using an access level of EXECUTE for the PROGRAM profile, and possibly an access level of EXECUTE for the library that contains the programs. Programs protected this way are called *execute-controlled*. This topic discusses the use of EXECUTE access when running in BASIC program security mode. Using EXECUTE in ENHANCED program security mode is discussed in [“Using EXECUTE access for programs and libraries in ENHANCED mode”](#) on page 316.

If you need to use EXECUTE access, you must ensure that the users running programs do so in a clean environment. Further details on setting up a clean environment for your users is discussed in [“Maintaining a clean environment in BASIC or ENHANCED mode”](#) on page 306. RACF requires a clean environment because, otherwise, a user could write his own program that would load the execute-controlled program into storage and dump its contents to a print file or just copy it to another file of the user's choosing. The user could then examine the program contents and see the data you had wanted to protect. Since RACF requires a clean environment for use of EXECUTE access, a user cannot write his own program to do this, because his program is not controlled (not defined by a PROGRAM profile) and would make the environment become dirty, preventing subsequent access to execute-controlled programs.

You can specify an access level of EXECUTE on the PROGRAM profile for the program that contains the sensitive data or algorithm. You can also specify EXECUTE as an access level for the library containing the program. In either case, when the user attempts to run the execute-controlled program, RACF prevents the loading of the module except into a clean environment. Once all execute-controlled modules the user has run have completed execution and the system has removed them from storage, RACF allows the environment to become dirty if the user then tries to run a non-controlled program.

To decide whether to use EXECUTE for only the PROGRAM profile or also for the DATASET profile that protects the program's library, you must consider certain aspects of the library. See [“Protecting program libraries” on page 309](#) for more information.

## Migrating from BASIC to ENHANCED program security mode

With ENHANCED-WARNING mode, RACF ensures that programs accessing data sets through PADS, accessing SERVAUTH resources by programs, or running execute-controlled programs meet the added restrictions of ENHANCED mode. However, if they do not meet the added restrictions, RACF still allows the access if it would have worked in BASIC mode. This allows you to test your setup to make sure it is suitable for ENHANCED mode while continuing to operate like BASIC mode while you adjust your profiles.

**Guideline:** Use ENHANCED-WARNING program security mode as part of your implementation of ENHANCED program security mode.

When you start this migration, you will have some profiles defined in the PROGRAM class but probably none of them specify APPLDATA( 'MAIN' ) or APPLDATA( 'BASIC' ) as those specifications do not mean anything in BASIC program security mode. You probably do not have the IRR.PGMSECURITY profile defined in the FACILITY class, or else you have it defined but specified APPLDATA( 'BASIC' ).

To begin your migration:

1. Begin figuring out which programs you must define as MAIN or BASIC. You can do this by examining your DATASET and SERVAUTH profiles to see which ones have conditional access lists that specify WHEN(PROGRAM(*program-name*)). To simplify this task, consider using database unload (IRRDBU00) and looking for the type 0402 and 0507 records. (Refer to [z/OS Security Server RACF Macros and Interfaces](#) for the description of these records).
2. From the type 0402 records for DATASET profiles, select the ones that have DSACC\_CATYPE=PROGRAM and find the program name in DSACC\_CANAME. These programs are candidates for the MAIN or BASIC attribute.
3. From the type 0507 records for SERVAUTH profiles, select the ones that have GRCACC\_CATYPE=PROGRAM and find the program name in GRCACC\_CANAME. These programs are candidates for the MAIN or BASIC attribute.
4. For each program (for example, program Y) that you find, determine if users execute program Y, or if they execute some other program (for example, program X) that in turn executes Y. In the first case, you probably only have Y in the conditional access list, and you would define Y as a MAIN or BASIC program. In the second case, both X and Y are in the conditional access list, so you should define X as a MAIN or BASIC program.
5. For each of those programs, determine whether the program needs to be run in batch, TSO, or both. If the program only needs to be run in batch, you can define the program as MAIN. If program needs to be run in TSO or in both batch and TSO, you must consider how the users run the programs in TSO. If the users use the TSO/E **TSOEXEC** command to run the programs, or run them from another program that uses IKJEFTSR to invoke them, you can define the programs as MAIN. If this is not the case, you must define the program as BASIC or find a way for the users to invoke them through TSOEXEC or IKJEFTSR. If these are not possible, you might need to consider the more difficult option of redesigning the programs to work with IKJEFTSR.
6. After you gather your list of programs that you will define with the MAIN or BASIC attribute, use the RLIST command to determine whether you have the programs defined with specific profiles (such as PROGRAM XYZ) or if you have them defined with nonspecific profiles (such as PROGRAM XY\* or PROGRAM \*\*). If the programs have specific profiles, the profiles can be modified using the RALTER command to specify APPLDATA( 'MAIN' ) or APPLDATA( 'BASIC' ). However, if you have protected the programs with nonspecific profiles, you must use the RDEFINE command to define a new specific profile for each of the programs you need to define as MAIN or BASIC.

Once these steps are complete you should:

1. Make a similar examination of IRRDBU00 output, looking at the records that indicate execute-controlled libraries:

- Type 0400 records with a DSBD\_UACC value of EXECUTE
- Type 0402 records with a DSCACC\_ACCESS value of EXECUTE
- Type 0404 records with a DSACC\_ACCESS value of EXECUTE

Examine the programs in those libraries to see which you need to define as MAIN or BASIC, using similar criteria as used for PADS.

2. Look at the records that indicate execute-controlled programs,

- Type 0500 records with GRBD\_CLASS\_NAME of PROGRAM that have GRBD\_UACC of EXECUTE
- Type 0505 records with GRACC\_CLASS\_NAME of PROGRAM that have GRACC\_ACCESS of EXECUTE

Examine the programs in those libraries to see which you need to define as MAIN or BASIC.

When this is complete, you can switch to ENHANCED-WARNING mode to find out if you have any other changes that must be made. To switch to ENHANCED-WARNING mode:

1. Use the RDEFINE command to define IRR.PGMSECURITY profile in the FACILITY class, and specify an APPLDATA value other than ENHANCED or BASIC. For example:

```
RDEFINE FACILITY IRR.PGMSECURITY APPLDATA('ENHWARN')
```

2. Issue the SETROPTS REFRESH command to change modes.

```
SETROPTS WHEN(PROGRAM) REFRESH
```

To ease migration from BASIC to ENHANCED program security mode, the mode switch does not affect systems running any release earlier than z/OS V1R4. It also does not affect jobs, started tasks, or TSO sessions that are already running. For this reason, you should IPL the system at least once while in ENHANCED-WARNING mode to ensure that you test any jobs, started tasks, and TSO users that started before you migrated from BASIC to ENHANCED program security mode.

While running in ENHANCED-WARNING mode, you might receive messages ICH427I or ICH430I to indicate the need for further necessary changes. After receiving the messages, making the relevant changes, and allowing a sufficient test period of running in ENHANCED-WARNING mode without getting further messages, you can switch to ENHANCED program security mode. To do this:

1. Modify the APPLDATA value for the IRR.PGMSECURITY profile by issuing:

```
RALTER FACILITY IRR.PGMSECURITY APPLDATA('ENHANCED')
```

2. Change modes by issuing:

```
SETROPTS WHEN(PROGRAM) REFRESH
```

Again, the mode switch does not affect systems running a release earlier than z/OS V1R4. It also does not affect any jobs, started tasks, or TSO sessions that are already running. So, again, you must IPL the system to fully implement the change. You should not have any problems as a result of this IPL, because you have already IPLed once in ENHANCED-WARNING mode and subsequently fixed any problems that caused RACF messages.

**Guideline:** If you have several systems, consider having a SPECIAL user logged on to TSO on one of them while you IPL to fix any unexpected problem that might arise.

## Protecting program libraries

Program libraries can be for public use or for private (limited) use. You designate a set of libraries as public by placing the libraries in the system link list concatenation, which is SYS1.LINKLIB and any program libraries concatenated to SYS1.LINKLIB through the use of the LNKSTxx member of SYS1.PARMLIB.

A private load library is one that is not in the system link list or one that is explicitly accessed as a private library (even if in the system link list) through:

- A JOBLIB, STEPLIB DD, or other form of tasklib
- A TSO/E CALL command of the form:

```
CALL 'library_name(program_name)'
```

For programs in a system link list library, the user generally does not need access to the library itself, and you could grant the user the following access authorities:

- READ: if the user is authorized to examine the content of programs or copying programs from the library
- NONE: if the user is not authorized to examine the content of programs or copy them.

**Note:** Users do not need access to libraries in the link list in order to run programs from them if they allow the system to load their programs from the link list itself. However, users need access to any library referenced as a private library (for example, using a JOBLIB, STEPLIB, or the TSO/E CALL 'library\_name(program\_name)' command). Even if you have users issuing the TSO/E CALL command, you can still grant NONE authority to the library if they use a different form of the command, CALL \*(program\_name). This command instructs the system to find the program in the LNKLIST or link-pack area without specifically accessing the library itself. If the users cannot use that form of CALL, or need to reference the library as part of JOBLIB or STEPLIB, you must treat the library as a private library if you do not want the user inspecting the library contents.

To protect a private library from a user viewing its content or copying programs from it, grant the user EXECUTE authority to the library itself through the DATASET profile that protects the library. For example, if you have a library named 'AAA.LIBRARY1', you could issue either of the following examples:

### Examples:

1. ADDSD 'AAA.LIBRARY1' UACC(EXECUTE)
2. ADDSD 'AAA.LIBRARY1' UACC(NONE)  
PERMIT 'AAA.LIBRARY1' ID(\*) ACCESS(EXECUTE)

When EXECUTE authority is the highest access level that users have to a private load library, the library is known as an execute-controlled library. If some users have EXECUTE authority, and some have READ authority, the library is execute-controlled for some users and not for others.

**Guideline:** In general, grant READ access rather than EXECUTE unless you have a strong need to prevent users from viewing the contents of a program library. Using EXECUTE requires that you keep the users' program execution environments clean, and requires more administrative effort and restrictions on how the users can access programs from the libraries.

**Restriction:** If you want EXECUTE restrictions to apply to a user who has OPERATIONS authority, you must explicitly PERMIT that user or one of the user's groups to the DATASET profile with EXECUTE authority. UACC(EXECUTE) or ID(\*) ACCESS(EXECUTE) does not make a library execute-controlled for a user with OPERATIONS authority.

## Program access to data sets (PADS) in BASIC mode

Program access to data sets allows users or groups of users to access data sets with higher authority than they would normally have, but only while running a specified program. This is useful when using a program that in some way restricts the user's view of the data, by applying additional validation to some action the user wants to take.

To set up program access to data sets, create a conditional access list for the data set profile protecting the data sets. To do this, specify WHEN(PROGRAM(program\_name)) with the ID and ACCESS operands on the PERMIT command. This specification grants the higher access only while the users run that program.

**Restriction:** Specifying ALTER in a conditional access list usually provides authority no greater than UPDATE (for non-VSAM data sets) or CONTROL (for VSAM data sets). Specifically, ACCESS(ALTER) with WHEN(PROGRAM(...)) does not allow users to delete or allocate the data set through JCL. Deletion or allocation only works through PADS when the program internally performs the deletion or allocation itself by invoking the appropriate system functions, but this is not typical behavior for most programs.

To understand which programs to specify in the conditional access list you need some information about how the user invokes the program and how the program operates. Several cases are illustrated at a high-level. Some of these cases are complex because of the rules that RACF must enforce in order to ensure a secure environment. Therefore, you might need to know more about the design of an application or the way a TSO user invokes a program than you already know. Such cases are described in greater detail but you might want to involve a systems programmer to evaluate them.

1. The user invokes the program PROG1 through JCL:

```
//stepname EXEC PGM=PROG1
//ddname DD DSN=some.dataset,DISP=SHR or OLD or MOD
```

Program PROG1 issues the OPEN for some.dataset itself. In this case, make sure that you have defined PROG1 in the PROGRAM class as a controlled program and specified PROG1 in the DATASET class on the conditional access list:

```
RDEFINE PROGRAM PROG1 UACC(READ)
ADDSD 'some.dataset' ACCESS(NONE)
PERMIT 'some.dataset' ID(userid or *) WHEN(PROGRAM(PROG1)) ACCESS(READ or UPDATE)
```

For the ID operand, you can supply a specific user ID or group name, or you can specify \* to indicate that any user allowed to run the program gets that access level if the standard access list did not grant a sufficient security level. If you define PROG1 with a specific PROGRAM profile, such as PROGRAM PROG1, and provide a specific access list for that profile, you might want to use ID(\*) in the conditional access list so you have only one access list (PROGRAM PROG1) to maintain. If you define PROG1 through PROGRAM \*\* or if you grant UACC(READ) or ID(\*) ACCESS(READ) to PROGRAM PROG1, you might want to supply a specific user ID or group name in the conditional access list for some.dataset.

2. The user invokes the program (PROG1) at the TSO/E READY prompt:

- Directly as a TSO command using the TSO/E CALL command
- Through a REXX exec that uses one of the following REXX statements:
  - address LINKMVS
  - address LINKPGM
  - address ATTACH
  - address ATTCHMVS
  - address ATTCHPGM

PROG1 then issues the OPEN itself.

Keeping a clean environment is generally harder to manage in TSO than in batch. As a result, you must take more care with the definition of PROGRAM \*\* than for the batch case. However, assuming that you have kept the user's environment clean by defining the appropriate programs and libraries, this case is the same as the batch case above and it is not described further.

**Guideline:** For the use of CALL or the REXX functions, make sure you use NOPADCHK when defining the entries in PROGRAM \*\*. If you use PADCHK, you might need to define additional programs in the conditional access list.

3. The user invokes the program (PROG1) under ISPF:

- Directly as a TSO command or using the TSO/E CALL command
- Through the ISPF SELECT CMD(PROG1) or SELECT PGM(PROG1) functions
- Through a REXX exec that uses one of the following REXX statements:
  - address LINKMVS
  - address LINKPGM
  - address ATTACH
  - address ATTCHMVS

- address ATTCHPGM

PROG1 then issues the OPEN itself.

This case is very similar to case 2 above. However, it raises the additional possibility that the user is running in ISPF split-screen mode. In split-screen mode, the user might have initiated several programs and they might all be active at the same time. Suppose that the user has split his ISPF screen, and has program PROGA active on the first screen while trying to run PROG1 on the second screen. In this case, in order for PADS to work for PROG1, PROGA must be defined to RACF, as well. If you defined it with the NOPADCHK attribute, you can simply specify PROG1 in the conditional access list as you did for cases 1 and 2 above. However, if you defined it with PADCHK, you must have a second conditional access list entry granting PROGA authority to the data set. For this situation, you would issue the following commands.

```
RDEFINE PROGRAM PROG1 UACC(READ)
RDEFINE PROGRAM PROGA UACC(READ)
ADDSD 'some.dataset' ACCESS(NONE)

PERMIT 'some.dataset' ID(userid) WHEN(PROGRAM(PROG1))
ACCESS(READ or UPDATE)
PERMIT 'some.dataset' ID(userid or *) WHEN(PROGRAM(PROGA))
ACCESS(READ or UPDATE)
```

4. The user invokes PROG1 under TSO as above (cases 2 or 3), but you cannot keep his environment clean. For example, perhaps the user must run programs that you do not want to define as controlled programs because you do not trust them. In that case, when the user runs such programs the environment becomes dirty, and subsequently he cannot invoke PROG1 in a normal fashion if you want PADS to work.

If you have this situation, and if PROG1 does not need to invoke ISPF services through ISPLINK, you can still allow use of PADS by having the user run PROG1 through the TSO/E TSOEXEC command or (from another program) by the TSO/E IKJEFTSR callable service. Both of these mechanisms can create a new, temporary program environment that is clean and safe to use with PADS. The user might invoke PROG1 by issuing TSOEXEC PROG1 or TSOEXEC CALL \*(PROG1) or some similar operation (such as a REXX exec or CLIST). You can then specify PROG1 in your conditional access list as you did for cases 1 and 2 above. Note that for this case, even if the user is using ISPF split-screen mode, you do not need to worry about putting other programs in the conditional access list.

5. The user invokes PROG1, and PROG1 invokes another program PROG2, and PROG2 OPENS the data set.

### Assumptions:

- a. You have defined both PROG1 and PROG2 as controlled programs using the NOPADCHK attribute.
  - b. PROG1 invokes PROG2 through the MVS LINK or ATTACH services. If PROG1 issues MVS LOAD for PROG2, you just define PROG1 in the conditional access list (allowing you to skip Step 6 below). Conversely, if PROG1 invokes PROG2 through the MVS XCTL service, you just define PROG2 in the conditional access list (allowing you to skip Step 6 below).
  - c. The user invoked PROG1 through some function that uses the MVS ATTACH service:
    - JCL
    - Directly as a TSO command
    - Through TSOEXEC or IKJEFTSR service
    - Through ISPF SELECT CMD(PROG1),
    - Through one of the following REXX statements:
      - address ATTACH
      - address ATTCHMVS
      - address ATTCHPGM
6. Considering these assumptions, you have two ways of specifying your conditional access list:



- a. Since PROG2 issues the OPEN, you can specify WHEN(PROGRAM(PROG2)). You do not need to mention PROG1 in the conditional access list unless you defined PROG1 with PADCHK.
- b. Starting with z/OS Version 1 Release 4, you can specify WHEN(PROGRAM(PROG1)). You do not need to mention PROG2 in the conditional access list unless you defined PROG2 with PADCHK.

**Note:** With programs written in IBM's C, C++, COBOL, or PL/I, you can usually specify the name of the main program in the conditional access list.

**Rules:**

1. When you are creating an entry in a conditional access list, the program name you specify cannot contain any generic characters. For example, if you specify WHEN (PROGRAM(\*) ), WHEN (PROGRAM(IKJ\*) ), or WHEN (PROGRAM(ABC00%)), the entry does not grant any authority. You must specify the exact program name. For more information, refer to [z/OS Security Server RACF Command Language Reference](#).
2. In certain cases, if you have a program (such as PROG1) that runs under ISPF and uses the ISPF LMOPEN service to open a data set, you can grant access using PADS. In order to do this, the user's ISPF dialog must invoke PROG1 using SELECT PGM(PROG1). PROG1 can then use ISPLINK to invoke LMOPEN. You must then specify WHEN(PROGRAM(PROG1)) in the conditional access list.
3. Program access to data sets does not allow different programs with the same name to exist in different libraries requiring different access levels. For example, if you have a program PROGA in library 'load.library1' and also have a program PROGA in 'another.load.library', there would be no distinction between these programs in the conditional access list.
4. ALTER access authority in a conditional access list does not allow the creation or deletion of the data set through JCL. To create or delete a particular data set while running a particular program, and if the data set is to be allocated through JCL, do not grant ALTER authority through PADS. At the time the system creates or deletes the data set through JCL, the program specified in the JCL is not running, and PADS cannot work. You can only allow the creation or deletion of a data set through PADS if the program asks the system to allocate or delete the data set during execution, which is not typical of normal program behavior.

To allow the allocation or deletion of data sets through PADS, write an application to use dynamic allocation (SVC99) to allocate a data set rather than allocate it using JCL. The creation of the data set while running the new program can be allowed, and the new program can then invoke the original application program.

**Note:** As mentioned in case 5 above, beginning with z/OS V1R4, you have more flexibility in specifying programs in a conditional access list. To explain this, some familiarity with certain MVS technical terms is necessary. Generally, programs run under MVS *tasks*. A program running in one task can *attach* a subtask (also known as a *child* task) using the MVS ATTACH service. A program can also invoke another program within the same task (rather than a subtask) using the MVS LINK service. MVS represents a task by a control block called a TCB, and it represents a program running in a task (TCB) with a program request block (PRB). As a result, if a program invokes another program through LINK, you now have one task (TCB) with two programs (PRBs).

When considering what program to specify in the conditional access list, you can use either the program issuing the OPEN (*current* PRB, in MVS terms) or you can specify one of the following:

1. The program represented by the first PRB for the current task (generally the first program run in the task, unless that program used the MVS XCTL service to transfer control to another program)
2. The program represented by the first PRB for the current task's parent task, or the parent task's parent, up to the job step task or the task established by **TSOEXEC** or IKJEFTSR.

Consequently, if:

1. The user executes PROG1
2. PROG1 LINKs to PROG2
3. PROG2 issues the OPEN

you can specify PROG2 or PROG1 in the conditional access list.

If:

1. The user executes PROG1
2. PROG1 LINKs to PROG2
3. PROG2 ATTACHes PROG3
4. PROG3 LINKs to PROG4
5. PROG4 issues the OPEN

you can specify:

- PROG4 (program issuing OPEN)
- PROG3 (first PRB in current task)
- PROG1 (first PRB in parent task)

## Choosing between the PADCHK and NOPADCHK operands

With the RDEFINE and RALTER commands for PROGRAM profiles, you can also specify PADCHK or NOPADCHK with the ADDMEM operand. Your choice affects how PADS operates, and which programs you must specify in the conditional access list for a data set when using PADS.

During PADS processing, RACF looks at the program that issued the OPEN for a data set and at other programs executing in the user's program environment. For example, in a TSO environment, when a user runs a program, such as PROG1, other programs will most likely be running concurrently (including such programs as ISPF and various parts of TSO/E). When RACF makes its decision to allow access through PADS, you must have one program in the conditional access list. This can either be the program that issued the OPEN or a *higher* program in the execution hierarchy (as mentioned before). Additionally, if the user has any other non-LPA programs active, and you defined those programs with PADCHK, you must include them in the conditional access list as well.

RACF also checks the PADCHK/NOPADCHK status of a program when a user tries to run a new program. RACF checks if the user has any data sets already open using PADS. If so, and if you define the new program with PADCHK, RACF ensures that the program is included in the data set's conditional access list before allowing the user to run the program.

PADCHK is the default when you define a PROGRAM to RACF or when you create a new ADDMEM entry for an existing PROGRAM profile.

### Guidelines:

1. If you are defining a program that you trust to operate in a safe manner and not attempt to violate security, specify NOPADCHK to help minimize the size of your conditional access lists and the work you need to do to administer them. Use NOPADCHK for the PROGRAM \*\* or PROGRAM \* profiles and for any other cases where you trust a program to access only the data it should.

PADCHK might be most appropriate for user-written programs that are not completely trusted. This includes the case where you are willing to call them *controlled* and define them with PROGRAM profiles so they can use PADS, but you want them usable only with data sets you have specifically authorized them to. You do not trust them to function appropriately in an environment with other programs running that also make use of PADS.

2. For best results, do not mix PADCHK and NOPADCHK definitions in the same PROGRAM profile. When some members of a PROGRAM profile have been defined with PADCHK and some with NOPADCHK, RACF uses PADCHK for all members.

## Program access to SERVAUTH resources in BASIC or ENHANCED mode

---

You can allow users to access IP addresses only when executing certain programs when you protect the names of network security zones (containing IP addresses) using SERVAUTH class resources. For example, when you control access to network security zones, you can permit network administrators to

access certain zones only when using the **ping** and **traceroute** commands. For more information about using SERVAUTH resources to control access to network security zones, see [z/OS Communications Server: IP Configuration Guide](#).

To set up program control for a SERVAUTH resource (representing a network security zone), create a profile in the SERVAUTH class specifying UACC(NONE), or specify ID(\*) ACCESS(NONE) to ensure no access by general users. Then, permit certain users using WHEN(PROGRAM(*program-name*)) with the ID and ACCESS operands on the PERMIT command:

**Example:**

```
RDEFINE SERVAUTH resource-name UACC(NONE)
PERMIT resource-name CLASS(SERVAUTH) ID(user or group or *) ACCESS(READ)
      WHEN(PROGRAM(program-name))
```

This example permits the specified users or groups to access network security zones protected by SERVAUTH resources only when executing the specified program or command.

Program access to SERVAUTH resources in ENHANCED program security mode operates much the same as it does in BASIC program security mode, with one exception. RACF allows program access to SERVAUTH resources to operate in ENHANCED program security mode only when one of the following is true:

- The program that established the current program environment has the MAIN attribute
- The current program or the first program executed in the current or a parent MVS task has the BASIC attribute

**Note:** For checking MAIN programs, the environment is considered *established* by the initial program executed in the job step, or the initial program executed by **TSOEXEC** or the IKJEFTSR service, or the initial UNIX program **exec()**ed or **spawn()**ed (non-local case only).

As with program access to data sets, you must maintain a clean environment to control program access to SERVAUTH resources. (For details, see [“Maintaining a clean environment in BASIC or ENHANCED mode”](#) on page 306.) Unlike program access to data sets, the PADCHK/NOPADCHK operands have no meaning and are ignored.

## ENHANCED program security mode

ENHANCED program security mode provides better security for the use of PADS, program access to SERVAUTH resources, execute-controlled programs, and optionally for UNIX servers and daemons.

**Rules:**

1. When you choose to run in ENHANCED program security mode, you must identify the programs that you expect users to execute that make use of PADS or program access to SERVAUTH resources, or run execute-controlled programs.
2. You must identify those programs to RACF as either MAIN (preferred) or BASIC programs to indicate your higher level of trust in the way they operate. RACF restricts the use of PADS, program access to SERVAUTH resources, and execute-controlled programs to a program environment established by one of those programs. This provides a more controlled, or restricted, environment for the use of PADS, program access to SERVAUTH resources, and execute-controlled programs. This further ensures that malicious users cannot misuse the system to inappropriately gain access to protected data.

## Program access to data sets (PADS) in ENHANCED mode

This topic addresses *only* the differences between using PADS running in ENHANCED program security mode and PADS running in BASIC program security mode. Refer to [“Program access to data sets \(PADS\) in BASIC mode”](#) on page 310 for additional information.

PADS in ENHANCED program security mode operates the same as PADS in BASIC program security mode, with one exception. RACF allows PADS to operate in ENHANCED program security mode only when one of the following is true:

- The program that established the current program environment has the MAIN attribute
- The current program or the first program executed in the current or a parent MVS task has the BASIC attribute

**Note:** For checking MAIN programs, the environment is considered *established* by the initial program executed in the job step, or the initial program executed by **TSOEXEC** or the IKJEFTSR service, or the initial UNIX program **exec()**ed or **spawn()**ed (non-local case only).

## Using EXECUTE access for programs and libraries in ENHANCED mode

This topic addresses *only* the differences caused by running in ENHANCED program security mode. Refer to [“More complex controls: Using EXECUTE access for programs or libraries \(BASIC mode\)”](#) on page 307 for additional information.

Just as with PADS, ENHANCED program security mode puts additional restrictions on the use of execute-controlled programs. It does not matter whether they are execute-controlled because the user has EXECUTE via a PROGRAM profile or via a DATASET profile; RACF treats both forms of execute-control the same for this purpose.

When running in BASIC program security mode, RACF allows access to execute-controlled programs only when the UACC or access list allowed the access and the user had a clean (controlled) program environment. When running in ENHANCED program security mode, just as with PADS, RACF has an additional requirement. One of the following must be true:

- The program that established the current program environment has the MAIN attribute
- The current program or the first program executed in the current or a parent MVS task has the BASIC attribute

## When to use MAIN or BASIC

When considering whether to define a controlled program as a MAIN program, you should choose one for the following:

- Programs that the user executes directly using JCL
- Programs that the user executes through the TSO/E **TSOEXEC** command or IKJEFTSR callable service

You should not specify MAIN for programs invoked in other ways because RACF does not honor the MAIN attribute in other cases.

Alternatively, if you have a need to use PADS or execute-controlled programs under TSO, but not through TSOEXEC or IKJEFTSR, you can define your trusted initial program as a BASIC program. Using BASIC programs provides less security against malicious users than using MAIN programs, but is required if you decide to use PADS or execute-controlled programs in TSO without using TSOEXEC or IKJEFTSR.

For example:

1. A user must run program PROG1 only in batch using JCL (such as `// EXEC PGM=PROG1`) and PROG1 must OPEN a data set through PADS. You expect the user to only run PROG1 in batch using JCL, and not to run it under TSO/E. You can define PROG1 with the MAIN attribute and allow this usage.
2. A user must run program PROG1 in batch and under TSO/E, but under TSO/E the user can use TSOEXEC or `TSOEXEC CALL *(PROG1)` to run it, and PROG1 must use PADS. Again, you can define PROG1 with the MAIN attribute.
3. A user must run program PROG2, which is an execute-controlled program. PROG2 should be run using JCL or TSOEXEC. You can define PROG2 with the MAIN attribute to allow this.
4. A user must run PROG1 (which uses PADS) or PROG2 (an execute-controlled program) under TSO/E without using TSOEXEC. In this case, you can define PROG1 or PROG2 with the BASIC attribute to allow it to run.
5. A user must run PROG3 (which invokes another program that uses PADS) or PROG4 (which invokes an execute-controlled program). If the user runs PROG3 or PROG4 only using JCL or TSOEXEC, you can

use the MAIN attribute when defining PROG3 or PROG4. However, if the user needs to run PROG3 or PROG4 as a normal TSO command, you would define them using the BASIC attribute.

**Note:** If a user runs an APF-authorized program in TSO, and you have identified that program to TSO/E (through member IKJTSOxx of your system parameter library) as one that should run with APF authority, TSO/E automatically uses the IKJEFTSR service to run the program, and you can define it as MAIN, rather than BASIC.

Effectively, when defining programs, you can indicate several levels of trust in the way that programs operate, based on the attributes you choose. You could define a program using the PADCHK operand, indicating that the program must have an entry in a data set's conditional access list before PADS is allowed with that program in storage. The program is still a controlled program but is not as trusted as a program defined with NOPADCHK. NOPADCHK indicates to RACF that you trust the program not to try to access a data set inappropriately when some other concurrently executing program opens a data set using PADS.

Beyond PADCHK and NOPADCHK, you can identify a program as MAIN, BASIC, or neither. You identify most programs as neither MAIN nor BASIC, by specifying PROGRAM \*, PROGRAM \*\*, or another PROGRAM profile with a name that ends with an asterisk (\*). Again, these programs are controlled, but it is possible that not enough is known about the way they operate to mark them as trusted (which initiates an environment in which PADS or execute-controlled programs are used).

#### Guidelines:

1. When deciding to define a program as MAIN or BASIC, select programs that perform a well-defined set of operations and do not accept the address of a user-supplied routine as a parameter.
2. Do not define the TSO/E terminal monitor program (TMP) or any part of it (such as IKJEFT01, IKJEFT1A, IKJEFT1B, IKJEFT02), or ISPF or any part of it (such as ISPF, ISPSTART, ISPMAIN) as MAIN or BASIC because these programs provide too much of a generalized environment controlled by the user.

If you have chosen to enable this stronger security for UNIX servers and daemons by defining FACILITY profile BPX.MAINCHECK (refer to *z/OS UNIX System Services Planning* for details), you must define some UNIX programs as MAIN, and possibly copy them from the UNIX file system into a standard MVS load library.

## Defining programs as MAIN or BASIC

Once you have decided which of your programs to define as MAIN and which as BASIC (if any), you assign these attributes using the APPLDATA operand on an RDEFINE PROGRAM or RALTER PROGRAM command. Specify an APPLDATA value of 'MAIN' or 'BASIC' on the RDEFINE or RALTER command for a PROGRAM profile whose name does not end with an asterisk (\*). RACF does not honor the MAIN or BASIC attributes if the profile name ends in an asterisk, but only honors it for profiles defining specific programs.

'MAIN' denotes the program as a MAIN program, assuming it is invoked as the first program in a job step or through the TSO/E **TSOEXEC** command or IKJEFTSR service. 'BASIC' denotes the program as one that can access data through PADS, or run EXECUTE-controlled programs, whether or not it runs within an environment started by a MAIN program.

A program cannot be both a MAIN and a BASIC program because RACF honors the APPLDATA specification only if it is 'MAIN' or 'BASIC' (possibly followed by blanks).

**Tip:** If a program needs both the MAIN and BASIC specifications, specify BASIC and accept the reduced level of security for all uses of the program, or create two differently named copies of the program and protect each separately with PROGRAM profiles, specifying one as 'MAIN' and one as 'BASIC'.

Since RACF restricts usage of PADS and execute-controlled programs to environments established by a MAIN or BASIC program, there might be situations where the program that establishes the environment resides in the system link pack area (LPA, PLPA, FLPA, MLPA, or dynamic LPA). If you need to define such

a program to RACF to indicate to RACF that it has the MAIN or BASIC attribute, use a library name of 'LPALST':

```
RDEFINE PROGRAM LPAPROG ADDMEM('LPALST') APPLDATA('MAIN')
```

For programs in the link pack area, RACF allows users to execute the program, regardless of the UACC or access list, and RACF treats the program as having the NOPADCHK attribute. Define it in the PROGRAM class only if you need to provide a MAIN or BASIC attribute for it.

### Note:

1. You can optionally specify blanks at the end of the APPLDATA value. RACF considers, for example, 'MAIN' and 'MAIN ', or 'BASIC' and 'BASIC ' as equivalent.
2. RACF does not validate the APPLDATA value when you specify it with the RDEFINE or RALTER command. When RACF is told to run in ENHANCED program security mode using FACILITY profile IRR.PGMSECURITY, if RACF reads a PROGRAM profile defining a specific program and finds that APPLDATA specifies the 'MAIN' or 'BASIC' values, it assigns the attribute to the program. This is done during the processing of SETROPTS WHEN(PROGRAM) or SETROPTS WHEN(PROGRAM) REFRESH, or during system initialization (IPL). If APPLDATA contains some other value, RACF ignores it without issuing an error message.
3. When invoking MVS load modules through z/OS UNIX (such as **exec()**, **exec\_mvs()**, or an exec where UNIX loads a load module rather than a z/OS UNIX file) the 'MAIN' setting for a PROGRAM is effective only in limited cases. Specifically, it is effective when the **exec()** processing results in a new job step task, but not for the *local spawn* **exec()** processing because this processing results in the creation of a new subtask rather than a job step task. Consequently, **exec()** of load module, **exec\_mvs()**, and non-local **spawn()**, or their z/OS UNIX assembler callable service equivalents, preserve the effect of the MAIN PROGRAM attribute.
4. When failing a request (or allowing it only due to ENHANCED-WARNING processing), RACF issues a message indicating the source and name of the non-MAIN program or the executable file that established the non-MAIN environment.

## How protection works for programs and PADS

---

This topic describes:

- How program control works
- Informational error messages for program control
- Authorization checking for access control to load modules
- Authorization checking for program access to data sets
- How RACF and DFP handle execute-controlled libraries

### How program control works

The WHEN(PROGRAM) operand on the SETROPTS command activates program control and the NOWHEN(PROGRAM) operand deactivates it. You need not activate the PROGRAM class to have program control active. When program control is active, during system initialization (IPL) RACF builds an in-storage profile table composed of the entries in the PROGRAM class (controlled programs). The table entries describe the programs and who can access them. To refresh this table, issue SETROPTS WHEN(PROGRAM) REFRESH.

While building this table, RACF also examines the FACILITY profile IRR.PGMSECURITY, if it exists and the FACILITY class is active, to determine whether you want the system to run in BASIC, ENHANCED, or ENHANCED-WARNING program security mode as described earlier. If you have FACILITY class profiles resident in-storage (by issuing the SETROPTS RACLIST(FACILITY) command), RACF examines the in-storage copy of the profile. Otherwise, RACF reads the profile from the RACF database.



When program control is active, the contents supervision component of MVS invokes RACF before processing each request to load a module. If the user is not authorized to execute the program, the system issues an abend and terminates the request.

**Note:** If a non-APF authorized program issues a LINK or LOAD and passes directory information through the DE operand, and the DE information is for a module from any library that contains a controlled program, contents supervision ignores the supplied DE information and reissues the BLDL macro just as it would if the DE information indicated that the requested module was coming from an APF-authorized library. For more information, refer to *z/OS MVS Programming: Assembler Services Reference IAR-XCT* and *z/OS MVS Programming: Authorized Assembler Services Guide*.

## Informational messages for program control

RACF provides several informational error messages in support of your implementation of program control. These messages are helpful for implementing WHEN(PROGRAM) access to data sets and SERVAUTH class resources, execute-control, and ENHANCED program security mode. They also help when implementing z/OS UNIX servers and daemons in secure systems where the BPX.DAEMON and BPX.SERVER resources are controlled in the FACILITY class. (See *z/OS UNIX System Services Planning* for more information about enabling z/OS UNIX servers and daemons.)

## Authorization checking for access control to load modules

When contents supervision invokes RACF to authorize the loading of a module, RACF makes several checks. Some of these checks involve program-accessed data sets. For more information on program-accessed data sets, see [“Program access to data sets \(PADS\) in BASIC mode”](#) on page 310.

The checks that RACF makes when a user makes a request to load (execute) a program are:

1. If program control has been activated with SETROPTS WHEN(PROGRAM)
2. If program control is active, RACF checks to see whether the program is protected by a profile in the PROGRAM class
3. If the program is not protected, RACF determines whether there are any data sets currently open using PADS or whether there are any execute-controlled programs in storage in the address space.
  - If there are no such data sets or programs, RACF marks the environment dirty (uncontrolled) and allows the user to execute the program.
  - If there are data sets currently opened using PADS, or programs to which the user has only EXECUTE authority, RACF fails the request and the system abends the task. RACF issues message ICH423I to document the execute-controlled programs, or message ICH424I to document the PADS data sets that caused the operation to fail. In this way, RACF prevents uncontrolled programs from gaining access to protected data or programs inappropriately.
4. If the program is protected by a profile but the user does not have at least EXECUTE authority to the program, RACF causes the system to abend the task because the user is not authorized to execute the program.
5. If the program is protected by a profile and the user has only EXECUTE authority to the PROGRAM profile or to the library that contains the program (when the program is loaded from a JOBLIB, STEPLIB, or tasklib), and if the job step or TSO session is running in ENHANCED program security mode, RACF checks whether an appropriate program established the program environment. RACF determines if the first program executed in the job step had the 'MAIN' attribute, or (if necessary) if the program invoked by **TSOEXEC** or IKJEFTSR had the 'MAIN' attribute. If the program does not have MAIN, RACF next determines if the first program run in the current task (TCB) or the first program executed in some parent task had the 'BASIC' attribute. If so, RACF allows the request. Otherwise, RACF fails the request and issues message ICH429I to describe the problem and tell you what program established the environment.
6. If the user is still authorized to execute the program and the program was defined with the PADCHK attribute, RACF checks whether any program-accessed data sets are open.
  - If no program-accessed data sets are open, RACF allows the user to execute the program.

- If program-accessed data sets are open, RACF checks the user and program combination to verify that the combination has at least the same authority to each data set in the list that was required when each data set was opened. For more information on the requirements, refer to [“Program access to data sets \(PADS\) in BASIC mode”](#) on page 310.
  - If the use or program combination has sufficient authority to all of the opened data sets, RACF allows the user to execute the program.
  - If the user or program combination does not have sufficient authority to all of the opened data sets, RACF causes the system to end the task (with abend code 306 or 806).

**Note:** If you are denied access to a requested resource and you implemented program control (with or without PADS), RACF's messages should provide sufficient information to determine the problem. If not, refer to [z/OS Security Server RACF Diagnosis Guide](#) for additional help in determining the cause of the authorization failure.

## Authorization checking for access control to data sets

Whenever a RACROUTE REQUEST=AUTH is issued, RACF performs *normal* authorization checking for access to a data set. In other words, RACF grants the request if the UACC is sufficiently high, if the user's user ID is in the access list with sufficient authority, and so forth. If the user is not granted access to the data set with normal authorization checking, RACF checks the data set's conditional access list if program control is active and the program currently executing is executing as a RACF-controlled program in a clean environment.

RACF authorizes the user to open the program-accessed data set with the currently executing program if all of the following conditions are met:

- The conditional access list contains the name of the currently running program, the name of the first program currently running in the current task (TCB), or the name of the first program currently running in a parent task, with the requested level of access or higher.
- The user's group or user ID is associated with the program name in the conditional access list.
- The current program environment (job step, or task established under TSO/E using **TSOEXEC** or **IKJEFTSR**) is controlled. In other words, it has not loaded an uncontrolled program. If either of these conditions are not met, the environment is considered uncontrolled. The user's attempt to open the program-accessed data set fails and the task ends with abend code 913. RACF issues message ICH417I, specifying what caused the environment to become uncontrolled.
- If the job step or TSO session is running in ENHANCED program security mode, one of the following is true:
  1. The current environment (job step or task created by **TSOEXEC** or **IKJEFTSR**) first ran a program defined with the 'MAIN' attribute.
  2. The current program running in the current task, or the first program run in the current task or a parent task, has the BASIC attribute.

If neither of these conditions is met, the user's attempt to open the program-accessed data set fails and the task ends with abend code 913. RACF issues message ICH426I, specifying the non-MAIN program that established the current environment.

- If there is more than one controlled program running in the current environment (job step or task created by **TSOEXEC** or **IKJEFTSR**), all of those programs defined with the PADCHK attribute have conditional access list entries allowing them to access the data set. If one or more programs in the environment are not authorized, the attempt fails and the task terminates with abend code 913. RACF issues message ICH418I specifying one or more programs that were missing from the conditional access list.

**Note:** If a TSO user has executed a non-controlled program during the current session, and then attempts to access a program-accessed data set, the attempt fails. The TSO user can either log off and log back on, or temporarily regain a controlled environment by invoking the controlled program through the **TSOEXEC** command. When writing a program, you can do the equivalent by invoking the TSO **IKJEFTSR** service. For information on using the **IKJEFTSR** service, see [z/OS TSO/E Programming Guide](#).



## Processing for execute-controlled libraries

When a user or program OPENS a library and has only EXECUTE access to the library, RACF calls this an execute-controlled library. RACF and the system only allow usage of the library for loading programs, and do not allow usage of the library for other I/O that a user's program might try to initiate. Effectively, the user can only use that library as a JOBLIB, STEPLIB, or some other kind of tasklib, for example, by issuing the TSO/E command:

```
CALL 'library_name(program_name)'
```

**Opening an execute-controlled library:** When a program issues an OPEN request to READ a data set, the system invokes RACF. If EXECUTE is the highest access authority that the user is given to the data set, RACF considers the request a failure, and informs the system of the failure, noting that the user did have EXECUTE authority. However, RACF does not audit this case unless the DATASET profile specifies AUDIT(ALL). The system recognizes that the user has EXECUTE and allows the OPEN to proceed, but marks the control blocks that represent that data set to indicate that the user had only EXECUTE authority. This means that for this user, the data set is an execute-controlled data set.

In order to load a program, the program libraries from which it comes must be opened. The user's program does not necessarily code the OPEN. For example, a JOBLIB or STEPLIB statement causes an OPEN to occur before the module is fetched.

### Note:

1. Libraries in the system link list concatenation are opened during IPL, and the programs in them are available to anyone unless the program is defined as a controlled program. The system link list libraries are never considered execute-controlled when a user fetches a program through the system link list. However, if the user specifies a system link list library in a JOBLIB, STEPLIB, or tasklib (for example, through the TSO/E **CALL** command specifying the library name) the library is considered execute-controlled for that user if the user has only EXECUTE authority.
2. If an execute-controlled data set is used in a concatenation of libraries in a DD statement, the entire concatenation is treated as execute-controlled.
3. If the user has any other access authority to the library (READ, UPDATE, CONTROL, or ALTER), the library is opened and the user is granted that access authority.

**Fetching a program from an execute-controlled library:** After an execute-controlled library is opened, the user can attempt to execute (fetch) a program from the library at any time. RACF checks the user's program environment (job step or task established by **TSOEXEC** or IKJEFTSR) to ensure that it is a controlled environment.

### Note:

1. If the user's program environment (job step or task established by **TSOEXEC** or IKJEFTSR) is not controlled, RACF does not allow a program from an execute-controlled library to be fetched into the environment. Therefore, the user's request to load a program from an execute-controlled library fails, and the task typically ends with abend code 306 or 806. RACF issues message ICH419I specifying the program that failed to load, and message ICH420I specifying the program that caused the environment to become uncontrolled.
2. If the user's program environment has used only controlled programs (or no programs at all), the environment is controlled. RACF allows a program from an execute-controlled library to be fetched into a controlled environment. Consequently, RACF grants the user's request to fetch a program from the library.
3. If the user's job step or TSO session is running in ENHANCED program security mode, RACF also checks to ensure that one of the following is true:
  - a. The first program run in the job step or in the task established by **TSOEXEC** or IKJEFTSR had the MAIN attribute
  - b. The first program run in the current task or the first program run in some parent task has the BASIC attribute

If neither of these conditions is true, RACF issues message ICH429I to describe the problem.

Additionally, if the user attempts to fetch a non-controlled program into a controlled environment, RACF ensures that the user has at least READ authority to all RACF-defined programs that are currently residing in the user's address space. RACF does not allow a user to fetch a non-controlled program into an address space that contains an execute-controlled program. The user's attempt to load the non-controlled program fails and the task typically ends with abend code 306 or 806. RACF issues ICH423I specifying one or more execute-controlled programs in the current environment.

If a program not running in supervisor state tries to access an execute-controlled data set other than to load a program through one of the MVS program-loading services, the system denies the request and abends the program.

EXECUTE access authority has meaning only for a partitioned data set that is used as a program library. If you specify EXECUTE for any other type of data set (such as a CLIST or EXEC), effectively the user will have an access authority of NONE.

**Auditing accesses to programs in execute-controlled libraries:** At the time that an execute-controlled library is opened, OPEN does not know if the user will later attempt to read the library or to execute a program from the library. OPEN issues a request for READ authority, and RACF fails the request and sets the reason code to indicate that the user did have EXECUTE authority. OPEN examines the reason code and determines that the user has EXECUTE authority, and allows the OPEN to succeed, but marks its control blocks so that any non-supervisor-state I/O causes an abend. When RACF detects that the user has only EXECUTE authority, it cannot predict if the access will eventually succeed or fail. If the library data set profile indicates that all access attempts should be audited, message ICH408I is issued. If the library data set profile says to audit both successes and failures, RACHECK interprets this as AUDIT(ALL).

This method of determining access can lead to two confusing scenarios:

1. If the profile says AUDIT(ALL), a user attempting to execute a program might receive message ICH408I saying that she does not have READ authority and then see that the program has successfully executed.
2. If the profile says AUDIT(FAILURES), an attempt to read the library can lead to an abend being issued but no message ICH408I being issued since the user has EXECUTE authority.

In addition, the SMF records produced for an attempt to read and an attempt to execute are identical for the same reasons described above.

## Examples of controlling programs and using PADS

---

This topic contains some examples of:

- Defining load modules as controlled programs
- Setting up program access to data sets (PADS)
- Setting up an execute-controlled library
- Setting up program control by system ID

## Examples of defining load modules as controlled programs

This topic contains examples of:

- Protecting programs without using PADS
- Protecting programs that are in several program libraries
- Protecting all programs on the SYSRES volume using '\*\*\*\*\*'
- Protecting a program on any volume by omitting the volume serial number

## Example 1. Protecting programs without using PADS

If you do not intend to use PADS, you need to protect a program in a link list application library to control its use, you can use something similar to the following:

```
1. SETROPTS WHEN(PROGRAM)
   /* activates program control */
2. ADDSD 'APPL.LOADLIB' UACC(NONE)
   /* prevents users from copying programs */
3. RDEFINE PROGRAM MYPROG ADDMEM('APPL.LOADLIB'/VOL123/NOPADCHK) UACC(NONE)
   /* makes MYPROG a controlled program. MYPROG must be a member */
   /* of 'APPL.LOADLIB' on volume 123 */
4. SETROPTS WHEN(PROGRAM) REFRESH
   /* puts the new PROGRAM profile into storage */
```

## Example 2. Protecting programs that are in several program libraries

To protect additional copies of the program in other program libraries, use the ADDMEM operand on the RALTER command:

```
RALTER PROGRAM MYPROG ADDMEM('APPL.ANOTHER.LIBRARY'/VOL456/NOPADCHK)
```

## Example 3. Using '\*\*\*\*\*' as the volume serial number

If you specify '\*\*\*\*\*' as the volume serial number, the profile controls programs in all program libraries residing on the system IPL volume. For example, use this RDEFINE command:

```
RDEFINE PROGRAM program-name ADDMEM(data-set-name/'*****'/NOPADCHK)
```

**Note:** Using '\*\*\*\*\*' works only for the single system IPL volume; it does not work for extensions of the system residence volume. Rather than using '\*\*\*\*\*' you might want to simply omit the volser if you have good controls over how users can create new data sets.

## Example 4. Omitting the volume serial number

You can omit the volume serial number in the ADDMEM, denoting that the controlled program can exist on the specified data set from any volume. For example, use one of these RDEFINE commands:

```
RDEFINE PROGRAM MYPROG ADDMEM('APPL.LOADLIB'//NOPADCHK)
   /* makes MYPROG a controlled program. MYPROG must be a member */
   /* of 'APPL.LOADLIB' */
or
RDEFINE PROGRAM MYPROG ADDMEM('APPL.LOADLIB')
   /* makes MYPROG a controlled program. MYPROG must be a member */
   /* of 'APPL.LOADLIB' */
```

## Examples of setting up program access to data sets

- You have a program named PROG1 in library 'APP.LOADLIB' that users execute in batch, and when using that program you want the users to have UPDATE access to data set 'ABC.DATA'. Otherwise, users should have READ access to the data set. Only users in group GROUPA should have access to PROG1 and 'ABC.DATA'. You should run in BASIC program security mode. Issue the following commands.

```
1. RDEFINE FACILITY IRR.PGMSECURITY APPLDATA('BASIC')
2. ADDSD 'APP.LOADLIB' UACC(READ)
3. RDEFINE PROGRAM PROG1 ADDMEM('APP.LOADLIB'//NOPADCHK) UACC(NONE)
4. PERMIT PROG1 CLASS(PROGRAM) ID(GROUPA) ACCESS(READ)
5. ADDSD 'ABC.DATA' UACC(NONE)
```

6. PERMIT 'ABC.DATA' ID(GROUPA) ACCESS(READ)
7. PERMIT 'ABC.DATA' ID(\*) ACCESS(UPDATE) WHEN(PROGRAM(PROG1))
8. SETR WHEN(PROGRAM)

However, if you have previously issued SETR WHEN(PROGRAM):

```
SETR WHEN(PROGRAM) REFRESH
```

- You have a program named PROG2 in library 'APP.LOADLIB' that users execute in batch, and when using that program you want the users to have UPDATE access to data set 'ABC.DATA'. Otherwise, users should have READ access to the data set. Only users in group GROUPA should have access to PROG2 and 'ABC.DATA'. You should run in ENHANCED program security mode. Issue the following commands:

1. ADDSD 'APP.LOADLIB' UACC(READ)
2. RDEFINE PROGRAM PROG2 ADDMEM('APP.LOADLIB'//NOPADCHK) UACC(NONE) APPLDATA('MAIN')
3. PERMIT PROG2 CLASS(PROGRAM) ID(GROUPA) ACCESS(READ)
4. ADDSD 'ABC.DATA' UACC(NONE)
5. PERMIT 'ABC.DATA' ID(GROUPA) ACCESS(READ)
6. PERMIT 'ABC.DATA' ID(\*) ACCESS(UPDATE) WHEN(PROGRAM(PROG2))
7. RDEFINE FACILITY IRR.PGMSECURITY APPLDATA('ENHANCED')
8. SETR WHEN(PROGRAM)

However, if you have previously issued SETR WHEN(PROGRAM):

```
SETR WHEN(PROGRAM) REFRESH
```

## Example of setting up an execute-controlled library

The following sequence of RACF commands illustrates one way you can set up an execute-controlled library. Assume the program is member XCLPGM in program library KBROWN.PGMLIB2.

1. If program control is not active, enter:

```
SETOPTS WHEN(PROGRAM)
```

After program control is active, it remains active until your installation deactivates it by issuing the SETROPTS command with the NOWHEN(PROGRAM) operand.

2. Define a data set profile to protect the private program library by issuing the ADDSD command with the appropriate operands. The following command defines a data set profile to protect program library KBROWN.PGMLIB2. The command assigns a UACC of EXECUTE to allow all users to execute but not otherwise access the library.

```
ADDSD 'KBROWN.PGMLIB2' UACC(EXECUTE)
```

3. Define a specific profile in the PROGRAM class that protects the controlled program. The following command identifies only program XCLPGM as a controlled program.

```
RDEFINE PROGRAM XCLPGM ADDMEM('KBROWN.PGMLIB2'/VOL6A/NOPADCHK)
```

**Note:** If you intend to run in ENHANCED program security mode, add APPLDATA('MAIN') to this RDEFINE command.

4. Refresh the in-storage program control tables by issuing the following command.

```
SETOPTS WHEN(PROGRAM) REFRESH
```

This ensures that the changes take effect immediately.

## Example of setting up program control by system ID

Suppose your installation has two systems in a sysplex and you want to let user Allen run program MYPROG from SYS1 but not from SYS2. You would use these commands.

```
1. SETROPTS WHEN(PROGRAM)
   /* activates program control */
2. ADDSD 'SYS1.LINKLIB' UACC(EXECUTE)
   /* prevents users from copying programs */
3. RDEFINE PROGRAM MYPROG ADDMEM('SYS1.LINKLIB'/123456/NOPADCHK) UACC(NONE)
   /* makes MYPROG a controlled program. MYPROG must */
   /* be a member of 'SYS1.LINKLIB' on volume 123456 */
4. PERMIT MYPROG CLASS(PROGRAM) ID(ALLEN) ACCESS(READ) WHEN(SYSID(SYS1))
   /* user ALLEN can only run the program from system SYS1 */
5. SETROPTS WHEN(PROGRAM) REFRESH
   /* puts the new PROGRAM profile into storage */
```



---

## Chapter 14. Program signing and verification

This chapter provides information about enabling users to digitally sign programs and enabling RACF to verify signed programs.

This chapter also provides instructions for enabling RACF support for Validated Boot for z/OS. Here, you must perform some set-up activities before using Validated Boot for z/OS to sign IPL data. The term *IPL data* includes IPL text and system load modules, such as the system residence volume (SYSRES) contents. With Validated Boot for z/OS, your installation can ensure that its IPL data is intact, untampered-with, and originates from a trusted build-time source. Information about RACF support for Validated Boot for z/OS is provided in [“IPL data signing for Validated Boot for z/OS”](#) on page 344.

---

### Overview of program signing and verification

You can use RACF to enable and control the digital signature and verification of programs. At your option, RACF can enforce that a program be digitally signed and verified before being loaded for execution on your z/OS system. In addition, you can authorize selected users to digitally sign programs that are bound at your installation.

If your installation develops programs, you might choose to enable users to digitally sign the programs you develop. By signing your programs, your customers or users can ensure that they are executing only valid, unchanged versions of the programs they obtain from you. This might be of interest if you are a software vendor.

**Guideline:** Before you begin enforcing program signing and verification, carefully plan and test procedures to enable your installation to recover from a detected signature failure. Depending on how you customize the signature verification options for a signed program, an improperly signed module might fail to load. If the module is part of a critical business application, ensure that you have a tested recovery procedure in place to minimize the business impact.

RACF supports program signing and verification only for program objects, which are modules stored as members of a partitioned data set extended (PDSE) library.

**Restriction:** Program signing and verification are not supported for the following program modules:

- Program objects that are stored in z/OS UNIX files
- Load modules that are stored as members of a partitioned data set (PDS) library

### Terms to know

In this topic, the following terms are synonymously used to refer to a program module stored as a PDSE member:

- program object
- program module
- program
- module

### Related information

For details about enabling signature verification for the modules of z/OS Cryptographic Services System Secure Sockets Layer (SSL), see [System SSL and FIPS 140-2 in z/OS Cryptographic Services System SSL Programming](#).

For programming information about using the SIGN binder option to sign program modules, see [z/OS MVS Program Management: User's Guide and Reference](#).

## Task roadmap for program signing and signature verification

### About this task

The following table shows the subtasks and associated instructions for enabling a user to digitally sign a program, and enabling RACF to verify a signed program.

Subtask	Associated instructions (see ...)
Enable a user to sign a program using code-signing certificates that you create using RACF.	<a href="#">“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331.</a>
Enable a user to sign a program using code-signing certificates that you obtain from an external certificate authority (CA).	<a href="#">“Steps for enabling a user to sign a program using external code-signing certificates” on page 333.</a>
Optionally, audit your installation's signed programs.	<a href="#">“Steps for discovering if signed programs currently execute on your systems (optional)” on page 338.</a>
Prepare RACF to verify signed programs. (This is a one-time setup.)	<a href="#">“Steps for preparing RACF to verify signed programs (one-time setup)” on page 340.</a>
Verify a signed program.	<a href="#">“Steps for verifying a signed program” on page 342.</a>

## Enabling a user to sign a program

This topic contains the following subtopics:

- [“Overview of enabling a user to sign a program” on page 328](#)
- [“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331](#)
- [“Steps for enabling a user to sign a program using external code-signing certificates” on page 333](#)

## Overview of enabling a user to sign a program

An authorized user, or program builder, can sign a program object using the SIGN binder option at the time the program object is bound. Once signed, the program object contains signature information that can be verified at load time.

This overview contains the following topics:

- [“Certificate objects required for program signing” on page 328](#)
- [“Details about defining IRR.PROGRAM.SIGNING\[.groupid\]\[.userid\] profiles” on page 329](#)
- [“Task roadmap for enabling a user to sign a program” on page 330](#)

## Certificate objects required for program signing

To enable a user to sign a program, you must add certificate objects that meet the following requirements. These certificate objects are added to RACF when you perform the steps in [“Task roadmap for enabling a user to sign a program” on page 330](#).

### Requirements:

- Each user must have access to a key ring or token, called a *program-signing* key ring or token that contains the following certificate objects:
  - An RSA private key to apply the digital signature.
  - The X.509 certificate, called a *code-signing* certificate, that corresponds to the RSA private key.
  - Each certificate-authority (CA) certificate (up to and including the root CA certificate) in the certificate chain of the code-signing certificate.



**Restrictions:**

- No more than ten certificates are supported in the certificate chain of the code-signing certificate.
- The code-signing certificate and each CA certificate in the chain must be signed by using one of the following signature algorithms:
  - sha256WithRSA
  - sha1WithRSA
- The code-signing certificate must have code-signing capability as follows:
  - Either the certificate has no KeyUsage extension, or the certificate has a KeyUsage extension with the digitalSignature indicator enabled.
- Each CA certificate in the chain must have certificate-signing capability in both of the following ways:
  - Either the certificate has no BasicConstraints extension, or the certificate has a BasicConstraints extension with the cA indicator enabled.
  - Either the certificate has no KeyUsage extension, or the certificate has a KeyUsage extension with at least the keyCertSign indicator enabled.

For examples of using RACDCERT GENCERT command to create certificates that meet these requirements, see [“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331](#). Otherwise, contact your external certificate authority (CA) and see [“Steps for enabling a user to sign a program using external code-signing certificates” on page 333](#).

For details about using the RACDCERT GENCERT command, see [z/OS Security Server RACF Command Language Reference](#).

## Details about defining IRR.PROGRAM.SIGNING[.groupid][.userid] profiles

When you perform the subtasks in [“Task roadmap for enabling a user to sign a program” on page 330](#), you define APPLDATA information in one or more discrete profiles in the FACILITY class to specify the following:

- The name of the program-signing key ring that contains all certificate objects required for each user who is an authorized program signer.
- The hash algorithm (or message digestion algorithm) that will be used to sign the program.

### Format of the profile name

The format of the **IRR.PROGRAM.SIGNING[.groupid][.userid]** profile name is based on how you choose to assign program-signing key rings to users who are authorized program signers.

The first three qualifiers of profile name must be IRR.PROGRAM.SIGNING. The rest of the profile name reflects the available options for assigning key rings to signers.

You can optionally append one or two additional qualifiers to the profile name, as shown in the following list. RACF checks the profiles in the order listed, and uses the first profile found that matches as follows:

#### 1. **IRR.PROGRAM.SIGNING.group.userid**

This profile assigns the key ring based on the signer's current-connect group and user ID.

#### 2. **IRR.PROGRAM.SIGNING.userid**

This profile assigns the key ring based on the signer's user ID.

#### 3. **IRR.PROGRAM.SIGNING.group**

This profile assigns the key ring based on the signer's current-connect group.

#### 4. **IRR.PROGRAM.SIGNING**

This profile assigns the same key ring to all authorized signers.

**Rule:** No generic characters are allowed in the name of a IRR.PROGRAM.SIGNING profile.

**Format of the APPLDATA value**

The format of the APPLDATA value in the IRR.PROGRAM.SIGNING profiles is as follows:  
[*hash-algorithm*] [*owning-userid*]/*key-ring-name* or [*hash-algorithm*] [*\*TOKEN\**]/*token-name*

The variables of the APPLDATA value are defined as follows:

- hash-algorithm**  
Specifies the message digestion algorithm to be used for program signing. The default value is SHA256. No other values are supported.
- owning-userid**  
Specifies the user ID that owns the program-signing key ring. If you omit this value, RACF uses the key ring of the authorized program signer.
- /key-ring-name**  
Specifies the fully qualified name of the program-signing key ring. This value must be preceded by the forward slash (/).
- token-name**  
Specifies the program-signing token name. This value must be preceded by \*TOKEN\* and the forward slash (/).

**Examples:**

```
RDEFINE FACILITY IRR.PROGRAM.SIGNING.BUILD.RAMOS
  APPLDATA('BUILDID/BUILD.CODE.SIGNING.KEYRING')
RDEFINE FACILITY IRR.PROGRAM.SIGNING.RAMOS
  APPLDATA('SHA256 RAMOS/RAMOS.CODE.SIGNING.KEYRING')
RDEFINE FACILITY IRR.PROGRAM.SIGNING.PROD
  APPLDATA('/PROD.CODE.SIGNING.KEYRING')
RDEFINE FACILITY IRR.PROGRAM.SIGNING
  APPLDATA('RACFADM/CODE.SIGNING.KEYRING')
```

**Rules:**

- The only space character allowed in the APPLDATA value is the single space following the *hash-algorithm* value. If *hash-algorithm* is omitted, no space is allowed in the APPLDATA value.
- No extraneous characters are allowed in the APPLDATA value.

RACF does not check the format of the APPLDATA value when you define a IRR.PROGRAM.SIGNING profile. RACF checks the format when a user signs a program and RACF finds a matching IRR.PROGRAM.SIGNING profile.

**Task roadmap for enabling a user to sign a program**

**About this task**

The following table shows the subtasks and associated instructions for enabling a user to digitally sign a program. Perform one of the following subtasks for each user you want to enable to digitally sign a program. Base your choice of subtask on how you acquire your code-signing certificates.

Subtask	Associated instructions (see ... )
Enable a user to sign a program using code-signing certificates that you create using RACF.	<a href="#">“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331.</a>
Enable a user to sign a program using code-signing certificates that you obtain from an external certificate authority (CA).	<a href="#">“Steps for enabling a user to sign a program using external code-signing certificates” on page 333.</a>

## Steps for enabling a user to sign a program using RACF code-signing certificates

### Before you begin:

- Determine your IRR.PROGRAM.SIGNING profile structure for assigning program-signing key rings to users who are authorized program signers.

The following steps are based on defining the IRR.PROGRAM.SIGNING.*userid* profile. Therefore, the following examples define a program-signing key ring for each authorized program signer. For details about other options, see [“Details about defining IRR.PROGRAM.SIGNING\[.groupid\]\[.userid\] profiles” on page 329.](#)

### Guidelines:

- If you choose instead to define a IRR.PROGRAM.SIGNING[.groupid][.userid] profile to assign the same key ring to all authorized signers, you can use a profile in the RDATALIB class instead of the FACILITY class to authorize users to access the program-signing ring. A profile in the RDATALIB class allows you to authorize users to access a specific key ring.
- For information about the authority needed to retrieve certificates and keys from a key ring, see the descriptions of the functions DataGetFirst and DataStage in [RACF Authorization for R\\_datalib \(IRRSDL00 or IRRSDL64\) in z/OS Security Server RACF Callable Services.](#)
- If you want to use an ICSF token in place of a RACF key ring, you must define a profile in the CSFSERV class. In this case, READ access to the ICSF resources CSF1TRL and CSF1GAV in the CSFSERV class is required.
- If you specify the PKDS (in Step [“1” on page 331](#)) to store the private key in the ICSF PKA key data set (PKDS), and the CSFSERV and CSFKEYS classes are active, you might need additional authority in those classes. For information about these resources, see [z/OS Cryptographic Services ICSF Administrator's Guide.](#)

Perform the following steps to enable a user to digitally sign a program using code-signing certificates that you create using RACF.

1. If not already created, create a certificate-authority (CA) certificate that you can use to issue code-signing certificates for users who need to sign programs.

**Guideline:** For added security, specify the PKDS option to generate and store the private key in the ICSF PKDS, if available.

### Example:

```
RACDCERT CERTAUTH GENCERT
  SUBJECTSDN(OU('MyCompany Code Signing CA') O('MyCompany') C('US'))
  SIZE(2048) RSA(PKDS) WITHLABEL('MyCompany Code Signing CA')
```

2. For each user, create a code-signing certificate signed by the CA certificate that you created in Step [“1” on page 331.](#)

### Example:

```
RACDCERT ID(RAMOS) GENCERT
  SUBJECTSDN(CN('Ramos Code Signing Cert') O('MyCompany') C('US'))
  SIZE(2048) WITHLABEL('Ramos Code Signing Cert')
  SIGNWITH(CERTAUTH LABEL('MyCompany Code Signing CA'))
  KEYUSAGE(HANDSHAKE DOCSIGN)
```

3. For each user, create a program-signing key ring to hold the certificates that you created in Steps [“1” on page 331](#) and [“2” on page 331.](#)

**Rule:** Specify only uppercase characters in the key ring name. This is because you must specify the ring name in the APPLDATA field of the FACILITY profile you create in Step [“5” on page 332.](#)

### Example:

```
RACDCERT ID(RAMOS) ADDRING(RAMOS.CODE.SIGNING.KEYRING)
```

- 
4. Add both of the certificates that you created in Steps “1” on page 331 and “2” on page 331 to the key ring that you created in Step “3” on page 331.

**Rule:** The code-signing certificate must be the default certificate in the ring.

### Example:

```
RACDCERT ID(RAMOS) CONNECT(CERTAUTH LABEL('MyCompany Code Signing CA')  
RING(RAMOS.CODE.SIGNING.KEYRING))  
RACDCERT ID(RAMOS) CONNECT(ID(RAMOS) LABEL('Ramos Code Signing Cert') DEFAULT  
RING(RAMOS.CODE.SIGNING.KEYRING))
```

- 
5. For each user, create a FACILITY class profile that specifies the hash algorithm and the name of the key ring to be used whenever the user digitally signs a program module.

### Example:

```
RDEFINE FACILITY IRR.PROGRAM.SIGNING.RAMOS  
APPLDATA('SHA256 RAMOS/RAMOS.CODE.SIGNING.KEYRING')
```

- 
6. Permit each user, if not already authorized, to access his own key rings by administering a profile in either the FACILITY or the RDATA LIB class.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)  
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RAMOS) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB RAMOS.CODE.SIGNING.KEYRING.LST UACC(NONE)  
PERMIT RAMOS.CODE.SIGNING.KEYRING.LST CLASS(RDATA LIB)  
ID(RAMOS) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

---

You have now enabled a user to digitally sign a program using code-signing certificates that you created using RACF.

## Steps for enabling a user to sign a program using external code-signing certificates

### Before you begin:

- Obtain or locate the root certificate-authority (CA) certificate of an external CA and store it in a cataloged, variable-byte (VB) MVS data set.
- Determine your IRR.PROGRAM.SIGNING profile structure for assigning program-signing key rings to users who are authorized program signers.

The following steps are based on defining the IRR.PROGRAM.SIGNING.*userid* profile. Therefore, the following examples define a program-signing key ring for each authorized program signer. For details about other options, see [“Details about defining IRR.PROGRAM.SIGNING\[.groupid\]\[.userid\] profiles” on page 329.](#)

**Guideline:** If you opt instead to define the IRR.PROGRAM.SIGNING profile to assign the same key ring to all authorized signers, you might use a profile in the RDATA LIB class instead of the FACILITY class to authorize users to access the program-signing ring. A profile in the RDATA LIB class allows you to authorize users to a specific key ring. For details, see [RACF Authorization for R\\_data lib \(IRRSDL00 or IRRSDL64\) in z/OS Security Server RACF Callable Services.](#)

Perform the following steps to enable a user to digitally sign a program using code-signing certificates that you obtain from an external certificate-authority (CA).

1. If not already done, add the root CA certificate of the external CA to RACF, specifying the name of the data set where it is stored.

### Example:

```
RACDCERT CERTAUTH ADD(CA.CERT.DSN) WITHLABEL('MyCompany Code Signing CA')
```

2. For each user, obtain a code-signing certificate from the external CA and add it to RACF. To do so, perform the following sub-steps.
  - a. Create a self-signed code-signing certificate (as a placeholder) that will be signed by the external CA.

### Example:

```
RACDCERT ID(RAMOS) GENCERT
  SUBJECTSDN(CN('Ramos Code Signing Cert') O('MyCompany') C('US'))
  SIZE(2048) WITHLABEL('Ramos Code Signing Cert')
  KEYUSAGE(HANDSHAKE DOCSIGN)
```

- b. Create a PKCS #10 certificate request based on the placeholder certificate you created in Step “2.a” on page 333, specifying the name of the MVS data set where the certificate request will be stored.

### Example:

```
RACDCERT ID(RAMOS) GENREQ(LABEL('Ramos Code Signing Cert'))
  DSN(RAMOS.CERT.REQUEST.DSN)
```

- c. Send the MVS data set (for example, RAMOS.CERT.REQUEST.DSN) containing the stored certificate request to the external CA.
- d. Receive the signed certificate returned by the external CA and store it in a cataloged, variable-byte (VB) MVS data set (for example, RAMOS.CERT.DSN).
- e. Add the new signed certificate to RACF, replacing the placeholder certificate you created in Step “2.a” on page 333.

### Example:

```
RACDCERT ID(RAMOS) ADD(RAMOS.CERT.DSN) WITHLABEL('Ramos Code Signing Cert')
```

- 
- For each user, create a program-signing key ring to hold the external certificates you added in Steps “1” on page 333 and “2” on page 333.

**Rule:** Specify only uppercase characters in the key ring name. This is because you must specify the ring name in the APPLDATA field of the FACILITY profile you create in Step “5” on page 334.

### Example:

```
RACDCERT ID(RAMOS) ADDRING(RAMOS.CODE.SIGNING.KEYRING)
```

- 
- Connect both of the certificates you added in Steps “1” on page 333 and “2” on page 333 to the key ring you created in Step “3” on page 334.

**Rule:** The code-signing certificate must be the default certificate in the ring.

### Example:

```
RACDCERT ID(RAMOS) CONNECT(CERTAUTH LABEL('MyCompany Code Signing CA')  
RING(RAMOS.CODE.SIGNING.KEYRING))  
RACDCERT ID(RAMOS) CONNECT(ID(RAMOS) LABEL('Ramos Code Signing Cert') DEFAULT  
RING(RAMOS.CODE.SIGNING.KEYRING))
```

- 
- For each user, create a FACILITY class profile that specifies the hash algorithm and the name of the key ring to be used whenever the user digitally signs a program module.

### Example:

```
RDEFINE FACILITY IRR.PROGRAM.SIGNING.RAMOS  
APPLDATA('SHA256 RAMOS/RAMOS.CODE.SIGNING.KEYRING')
```

- 
- Permit each user to access his own key rings, if not already authorized, by administering a profile in either the FACILITY or the RDATALIB class.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)  
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RAMOS) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATALIB class:

```
RDEFINE RDATALIB RAMOS.CODE.SIGNING.KEYRING.LST UACC(NONE)  
PERMIT RAMOS.CODE.SIGNING.KEYRING.LST CLASS(RDATALIB)  
ID(RAMOS) ACCESS(READ)
```

- If the RDATALIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATALIB) RACLIST(RDATALIB)
```

- If the RDATALIB class is already active and RACLISTed, refresh it.

SETROPTS RACLIST(RDATALIB) REFRESH

-----

You have now enabled a user to digitally sign a program using code-signing certificates that you obtained from an external certificate-authority (CA).

## Enabling RACF to verify signed programs

This topic contains the following subtopics:

- [“Overview of enabling RACF to verify signed programs” on page 335](#)
- [“Steps for discovering if signed programs currently execute on your systems \(optional\)” on page 338](#)
- [“Steps for preparing RACF to verify signed programs \(one-time setup\)” on page 340](#)
- [“Steps for verifying a signed program” on page 342](#)

## Overview of enabling RACF to verify signed programs

You can enable RACF to verify signed programs by performing some (one-time) setup steps and then specifying the programs you want RACF to verify. Once RACF verifies the authority of a user to execute a controlled program, RACF (optionally) performs signature verification at the time the program object is loaded for execution.

This overview contains the following topics:

- [“Initializing RACF program signature verification” on page 335](#)
- [“Certificate objects required for verifying signed programs” on page 335](#)
- [“Details about defining the IRR.PROGRAM.SIGNATURE.VERIFICATION profile” on page 336](#)
- [“Customizing the SIGVER segment of PROGRAM profiles” on page 336](#)
- [“Delegating the authority for specifying signature verification options” on page 337](#)
- [“Discovering if signed programs currently execute on your systems” on page 337](#)
- [“Task roadmap for enabling RACF to verify signed programs” on page 338](#)

## Initializing RACF program signature verification

When you perform [“Steps for preparing RACF to verify signed programs \(one-time setup\)” on page 340](#), you initialize program signature verification. These steps include preparing several certificate objects and general resource profiles, defining the program verification module (IRRPVERS) as a signed program that must be verified when it is loaded, and finally loading and successfully verifying the signature of the IRRPVERS module for the first time.

To verify IRRPVERS, RACF uses the IBM root CA certificate that is supplied with RACF. For a listing of the output from the RACDCERT LIST command for this certificate, see [Appendix C, “Listings of RACF supplied certificates,” on page 709](#).

## Certificate objects required for verifying signed programs

To enable RACF to verify signed programs, you must add certificate objects that meet the following requirements. These certificate objects are added to RACF when you perform the steps in [“Task roadmap for enabling RACF to verify signed programs” on page 338](#).

### Requirements:

- You must add the TRUST attribute to the code-signing certificate-authority (CA) certificate that is supplied with RACF, so that RACF can use it.
- You must add a key ring, called the *signature-verification* key ring, and connect the RACF code-signing CA certificate and a root CA certificate for *each* trusted program signer. (For a program signed by a user

in your installation, this root CA certificate is the CA certificate you added to the user's code-signing key ring in [“Enabling a user to sign a program”](#) on page 328.)

Your installation can have only *one* signature-verification ring. This single ring represents your installation's trust policy for trusted program signers, and must contain the RACF code-signing CA certificate and all root CA certificates required to verify all the signed programs that you want RACF to verify.

**Restriction:** Do not use a PKCS #11 token as a substitute for the *signature-verification* key ring.

- Each root CA certificate must be signed using one of the following signature algorithms:
  - sha256WithRSAEncryption
  - sha1WithRSAEncryption
- Each root CA certificate must have certificate-signing capability in *both* of the following ways:
  - Either the certificate has no BasicConstraints extension, *or* the certificate has a BasicConstraints extension with the cA indicator enabled.
  - Either the certificate has no KeyUsage extension, *or* the certificate has a KeyUsage extension with at least the keyCertSign indicator enabled.

## Details about defining the IRR.PROGRAM.SIGNATURE.VERIFICATION profile

When you perform the subtasks in [“Task roadmap for enabling RACF to verify signed programs”](#) on page 338, you define APPLDATA information in a discrete profile called IRR.PROGRAM.SIGNATURE.VERIFICATION in the FACILITY class to specify the following:

- The name of the signature-verification key ring that contains all certificate objects required to verify each signed program.

**Rule:** No generic characters are allowed in the name of the IRR.PROGRAM.SIGNATURE.VERIFICATION profile.

### Format of the APPLDATA value

The format of the APPLDATA value in the IRR.PROGRAM.SIGNATURE.VERIFICATION profile is as follows:

*owning-userid/key-ring-name*

The variables of the APPLDATA value are defined as follows:

#### ***owning-userid***

Specifies the user ID that owns the signature-verification key ring.

#### ***/key-ring-name***

Specifies the fully qualified name of the signature-verification key ring. This value must be preceded by the forward slash (/).

#### **Example:**

```
RDEFINE FACILITY IRR.PROGRAM.SIGNATURE.VERIFICATION
  APPLDATA('RACFADM/CODE.SIGNATURE.VERIFICATION.KEYRING')
```

**Rule:** No spaces or extraneous characters are allowed in the APPLDATA value.

RACF does not check the format of the APPLDATA value when you define a IRR.PROGRAM.SIGNATURE.VERIFICATION profile. RACF typically checks the format when it verifies the signature of a signed program.

## Customizing the SIGVER segment of PROGRAM profiles

The SIGVER segment of a profile in the PROGRAM class contains the signature verification options that apply to one or more programs that are controlled by the profile. Customize the fields of the SIGVER segment using the SIGVER operand of the RALTER or RDEFINE command.



When you customize the SIGVER segment of a PROGRAM profile, you can specify options for the following suboperands of the SIGVER operand:

#### **SIGREQUIRED**

Specifies whether the program must be digitally signed.

#### **FAILLOAD**

Specifies the conditions under which the program should fail to load in the event of a signature verification failure.

#### **SIGAUDIT**

Specifies which signature verification events are logged.

For details about customizing the SIGVER segment using the RALTER and RDEFINE commands, see [z/OS Security Server RACF Command Language Reference](#).

For examples of customizing the SIGVER segment, see [“Steps for verifying a signed program” on page 342](#).

If you want to delegate authority for customizing the SIGVER segment to auditors or other users who do not have the SPECIAL attribute, see [“Delegating the authority for specifying signature verification options” on page 337](#).

### **Delegating the authority for specifying signature verification options**

If you want to delegate the authority for specifying signature verification options to users who do not have the SPECIAL attribute, you must use field-level access checking to authorize UPDATE access to the appropriate fields in the SIGVER segment of PROGRAM class profiles.

Users with the AUDITOR attribute cannot specify auditing options for signature verification unless you authorize them with UPDATE access to the SIGAUDIT field.

The following example authorizes a group called SIGNGRP to specify all signature verification options, and authorizes a second group called AUDGRP to control only the auditing options for signature verification.

#### **Example:**

```
SETROPTS CLASSACT(FIELD) RACLIST(FIELD)

RDEFINE FIELD PROGRAM.SIGVER.* UACC(NONE)
PERMIT PROGRAM.SIGVER.* CLASS(FIELD) ID(SIGNGRP) ACCESS(UPDATE)

RDEFINE FIELD PROGRAM.SIGVER.SIGAUDIT UACC(NONE)
PERMIT PROGRAM.SIGVER.SIGAUDIT CLASS(FIELD) ID(SIGNGRP AUDGRP) ACCESS(UPDATE)

SETROPTS RACLIST(FIELD) REFRESH
```

For a complete list of the resource name qualifiers that control each field of the SIGVER segment, see the details about the SIGVER segment in [Table 18 on page 205](#).

### **Discovering if signed programs currently execute on your systems**

You can optionally enable SMF logging of signature verification events by performing [“Steps for discovering if signed programs currently execute on your systems \(optional\)” on page 338](#). By doing so, you can later examine the SMF records using the SMF data unload utility (IRRADU00) to discover if any of your controlled programs are digitally signed and if so, by whom. Once you identify a signer, obtain the signer's root CA in preparation for completing [“Steps for verifying a signed program” on page 342](#).

For information about using the SMF data unload utility (IRRADU00), see [z/OS Security Server RACF Auditor's Guide](#).

To enable SMF logging for this purpose, modify one or more PROGRAM profiles to specify the following signature verification options. Using these specific options ensures that no load failures occur due to signature verification failures.

### **SIGREQUIRED(NO)**

Specifies that no digital signatures are required.

### **FAILLOAD(NEVER)**

Specifies that no program load should fail due to a signature verification failure.

### **SIGAUDIT(ALL)**

Specifies that all signature verification events will be logged, regardless of success or failure.

Once you perform “Steps for discovering if signed programs currently execute on your systems (optional)” on page 338, if any controlled program is digitally signed, RACF will attempt to verify the signature upon load. Each signature verification will result in a failure until you complete “Steps for preparing RACF to verify signed programs (one-time setup)” on page 340 and “Steps for verifying a signed program” on page 342. Each signature verification failure will be logged to SMF and related error messages will be issued to the console.

#### **Sample error messages:**

```
ICH441I Program signature error 0x00/0x00000074 for program PROGXYZ
        in library LXYZR11.LIBRARY. Load processing continues.
ICH450I The RACF program verification module is not loaded.
        Program signature verification is not available.
```

## **Task roadmap for enabling RACF to verify signed programs**

### **About this task**

The following table shows the subtasks and associated instructions for enabling RACF to verify signed programs.

<b>Subtask</b>	<b>Associated instructions (see ... )</b>
Optionally discover if signed programs currently execute on your systems.	<a href="#">“Steps for discovering if signed programs currently execute on your systems (optional)” on page 338.</a>
Prepare RACF to verify signed programs. (This is a one-time setup.)	<a href="#">“Steps for preparing RACF to verify signed programs (one-time setup)” on page 340.</a>
Verify a signed program.	<a href="#">“Steps for verifying a signed program” on page 342.</a>

## **Steps for discovering if signed programs currently execute on your systems (optional)**

### **Before you begin:**

- For background information about these steps, see [“Discovering if signed programs currently execute on your systems” on page 337.](#)
- **Important:** When specifying SIGVER options for a generic program profile (such as the \*\* profile) for the purpose of discovering signed programs, observe the following guidelines. They are based on the assumption that while you are beginning to evaluate program verification, you have not yet planned for the impact it might have on your installation.

### **Guidelines:**

- Set the SIGVER options as shown in the examples in Step [“1” on page 339.](#)
- Avoid specifying SIGREQUIRED(YES) for a generic program profile because it might cause excessive logging and failure messages to the console, based on the SIGAUDIT setting.
- Avoid specifying FAILLOAD(BADSIGNONLY) or FAILLOAD(ANYBAD) for a generic program profile because it might fail critical programs and cause system failure.
- You might need assistance from your system programmer to complete Step [“4” on page 340.](#)

Optionally, perform the following steps to enable RACF to perform and log signature verification events for one or more controlled programs, without causing any program loads to fail due to signature verification failures.

1. Modify one or more PROGRAM class profiles that protect controlled programs to enable signature verification and logging without causing any program load to fail due to a signature verification failure.

**Example 1:**

```
RALTER PROGRAM MYPROG14
  SIGVER(SIGREQUIRED(NO) FAILLOAD(NEVER) SIGAUDIT(ALL))
```

**Important:** When a controlled program has an *alias* (an alternate name that can be used to execute it), define both the real name and the alias name. This might require additional PROGRAM profiles. For an example, [“When a controlled program has an alias name” on page 305](#).

If your installation already defined a PROGRAM class profile called \*\* to control all programs residing in controlled program libraries, you might want to enable signature verification logging for all of these programs by modifying this profile.

**Example 2:**

```
RALTER PROGRAM **
  SIGVER(SIGREQUIRED(NO) FAILLOAD(NEVER) SIGAUDIT(ALL))
```

**Important:** For important guidelines about modifying a generic program profile, such as the \*\* profile shown in Example 2, see **"Before you begin"**.

- 
2. Activate your profile changes in the PROGRAM class, as follows.

- If the PROGRAM class is not already active, activate the PROGRAM class.

**Example:**

```
SETROPTS WHEN(PROGRAM)
```

- If the PROGRAM class is already active, refresh the PROGRAM class.

**Example:**

```
SETROPTS WHEN(PROGRAM) REFRESH
```

- 
3. Display the SIGVER segment information for the profiles you modified in Step [“1” on page 339](#) and review your options.

**Example:**

```
RLIST PROGRAM ** SIGVER NORACF
```

Your results will be similar to the following:

**Results:**

```
PROGRAM **
SIGVER INFORMATION
-----
SIGREQUIRED=NO
FAILLOAD=NEVER
SIGAUDIT=ALL
```

4. (Optional) Ensure that your system programmer enables caching for program signature verification using the virtual lookaside facility (VLF) and restarts VLF. This avoids increasing load times for signed programs.

For programming information, see [VLF considerations for program signature verification in z/OS Security Server RACF System Programmer's Guide](#).

---

You have now enabled RACF to log signature verification events for one or more controlled programs, *without* causing any program loads to fail due to signature verification failures.

Now, each time a signed program loads, RACF logs a signature verification failure to SMF and issues a failure message to the console. These failures continue until you complete [“Steps for preparing RACF to verify signed programs \(one-time setup\)”](#) on page 340 and [“Steps for verifying a signed program”](#) on page 342.

After an appropriate time interval during which these programs are loaded, examine the output of the SMF unload utility (IRRADU00) to discover if any controlled programs are digitally signed and if so, by whom. Once you identify the signer, obtain the signer's root CA in preparation for completing [“Steps for verifying a signed program”](#) on page 342.

When your analysis is complete, proceed to [“Steps for preparing RACF to verify signed programs \(one-time setup\)”](#) on page 340.

## Steps for preparing RACF to verify signed programs (one-time setup)

By performing these steps, you prepare RACF to verify signatures. However, RACF does not begin verifying the signatures of your programs until you complete [“Steps for verifying a signed program”](#) on page 342.

**Before you begin:** You will need assistance from your system programmer to complete Step [“8”](#) on page 341.

Perform the following steps to prepare RACF to verify signed programs. Complete these steps one time only.

1. Create a key ring for your installation to use for signature verification. Specify the ring name of your choice.

### Example:

```
RACDCERT ID(RACFADM) ADDRING(CODE.SIGNATURE.VERIFICATION.KEYRING)
```

**Rule:** Specify only uppercase characters in the key ring name. This is because you must specify the ring name in the APPLDATA field of the FACILITY profile you create in Step [“4”](#) on page 341.

**Guideline:** Do not skip this step so that you can use the virtual CERTAUTH key ring. For best performance, define your signature verification ring by issuing a RACDCERT ADDRING command.

- 
2. Add the TRUST attribute to the code-signing CA certificate that is supplied for RACF program verification. See [Appendix C, “Listings of RACF supplied certificates,”](#) on page 709.

### Example:

```
RACDCERT CERTAUTH ALTER(LABEL('<label of the STG code signing CA certificate>')) TRUST
```

If the DIGTCERT class is RACLISTed, refresh the in-storage profiles.

```
SETR RACLIST(DIGTCERT) REFRESH
```

- 
3. Add the code-signing CA certificate that is supplied with RACF to the key ring you created in Step [“1”](#) on page 340.

**Example:**

```
RACDCERT ID(RACFADM) CONNECT(CERTAUTH LABEL('<label of the STG code signing
CA certificate>') RING(CODE.SIGNATURE.VERIFICATION.KEYRING))
```

- 
4. Create a FACILITY class profile that specifies the name of the key ring you created in Step “1” on page 340.

**Example:**

```
RDEFINE FACILITY IRR.PROGRAM.SIGNATURE.VERIFICATION
APPLDATA('RACFADM/CODE.SIGNATURE.VERIFICATION.KEYRING')
```

- 
5. Activate your profile changes in the FACILITY class, as follows.

- If the FACILITY class is not already active, activate and RACLIST the FACILITY class.

**Example:**

```
SETOPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh the FACILITY class.

**Example:**

```
SETOPTS RACLIST(FACILITY) REFRESH
```

- 
6. Create a PROGRAM class profile to control the program verification module called IRRPVERS and specify the signature verification options.

**Example:**

```
RDEFINE PROGRAM IRRPVERS ADDMEM('SYS1.SIEALNKE'//NOPADCHK) UACC(READ)
SIGVER(SIGREQUIRED(YES) FAILLOAD(ANYBAD) SIGAUDIT(ANYBAD))
```

- 
7. Activate your profile changes in the PROGRAM class, as follows.

- If the PROGRAM class is not already active, activate the PROGRAM class.

**Example:**

```
SETOPTS WHEN(PROGRAM)
```

- If the PROGRAM class is already active, refresh the PROGRAM class.

**Example:**

```
SETOPTS WHEN(PROGRAM) REFRESH
```

- 
8. Contact your system programmer to complete this step.

- a. Notify your system programmer to initialize program signature verification by running the IRRVERLD program. The IRRVERLD program loads and verifies the program verification module (IRRPVERS) and must be run on all systems in a sysplex.

For programming information, see [Initializing RACF verification of signed programs \(IRRVERLD\) in z/OS Security Server RACF System Programmer's Guide](#).

- b. Check with your system programmer to ensure that IRRVERLD successfully completed. If it did not, work with your system programmer to resolve error messages and then rerun.

- c. (Optional) Ensure that your system programmer enables caching for program signature verification using the virtual lookaside facility (VLF) and restarts VLF. This avoids increasing load times for signed programs.

For programming information, see [VLF considerations for program signature verification in z/OS Security Server RACF System Programmer's Guide](#).

---

When the IRRVERLD program successfully executes, you have completed the one-time setup to prepare RACF to verify signed programs. To begin verifying one of your own signed programs, proceed to [“Steps for verifying a signed program” on page 342](#).

## Steps for verifying a signed program

### Before you begin:

- Do not perform these steps until you complete [“Steps for preparing RACF to verify signed programs \(one-time setup\)” on page 340](#).
- Do not perform these steps to enable RACF to verify the modules of z/OS System SSL. Instead, see [System SSL and FIPS 140-2 in z/OS Cryptographic Services System SSL Programming](#).
- For each signed program you want RACF to verify:
  - Obtain or locate the root certificate-authority (CA) certificate of each code signer.
    - If the program was acquired from a software vendor, review the software documentation for information about obtaining the vendor's root CA certificate and adding it to RACF. Once you obtain the root CA certificate, store it in a cataloged, variable-byte (VB) MVS data set.
    - If the program was signed by your installation (in Step “1” on page 331 of [“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331](#)) or signed by an external CA (in Step “1” on page 333 of [“Steps for enabling a user to sign a program using external code-signing certificates” on page 333](#)), locate the label name of root CA certificate in the RACF database.

To list all CA certificates in the RACF database, issue the RACDCERT LIST CERTAUTH command.

- Determine if the program is already controlled by a generic program profile. If so, create a new program profile to control this program or ensure that all programs controlled by this profile are signed. An unsigned program controlled by a profile with the SIGVER options shown in Step “3” on [page 343](#) will fail to load. If several similarly named programs can be verified using the same SIGVER options, you might choose to create a generic profile such as ABC\*. If you do, ensure that no other programs are unintentionally verified based on their similar program names.

**Important:** Do not specify the SIGVER options shown in Step “3” on [page 343](#) for the \*\* program profile because this might fail critical programs and lead to system failure.

- If you share the RACF database with other z/OS systems, determine if another version of this program runs on a shared system. If so, ensure that the version on the shared system is signed. Alternatively, control the other version with a separate program profile.
- Determine if the program has an *alias* (an alternate name that can be used to execute it). If so, control both the real name and the alias name. This might require an additional PROGRAM profile in Step “3” on [page 343](#). For an example, [“When a controlled program has an alias name” on page 305](#).

Perform the following steps for *each* signed program you want RACF to verify.

1. Add the root CA certificate of the code signer to RACF as a trusted CA.

Skip this step if you created the root CA of the code signer (in Step “1” on page 331 of [“Steps for enabling a user to sign a program using RACF code-signing certificates” on page 331](#)), or if you obtained the root CA of the code signer from an external CA and added it to RACF (in Step “1” on page 333 of [“Steps for enabling a user to sign a program using external code-signing certificates” on page 333](#)).

- a. If you obtained the root CA certificate of the code signer from a software vendor, add it to RACF, specifying the name of the data set where it is stored.

**Example:**

```
RACDCERT CERTAUTH ADD(VENDOR.CA.CERT.DSN)
  WITHLABEL('Vendor Code Signing CA')
  TRUST
```

- b. If the vendor's root CA certificate is already added to RACF, add the TRUST attribute if it is not already trusted.

**Example:**

```
RACDCERT CERTAUTH ALTER(LABEL('Vendor Code Signing CA')) TRUST
```

- 
2. Add the root CA certificate to the key ring that your installation uses for signature verification. This is the ring you created in Step “1” on page 340 of [“Steps for preparing RACF to verify signed programs \(one-time setup\)”](#) on page 340.

**Examples:**

```
RACDCERT ID(RACFADM) CONNECT(CERTAUTH LABEL('Vendor Code Signing CA')
  RING(CODE.SIGNATURE.VERIFICATION.KEYRING))

-OR-

RACDCERT ID(RACFADM) CONNECT(CERTAUTH LABEL('MyCompany Code Signing CA')
  RING(CODE.SIGNATURE.VERIFICATION.KEYRING))
```

- 
3. Create or modify the PROGRAM class profile that controls the signed program and specify the signature verification options.

The following examples specify that the load of program MYPROG14 should fail if the signature cannot be verified for any reason and that only failures should be logged.

**Examples:**

```
RDEFINE PROGRAM MYPROG14 ADDMEM('SYS1.TEST.LOADDLL'//NOPADCHK) UACC(READ)
  SIGVER(SIGREQUIRED(YES) FAILLOAD(ANYBAD) SIGAUDIT(ANYBAD))

-OR-

RALTER PROGRAM MYPROG14
  SIGVER(SIGREQUIRED(YES) FAILLOAD(ANYBAD) SIGAUDIT(ANYBAD))
```

If you want to delegate authority to perform this step to a user who does not have the SPECIAL attribute, see [“Delegating the authority for specifying signature verification options”](#) on page 337.

- 
4. Activate your profile changes in the PROGRAM class.

**Example:**

```
SETROPTS WHEN(PROGRAM) REFRESH
```

---

You have now enabled RACF to verify a signed program. If you specified the signature verification options shown in the example in Step “3” on page 343, the program will fail to load if RACF cannot verify the signature for any reason. If the program is part of a critical business application, be prepared to invoke a recovery procedure to minimize the business impact.

## IPL data signing for Validated Boot for z/OS

---

With Validated Boot for z/OS, your installation can ensure that its system IPL data is intact, untampered-with, and originates from a trusted build-time source. To use Validated Boot for z/OS, you must configure RACF to enable IPL data signing, as described in this topic.

This topic contains the following subtopics:

- [“Overview of enabling your system for signed IPL data” on page 344](#)
- [“Certificate requirements for signing IPL data” on page 344](#)
- [“Defining the IRR.PROGRAM.V2.SIGNING profile” on page 345](#)
- [“Enabling IPL data signing for Validated Boot for z/OS” on page 347](#)

### Overview of enabling your system for signed IPL data

In RACF, the callable service **R\_PgmSignVer (IRRSPS00)** is used for enabling users to digitally sign programs. This service is also used for signing IPL data (IPL text and load modules) for Validated Boot for z/OS operations. When used for Validated Boot operations, **R\_PgmSignVer** produces signing output for IPL data in the required formats, such as Public Key Cryptographic Standards #7 (PKCS#7) Distinguished Encoding Rules (DER) format, or a simplified structure suitable for signature verification with the KDSA instruction.

In Validated Boot for z/OS, the following z/OS services use **R\_PgmSignVer** to sign IPL data:

- Device Support Facility (ICKDSF), which signs IPL text and requests the signature output in the PKCS#7 format.
- z/OS Binder, which signs load modules and requests the signature output in the simplified KDSA structure format.

For more information about the **R\_PgmSignVer** service, see [z/OS Security Server RACF Callable Services](#).

### Certificate requirements for signing IPL data

To enable your system for signing IPL data, you must create the required certificates for RACF.

This topic contains the following subtopics:

- [“Required certificates” on page 344](#)
- [“Supported signature algorithms” on page 345](#)
- [“Required certificate extensions” on page 345](#)

For examples of using RACDCERT GENCERT command to create certificates that meet these requirements, see [“Steps for using a RACF-generated signing certificate stored in a key ring” on page 348](#). Otherwise, contact your external certificate authority (CA) and see [“Steps for using an external signing certificate stored in a key ring” on page 350](#).

For details about using the RACDCERT GENCERT command, see [z/OS Security Server RACF Command Language Reference](#).

### Required certificates

Your installation must supply a RACF key ring or an ICSF token that contains the following certificates:

- The default certificate in the RACF key ring or ICSF Token must have a private key and a key type of NISTEC with a 521-bit key size. This object is referred to as the *code-signing certificate*. The code-signing certificate can be self-signed or signed by a CA certificate.
- If the code-signing certificate is not a self-signed certificate, add the chain of CA certificates that signed the code-signing certificate up to the root CA. The R\_PgmSignVer service supports a maximum of ten CA certificates in the key ring or ICSF token.



## Supported signature algorithms

The code-signing certificate and each CA certificate in the code-signing certificate chain must be signed by using one of the following signature algorithms:

- sha256RSA
- sha224RSA
- sha384RSA
- sha512RSA
- sha256RSAPSS
- sha224RSAPSS
- sha384RSAPSS
- sha512RSAPSS
- sha256ECDSA
- sha224ECDSA
- sha384ECDSA
- sha512ECDSA

## Required certificate extensions

The certificates require extensions, as follows:

- The code-signing certificate must have the KeyUsage extension with at least the digitalSignature indicator enabled.
- The code-signing certificate must have the SubjectKeyIdentifier extension. Also, the keyIdentifier of the extension must be a 160-bit SHA-1 hash of the subject public key of the code-signing certificate.
- Each CA certificate in the code-signing certificate chain must have the BasicConstraints extension with the cA indicator enabled, or must not have a BasicConstraints extension.

## Defining the IRR.PROGRAM.V2.SIGNING profile

This topic describes the IRR.PROGRAM.V2.SIGNING profile, which is used for specifying the RACF key ring or ICSF token and the hash algorithm for Validated Boot for z/OS.

You must define APPLDATA information in one or more discrete profiles in the FACILITY class to specify the following information:

- Name of the z/OS signing key ring or token.
- Hash algorithm (or message digestion algorithm) that is to be used for signing IPL data. If you omit this value, SHA512 is used by default.

## Format of the profile name

The format of the IRR.PROGRAM.V2.SIGNING profile name is based on how you choose to assign signing key rings or tokens to users who are authorized to sign. The first four qualifiers of the profile name must be IRR.PROGRAM.V2.SIGNING.

The rest of the profile name reflects the available options for assigning key rings or tokens to signers. You can optionally append one or two more qualifiers to the profile name, as shown in the following list. RACF checks the profiles in the order that is listed and uses the first profile found that matches as follows.

### **IRR.PROGRAM.V2.SIGNING.groupid.userid**

This profile assigns a key ring for the specified user when the user's current connect group is the specified group ID.

### **IRR.PROGRAM.V2.SIGNING.userid**

This profile assigns a key ring for the specified user ID.

### **IRR.PROGRAM.V2.SIGNING.groupid**

This profile assigns a key ring for the user IDs whose current connect group matches the group ID specified.

### **IRR.PROGRAM.V2.SIGNING**

This profile assigns a key ring to be used by all users in the RACF database for Validated Boot for z/OS signing.

**Rule:** No generic characters (\*) are allowed in the name of a IRR.PROGRAM.V2.SIGNING[.groupid][.userid] profile.

**Note:** No access check is performed on the IRR.PROGRAM.V2.SIGNING[.groupid][.userid] profile.

## **Format of the APPLDATA value**

The format of the APPLDATA value in the IRR.PROGRAM.V2.SIGNING profile is as follows:

```
[hash-algorithm] [owning-userid]/key-ring-name
```

or

```
[hash-algorithm] *TOKEN*/token-name
```

The variables of the APPLDATA value are defined as follows:

### **hash-algorithm**

Specifies the message digestion algorithm to be used for Validated Boot for z/OS signing. The supported value is SHA512; no other values are supported. If you omit this value, RACF uses SHA512 by default. The hash-algorithm value is overridden if it is also specified in the R\_PgmSignVer function call parameter list.

### **owning-userid**

Specifies the user ID that owns the Validated Boot for z/OS signing key ring. If you omit this value, RACF uses the user ID of the user who invokes the R\_PgmSignVer signing service.

### **/key-ring-name**

Specifies the fully qualified name of the Validated Boot for z/OS signing key ring. This value must be preceded by the forward slash (/).

### **/token-name**

Specifies the Validated Boot for z/OS token name to be used for signing. This value must be preceded by \*TOKEN\* and the forward slash (/).

### **Rules:**

- The only space character that is allowed in the APPLDATA value is the single space after the *hash-algorithm* value. If *hash-algorithm* is omitted, no space is allowed in the APPLDATA value.
- No extraneous characters are allowed in the APPLDATA value.

### **Notes:**

- RACF does not check the format of the APPLDATA value when you define a IRR.PROGRAM.V2.SIGNING profile.
- The RDEFINE command that is used to create this profile converts the key ring or token name that is defined in the APPLDATA to all uppercase characters. Therefore, do not use mixed case characters when you define the key ring name or token name.

## **Examples of profile names**

The format of the IRR.PROGRAM.V2.SIGNING profile name is based on how you choose to assign the signing key rings or tokens to users who are authorized to sign.

- The following profile defines that user ID ZSIGNER, when this user's current connect group ID is BUILD, uses the VB\_SIGNING\_KEYRING key ring defined for user ID BUILDID for Validated Boot for z/OS signing operations.

```
RDEFINE FACILITY IRR.PROGRAM.V2.SIGNING.BUILD.ZSIGNER
APPLDATA('SHA512 BUILDID/VB_SIGNING_KEYRING')
```

- The following profile defines that user ID ZSIGNER uses the VB\_SIGNING\_KEYRING key ring that is defined to user ID ZSIGNER for Validated Boot for z/OS signing operations, unless a more specific profile applies.

```
RDEFINE FACILITY IRR.PROGRAM.V2.SIGNING.ZSIGNER
APPLDATA('SHA512 ZSIGNER/VB_SIGNING_KEYRING')
```

- The following profile defines that users whose current connect group is PROD will use the VB\_SIGNING\_KEYRING key ring that is defined to user ID PRODIG for Validated Boot for z/OS signing operations, unless a more specific profile applies.

```
RDEFINE FACILITY IRR.PROGRAM.V2.SIGNING.PROD
APPLDATA('SHA512 PRODIG/VB_SIGNING_KEYRING')
```

- The following profile defines that users will use the VB\_SIGNING\_KEYRING key ring that is defined to user ID RACFADM for Validated Boot for z/OS signing operations, unless a more specific profile applies.

```
RDEFINE FACILITY IRR.PROGRAM.V2.SIGNING
APPLDATA('SHA512 RACFADM/VB_SIGNING_KEYRING')
```

- The following profile defines that users will use the RACFADM.VBTOKEN ICSF token for Validated Boot for z/OS signing operations, unless a more specific profile applies.

```
RDEFINE FACILITY IRR.PROGRAM.V2.SIGNING
APPLDATA('SHA512 *TOKEN*/RACFADM.VBTOKEN')
```

## Enabling IPL data signing for Validated Boot for z/OS

It is possible to perform IPL data signing by using RACF-generated or externally supplied certificates, and storing the signing key in either a RACF key ring or ICSF token, as needed. This topic provides instructions for each of these scenarios.

This topic describes the following scenarios for signing IPL data. Choose the one that best matches your particular use case.

Table 27. Scenarios for signing IPL data	
Scenario	See the following topic...
You plan to use: <ul style="list-style-type: none"> <li>• RACF key ring</li> <li>• Certificate generated by using RACF</li> <li>• Signing key stored in the PKDS data set.</li> </ul>	<a href="#">“Steps for using a RACF-generated signing certificate stored in a key ring” on page 348</a>
You plan to use: <ul style="list-style-type: none"> <li>• RACF key ring</li> <li>• Certificate imported from an external source</li> <li>• Signing key stored in RACF.</li> </ul>	<a href="#">“Steps for using an external signing certificate stored in a key ring” on page 350</a>
You plan to use: <ul style="list-style-type: none"> <li>• ICSF token</li> <li>• Certificate generated by using RACF</li> <li>• Signing key stored in an ICSF token in the TKDS data set.</li> </ul>	<a href="#">“Steps for using a RACF-generated signing certificate stored in an ICSF token” on page 352</a>

## Steps for using a RACF-generated signing certificate stored in a key ring

**Note:** This procedure creates a CA certificate and then a code-signing certificate signed by that CA. However, it is acceptable to create just a self-signed code-signing certificate with the required attributes.

To enable a group to digitally sign IPL data by using a signing certificate that you create with RACF, perform the following steps.

1. If it does not already exist, create a certificate-authority (CA) certificate that you can use to issue a signing certificate for users who will perform Validated Boot for z/OS signing.

**Note:** It is recommended, but not required, to create the public and private keys for the CA certificate in the ICSF PKDS. Doing so provides the strongest level of security of the private key. The RSA(PKDS) keyword causes public and private keys to be stored in the ICSF PKDS.

### Example:

```
RACDCERT CERTAUTH GENCERT
SUBJECTSDN(OU('MyCompany Validated Boot Signing CA') O('MyCompany') C('US'))
SIZE(4096) RSA(PKDS) WITHLABEL('Validated Boot Signing CA')
```

2. Create a code-signing certificate for users who will perform Validated Boot for z/OS signing. In the following example, the code signing certificate is owned by user ZSIGNER and signed by the CA certificate.

**Note:** It is recommended, but not required, to create the public and private keys for the code-signing certificate in the ICSF PKDS. Doing so provides the strongest level of security of the private key. The NISTECC(PKDS(VBSIGNINGKEY)) keyword causes public and private keys to be stored in the ICSF PKDS with the VBSIGNINGKEY label. This makes it easier to share the signing key with the members of the group.

### Example:

```
RACDCERT ID(ZSIGNER) GENCERT SUBJECTSDN(CN('Validated Boot Signing Cert')
O('MyCompany') C('US')) NISTECC(PKDS(VBSIGNINGKEY)) SIZE(521)
WITHLABEL('Validated Boot Signing Cert') SIGNWITH(CERTAUTH LABEL('Validated Boot Signing
CA'))
KEYUSAGE(HANDSHAKE)
```

3. Create a RACF key ring to hold the certificates that you created in Steps “1” on page 348 and “2” on page 348. In the following example, the key ring is owned by user ZSIGNER.

**Rule:** Specify the key ring name in all uppercase characters so that it matches the key ring that is specified in the IRR.PROGRAM.V2.SIGNING profile APPLDATA operand, which is converted to uppercase when created with the RDEFINE command.

### Example:

```
RACDCERT ID(ZSIGNER) ADDRING(VB_SIGNING_KEYRING)
```

4. Connect both certificates that you created in Steps “1” on page 348 and “2” on page 348 to the key ring you created in Step “3” on page 348.

**Rule:** The signing certificate must be the default certificate in the ring.

### Example:

```
RACDCERT ID(ZSIGNER) CONNECT(CERTAUTH LABEL('Validated Boot Signing CA')
RING(VB_SIGNING_KEYRING))
RACDCERT ID(ZSIGNER) CONNECT(ID(ZSIGNER) LABEL('Validated Boot Signing Cert') DEFAULT
RING(VB_SIGNING_KEYRING))
```

- Define the RDATA LIB class profile that covers the key ring that was created in the previous steps. Permit the members of the BUILDGRP group to sign the code by using the key ring and private key of the code-signing certificate.

**Example:**

```
RDEFINE RDATA LIB ZSIGNER.VB_SIGNING_KEYRING.LST UACC(NONE)
PERMIT ZSIGNER.VB_SIGNING_KEYRING.LST CLASS(RDATA LIB) ID(BUILDGRP) ACCESS(UPDATE)
```

In this example, the owner of the key ring is user ZSIGNER, and the key ring name is VB\_SIGNING\_KEYRING. Thus, the RDATA LIB profile name that covers that resource is: ZSIGNER.VB\_SIGNING\_KEYRING.LST (<ring owner>.<ringname>.LST).

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

- 
- Define the signing profile. In the following example, a signing profile is defined for the group of authorized signers called BUILDGRP.

**Example:**

```
RDEF FACILITY IRR.PROGRAM.V2.SIGNING.BUILDGRP APPLDATA('SHA512 ZSIGNER/VB_SIGNING_KEYRING')
```

For more information about the profile structure, see [“Defining the IRR.PROGRAM.V2.SIGNING profile” on page 345](#).

- 
- Permit the users who are connected to the BUILDGRP group to perform the signing, using the signing key stored in the PKDS. In this example, assume that a CSFDSG profile is already defined in the CSFSERV resource class.



**CAUTION:** IBM recommends that the CSFSERV and CSFKEYS classes be RACLISTed. However, be aware that a number of ICSF services are available when the CSFSERV or CSFKEYS classes are not active. Therefore, if you activate and RACLIST these classes, any product that currently uses the services that are protected by these classes can fail due to lack of authorization.

**Example:**

```
PERMIT CSFDSG CLASS(CSFSERV) ID(BUILDGRP) ACCESS(READ)
PERMIT VBSIGNINGKEY CLASS(CSFKEYS) ID(BUILDGRP) ACCESS(READ)
```

If the CSFSERV and CSFKEYS classes are not already active, activate and RACLIST them.

```
SETROPTS CLASSACT(CSFSERV CSFKEYS) RACLIST(CSFSERV CSFKEYS)
```

If the CSFSERV and CSFKEYS classes are already active and RACLISTed, refresh them.

```
SETROPTS RACLIST(CSFSERV CSFKEYS) REFRESH
```

---

You have now enabled a RACF group to digitally sign IPL data by using a signing certificate that you created with RACF.

## Steps for using an external signing certificate stored in a key ring

**Note:** This procedure creates a CA certificate and a code-signing certificate that is signed by that CA. However, it is acceptable to create just a self-signed code-signing certificate with the required attributes.

**Before you begin:** Obtain or locate the root certificate-authority (CA) certificate of an external CA — and the intermediate CAs, if needed — and store them in a cataloged, variable-blocked (VB) MVS data set.

To enable a group to digitally sign IPL data by using a signing certificate that you obtain from an external certificate-authority (CA), perform the following steps.

1. Add the root CA certificate of the external CA to RACF, specifying the data set in which it is stored.

### Example:

```
RACDCERT CERTAUTH ADD(<data set containing the root CA cert>) WITHLABEL('External VB root CA')
```

If you want to use an intermediate CA, enter this command to add it. To add multiple intermediate CAs, repeat this command for each of them with the appropriate labels.

### Example:

```
RACDCERT CERTAUTH ADD(<data set containing the intermediate CA cert>) WITHLABEL('External VB intermediate CA')
```

2. Add the signing certificate to RACF. In the following example, the signing certificate is owned by user ZSIGNER and the signing certificate is contained in a PKCS #12 package that is stored in a data set.

### Example:

```
RACDCERT ID(ZSIGNER) ADD(<data set containing the signing cert in pkcs12 package>) WITHLABEL('Validated Boot Signing Cert') PASSWORD('<pw>')
```

3. Create a RACF key ring to hold the certificates you added in Steps “1” on page 350 and “2” on page 350. In the following example, the key ring is owned by user ZSIGNER.

**Rule:** Specify the key ring name in all uppercase characters so that it matches the key ring that is specified in the IRR.PROGRAM.V2.SIGNING profile APPLDATA operand. This value is converted to uppercase when the profile is created with the RDEFINE command.

### Example:

```
RACDCERT ID(ZSIGNER) ADDRING(VB_SIGNING_KEYRING)
```

4. Connect all of the certificates that you obtained in Steps “1” on page 350 and “2” on page 350 to the key ring that you created in Step “3” on page 350.

**Rule:** The signing certificate must be the default certificate in the ring.

### Example:

```
RACDCERT ID(ZSIGNER) CONNECT(CERTAUTH LABEL('External VB root CA')
RING(VB_SIGNING_KEYRING))

RACDCERT ID(ZSIGNER) CONNECT(CERTAUTH LABEL('External VB intermediate CA')
RING(VB_SIGNING_KEYRING))

RACDCERT ID(ZSIGNER) CONNECT(ID(ZSIGNER) LABEL('Validated Boot Signing Cert') DEFAULT
RING(VB_SIGNING_KEYRING))
```

- Define the RDATA LIB class profile that covers the key ring that was created in the previous steps. Permit the members of the BUILDGRP group to sign the code by using the key ring and private key of the code-signing certificate.

**Example:**

```
RDEFINE RDATA LIB ZSIGNER.VB_SIGNING_KEYRING.LST UACC(NONE)
PERMIT ZSIGNER.VB_SIGNING_KEYRING.LST CLASS(RDATA LIB) ID(BUILDGRP) ACCESS(UPDATE)
```

In this example, the owner of the key ring is user ZSIGNER, and the key ring name is VB\_SIGNING\_KEYRING. Thus, the RDATA LIB profile name that covers that resource is: ZSIGNER.VB\_SIGNING\_KEYRING.LST (<ring owner>.<ringname>.LST).

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

- 
- Define the signing profile. In the following example, a signing profile is defined for a group of authorized signers called BUILDGRP.

**Example:**

```
RDEF FACILITY IRR.PROGRAM.V2.SIGNING.BUILDGRP APPLDATA('SHA512 ZSIGNER/VB_SIGNING_KEYRING')
```

Refresh the FACILITY class.

```
SETR RACLIST(FACILITY) REFRESH
```

For more information about the profile structure, see [“Defining the IRR.PROGRAM.V2.SIGNING profile” on page 345](#).

- 
- Permit the users who are connected to the BUILDGRP group to perform the signing, using the signing key stored in the RACF database. In this example, assume that the CSF1TRC, CSF1PKS and CSF1TRD profiles are already defined in the CSFSERV resource class.



**CAUTION:** IBM recommends that the CSFSERV class be RACLISTed. However, be aware that a number of ICSF services are available when the CSFSERV class is not active. Therefore, if you activate and RACLIST this class, any product that currently uses the services that are protected by this class can fail due to lack of authorization.

**Example:**

```
PERMIT CSF1TRC CLASS(CSFSERV) ID(BUILDGRP) ACCESS(READ)
PERMIT CSF1PKS CLASS(CSFSERV) ID(BUILDID) ACCESS(READ)
PERMIT CSF1TRD CLASS(CSFSERV) ID(BUILDID) ACCESS(READ)
```

If the CSFSERV class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(CSFSERV) RACLIST(CSFSERV)
```

If the CSFSERV class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(CSFSERV) REFRESH
```

---

You have now enabled a RACF group to digitally sign IPL data by using a signing certificate that you obtained from an external certificate-authority (CA).

## Steps for using a RACF-generated signing certificate stored in an ICSF token

**Note:** This example creates a CA certificate and then a code-signing certificate signed by that CA. However, it is acceptable to create just a self-signed code-signing certificate with the required attributes.

To enable a group to digitally sign IPL data by using a certificate that you create with RACF and store in an ICSF token, perform the following steps.

1. Create an ICSF token with the RACDCERT command. The token will contain the certificates and keys that you create in Steps “2” on page 352 and “3” on page 352.

**Rule:** Specify the token name to match the token name that is specified in the IRR.PROGRAM.V2.SIGNING profile APPLDATA operand.

### Example:

```
RACDCERT ADDTOKEN(ZSIGNER.VBTOKEN)
```

2. If it does not already exist, create a certificate-authority (CA) certificate that you can use to issue a signing certificate for users who will perform Validated Boot for z/OS signing.

### Example:

```
RACDCERT CERTAUTH GENCERT
SUBJECTSDN(OU('MyCompany Validated Boot Signing CA') O('MyCompany') C('US'))
SIZE(4096) RSA WITHLABEL('Validated Boot Signing CA')
```

3. Create a code-signing certificate for users who will perform Validated Boot for z/OS signing. In the following example, the code signing certificate is owned by user ZSIGNER and signed by the CA certificate.

### Example:

```
RACDCERT ID(ZSIGNER) GENCERT SUBJECTSDN(CN('Validated Boot Signing Cert')
O('MyCompany') C('US')) NISTECC(TOKEN(ZSIGNER.VBTOKEN)) SIZE(521) WITHLABEL('Validated Boot
Signing Cert')
SIGNWITH(CERTAUTH LABEL('Validated Boot Signing CA')) KEYUSAGE(HANDSHAKE)
```

4. Bind the code-signing certificate and its CA certificate to the ICSF token.

### Example:

```
RACDCERT BIND(CERTAUTH LABEL('Validated Boot Signing CA') TOKEN(ZSIGNER.VBTOKEN))
RACDCERT BIND(ID(ZSIGNER) LABEL('Validated Boot Signing Cert') DEFAULT USAGE(PERSONAL)
TOKEN(ZSIGNER.VBTOKEN))
```

5. Define the signing profile. In the following example, a signing profile is defined for a group of authorized signers called BUILDGRP.

### Example:

```
RDEF FACILITY IRR.PROGRAM.V2.SIGNING.BUILDGRP APPLDATA('SHA512 *TOKEN*/ZSIGNER.VBTOKEN')
```

For more information about the profile structure, see [“Defining the IRR.PROGRAM.V2.SIGNING profile” on page 345](#).

6. Permit the users who are connected to the BUILDGRP group to access the signing key. The key is stored in the ICSF token and is protected by the CSFSERV and CRYPTOZ classes. In this example, assume that the profiles for CSFDSG, CSF1TRL, and CSF1GAV are already defined in the CSFSERV resource class.





**CAUTION:** IBM recommends that the CSFSERV and CRYPTOZ classes be RACLISTed. However, be aware that some ICSF services are available when the CSFSERV or CRYPTOZ class is not active. Therefore, if you activate and RACLIST these classes, any product that currently uses the services that are protected by these classes can fail due to lack of authorization.

**Example:**

```
PERMIT CSFDSG CLASS(CSFSERV) ID(BUILDGRP) ACCESS(READ)
PERMIT CSF1TRL CLASS(CSFSERV) ID(BUILDGRP) ACCESS(READ)
PERMIT CSF1GAV CLASS(CSFSERV) ID(BUILDGRP) ACCESS(READ)
PERMIT USER.ZSIGNER.VBTOKEN CLASS(CRYPTOZ) ID(BUILDGRP) ACCESS(READ)
```

- If the CSFSERV and CRYPTOZ classes are not already active, activate and RACLIST them.

```
SETROPTS CLASSACT(CSFSERV CRYPTOZ) RACLIST(CSFSERV CRYPTOZ)
```

- If the CSFSERV and CRYPTOZ classes are already active and RACLISTed, refresh them.

```
SETROPTS RACLIST(CSFSERV CRYPTOZ) REFRESH
```

---

You have now enabled a group to digitally sign IPL data by using a signing certificate that you created with RACF and stored in an ICSF token.



---

## Chapter 15. Operating considerations

This topic discusses operating RACF on your system.

---

### Coordinating profile updates

You should plan to update profiles so that they remain consistent with other profiles on the database while making sure that the updating process does not interfere with other jobs running in the system.

When RACF is enabled for sysplex communication, members of a data sharing group are notified to create, refresh, or delete their in-storage profiles. The command is coordinated to ensure that all systems begin to use the refreshed profiles simultaneously. See *z/OS Security Server RACF Command Language Reference* for more information on the operands you need for this.

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

**Note:** If a user is logged on, and you update the user's attributes in the RACF database using ALTUSER or CONNECT, some changes might not take effect until the next time the user enters the system. However, a LISTUSER or LISTGRP command issued immediately after the change shows the new values.

Some of the changes that are delayed until the user logs on again are the SPECIAL, OPERATIONS, AUDITOR, and ROAUDIT attributes and the list of connected groups examined by RACROUTE REQUEST=FASTAUTH.

#### Example:

In this example, the security administrator inadvertently creates a situation where a profile exists, but it does not have an owner. The security administrator issues DELUSER to delete a user from RACF. At the same time, the other user (who has the ADSP attribute and is logged on) creates a permanent user data set, which automatically creates a discrete data set profile.

The DELUSER command performs the following operations on the RACF database:

1. Locates the user profile in the RACF database.
2. Locates any user data set profiles.
3. Ensures that the user does not have any user data sets whose high-level qualifier is his user ID. (RACF cannot delete the user profile until all of his user data sets are deleted.)
4. Deletes the user profile.
5. Updates the group profile to remove the user as an eligible member of the group.

As a result of the ADSP attribute, RACF performs one operation on the RACF database: it adds a data set profile for the permanent user data set.

In this example, if the user adds the new data set profile between Steps 2 and 3 of the DELUSER command processing, RACF adds a user data set profile to the RACF database. However, RACF has already deleted the user who owns the profile. This creates an ownerless profile.

To prevent the creation of ownerless profiles, do not delete a user who is logged on. Instead, make sure the user is logged off and cannot log on again. If necessary, have the operator force the user off the system first. Then follow the steps described in [“Summary of steps for deleting users” on page 77](#).

## RACF commands for flushing a VLF cache

For installations using the IRRACEE class to store security environments with the Virtual Lookaside Facility (VLF), administrators should be aware that issuing certain RACF commands can delete one or more such objects.

Examples of commands that delete the stored security environment for a user are DELUSER, PASSWORD, and ALTUSER.

You can determine the fields that cause VLF purging on ACEE by referring to the RACF database templates in [z/OS Security Server RACF Macros and Interfaces](#). A security-sensitive field has bit 0 of flag 2 turned on. Changes to such a field trigger VLF purging.

In an installation where no RACF database sharing occurs, issuing commands that deal with certain general resource classes or profiles can delete *all* stored security environments. Examples of this include activating, deactivating, or issuing SETROPTS NORACLIST(*classname*) or SETROPTS RACLIST(*classname*) REFRESH for these classes:

- APPCPORT
- APPL
- CONSOLE
- FACILITY (only when SETROPTS MLS is in effect)
- GTERMINL
- JESINPUT
- MFADEF
- SECLABEL
- SERVAUTH
- TERMINAL

For participants sharing a RACF database, deleting one or more stored security environments on one system causes *all* stored security environments to be deleted by the other participants. Thus, the administration of user profiles in a shared environment with a performance-oriented participant should be administered from that system, if possible.

In all cases, any deleted security environment can be restored on demand through actions such as legitimate logging on or job submission.

For information on using VLF for mapping z/OS UNIX user identifiers (UIDs) and z/OS UNIX group identifiers (GIDs) in the UNIXMAP class, see [“Using the UNIXMAP class and Virtual Lookaside Facility \(VLF\)”](#) on page 524.

For more information on VLF, see [z/OS MVS Planning: Operations](#), [z/OS MVS Initialization and Tuning Guide](#), and [z/OS MVS Initialization and Tuning Reference](#).

## Getting started with RACF (after first installing RACF)

After you initialize your system with RACF active for the first time, you can quickly achieve system security. During RACF installation, the following profile is created in the RACF database:

Group	Superior group	Owner	Connected users (group authority)
SYS1	—	IBMUSER	IBMUSER (JOIN)

There is only one user:

User	Default group (group authority)	Attributes	Connected groups (group authority)
IBMUSER	SYS1 (JOIN)	SPECIAL and OPERATIONS	—

And there are four security labels:

Name	Owner	UACC
SYSHIGH	IBMUSER	NONE
SYSLOW	IBMUSER	NONE
SYSNONE	IBMUSER	NONE
SYSMULTI	IBMUSER	NONE

There is a set of supplied certificate authority (CA) certificates. They are not used to authenticate CAs until you decide to use them. For more information, see [“Supplied digital certificates”](#) on page 589.

**Restrictions:** The basic set of profiles described in this topic is supplied with RACF. These profiles cannot be defined by your installation and should not be deleted. They must exist at initialization time or RACF initialization will automatically add them.

## Logging on as IBMUSER and checking initial conditions

IBMUSER is the first user ID that the security administrator can use. This user ID has the SPECIAL attribute, which allows IBMUSER to issue most of the RACF commands (except for those reserved for users with the AUDITOR attribute) and the OPERATIONS attribute, which allows IBMUSER to access many RACF-protected resources.

When you enter the system for the first time with the IBMUSER user ID, you must change the initial password, SYS1, to a new password. A new password prevents any other user from entering the system as IBMUSER.

Log on as IBMUSER:

```
LOGON IBMUSER
```

After entering IBMUSER's old password (SYS1) and defining a new password, list the system-wide RACF options that are in effect:

```
SETROPTS LIST
```

Read through this list to familiarize yourself with the options that are in effect. For an explanation of what some of the options are and what they mean, see [“Using the SETROPTS command”](#) on page 105.

**Note:** Not all options are displayed at this point, because IBMUSER does not have the AUDITOR or ROAUDIT attribute. If you want to see the status of these options, grant IBMUSER the ROAUDIT attribute, log off, and log on again. To see all of the options, issue SETROPTS LIST again.

For a complete listing of all of the options that are available, see [z/OS Security Server RACF Command Language Reference](#).

**Important:** The option for the TERMINAL resource class should be specified as READ. Do not change it to NONE unless you have defined your terminals to RACF and authorized the appropriate users and groups to access them. If you specify TERMINAL(NONE) without first defining your terminals to RACF, you cannot access your terminals and, consequently, you will be locked out of your system.

### Defining administrator user IDs for your own use

Define a new user (for example, RACFADM) to RACF for your own use. This user should have at least the SPECIAL and OPERATIONS attributes. If you are also the system-wide auditor, you should also give this user ID the AUDITOR attribute. Depending on the attributes you select, enter one of the following commands.

- `ADDUSER RACFADM SPECIAL OPERATIONS`
- `ADDUSER RACFADM SPECIAL OPERATIONS AUDITOR`

**Note:** You should also plan this user's SYS1.UADS entry (see [z/OS TSO/E Customization](#)) or TSO segment. For example, if an administrator (including IBMUSER) is to work with data set profiles (using the ADDSD or ALTDSD commands), you should ensure that the user can have volumes mounted during a TSO session. You can do this by giving the user the MOUNT attribute in the SYS1.UADS entry, or by giving the user READ access authority to the MOUNT profile in the TSOAUTH class. See [“Protecting TSO resources”](#) on page 507.

### Defining at least one user ID to be used for emergencies only

To handle emergency situations that could arise, such as if RACF becomes inoperative or all SPECIAL users become revoked, you should consider setting up at least one, and preferably two, "emergency" user IDs. These user IDs should *never* be used except in extreme cases, under management supervision. They should have no TSO segment in their RACF user profiles, and their entries in the SYS1.UADS data set should give them all attributes.

### Logging on as RACFADM, checking groups and users, and revoking IBMUSER

Log on as RACFADM and use the default password, SYS1 in this case (IBMUSER's default group).

You receive a message stating that your password expired. Immediately change the password, SYS1, to a new password.

First, list all users to ensure that only RACFADM and IBMUSER are defined to RACF, and that they have the proper attributes.

```
LISTUSER *
```

Then, list all of the groups that are defined to RACF:

```
LISTGRP *
```

Connect RACFADM to each group and make RACFADM the owner of the group:

```
CONNECT RACFADM GROUP(SYS1) AUTH(JOIN)
ALTGROUP SYS1 OWNER(RACFADM)
```

Then, revoke the IBMUSER user ID so that another user cannot use it:

```
ALTUSER IBMUSER REVOKE
```

**Note:** You cannot delete the IBMUSER user profile.

Define another user to RACF (for example, user ID RACFAD2), to act as your assistant. Make the new user's default group SYS1, and give this assistant the SPECIAL and OPERATIONS user attributes.

```
ADDUSER RACFAD2 DFLTGRP(SYS1) AUTH(JOIN) SPECIAL OPERATIONS
```

### Defining the groups needed for the first users

At this point you should consider creating the groups that you need.

The following commands show an example of adding four groups. Three are departmental groups (GROUP1, GROUP2, and GROUP3), and GROUP2 and GROUP3 are owned by GROUP1 so that certain authorities can be propagated. The fourth group (DATAMGT) has global pack maintenance responsibility.

```
ADDGROUP (GROUP1 DATAMGT)
ADDGROUP (GROUP2 GROUP3) OWNER(GROUP1) SUPGROUP(GROUP1)
```

## Defining a system-wide auditor

Define a user (for example, AUDCCC) who has system-wide auditing responsibilities and privileges.

```
ADDUSER AUDCCC AUDITOR
```

## Defining a system-wide read-only auditor

Define a user (for example, AUDMON) who has system-wide responsibility and privilege to monitor and view system audit information using the existing system auditing controls. Such a user would be useful in assuring system integrity without being able to alter the system auditing controls.

```
ADDUSER AUDMON ROAUDIT
```

## Defining users and groups

You now add a user (D03DIK) to GROUP3 with authority to protect group data sets.

```
ADDUSER D03DIK OWNER(GROUP3) AUTH(CREATE) DFLTGRP(GROUP3)
```

**Note:** For more information, see [“Summary of steps for defining users” on page 75](#).

## Defining group administrators, group auditors, and data managers

For each group, define a group administrator with the group-SPECIAL attribute. Only the administrator for GROUP1 has the authority to define new users in that group. Each of the other administrators has authority over the resources owned by his or her group, as well as the resources owned by users who are owned by his or her group.

```
ADDUSER D01RHG DFLTGRP(GROUP1) CLAUTH(USER) DATA('GROUP1 ADM')
CONNECT D01RHG GROUP(GROUP1) AUTH(JOIN) SPECIAL

ADDUSER D02JMP DFLTGRP(GROUP2) DATA('GROUP2 ADM')
CONNECT D02JMP GROUP(GROUP2) AUTH(CREATE) SPECIAL

ADDUSER D03ABL DFLTGRP(GROUP3) DATA('GROUP3 ADM')
CONNECT D03ABL GROUP(GROUP3) AUTH(CREATE) SPECIAL
```

For groups GROUP1, GROUP2, and GROUP3, define a group-auditor. Connect the user to GROUP1 and give the user the group-AUDITOR attribute. Because GROUP2 and GROUP3 are owned by GROUP1, the user has auditor authority over the resources and users belonging to those groups, as well as to GROUP1. The user does not have auditor authority in any other group.

```
ADDUSER D01GPB DFLTGRP(GROUP1) DATA('AUDITOR G1 G2 G3')
CONNECT D01GPB GROUP(GROUP1) AUDITOR
```

The administrator for the data management group, the data manager, is able to define DASD volumes to RACF in order to perform dump, restore, and data cleanup operations.

```
ADDUSER DMGJFS DFLTGRP(DATAMGT) AUTH(JOIN) CLAUTH(USER DASDVOL)
DATA('DATA MGT ADM')
```

## Operating considerations

Because of his or her duties, the data manager is connected to SYS1, allowing the manager to access data sets with SYS1 in their access list and to define SYS1 data set profiles to RACF. The data manager has the group-SPECIAL attribute in group SYS1.

```
CONNECT DMGJFS GROUP(SYS1) AUTH(CREATE) UACC(READ) SPECIAL
```

At the end of the session, the defined group structure is:

Group	Superior group	Owner	Connected users (group authority)
SYS1	—	RACFADM	IBMUSER (JOIN) RACFADM (JOIN) RACFAD2 (JOIN) DMGJFS (CREATE)
GROUP1	SYS1	RACFADM	D01RHG (JOIN) D01GPB (USE)
GROUP2	SYS1	GROUP1	D02JMP (CREATE)
GROUP3	SYS1	GROUP1	D03ABL (CREATE)
DATAMGT	SYS1	RACFADM	DMGJFS (JOIN)

The defined users are:

User	Default group (group authority)	Attributes	Connected groups (group authority)
IBMUSER	SYS1 (JOIN)	SPECIAL, OPERATIONS, REVOKE	—
RACFADM	SYS1 (JOIN)	SPECIAL, AUDITOR, OPERATIONS	—
RACFAD2	SYS1 (JOIN)	SPECIAL, OPERATIONS	—
DMGJFS	DATAMGT (JOIN), SYS1(CREATE)	CLAUTH(USER DASDVOL), SPECIAL	SYS1(CREATE)
D01RHG	GROUP1 (JOIN)	CLAUTH(USER), group-SPECIAL	—
D02JMP	GROUP2 (USE)	group-SPECIAL	—
D03ABL	GROUP3 (CREATE)	group-SPECIAL	—
D01GPB	GROUP1 (CREATE)	group-AUDITOR	—
D03DIK	GROUP3 (CREATE)	—	—
AUDCCC	SYS1 (USE)	AUDITOR	—
AUDMON	SYS1 (USE)	ROAUDIT	—

## Protecting system data sets

Create data set profiles to protect your system data sets. These should include the data sets described in Table 52 on page 711, as well as other data sets that your installation considers sensitive. The following are some candidates:

- General installation libraries
  - PROCLIB
  - TSO help and CLISTs
  - Compiler libraries
  - SORTLIB
- System control libraries
  - Nucleus, SVCLIB, LPALIB
  - Spool and paging data sets



- APF-authorized libraries
- Master catalog
- DLIB data
- SYS1.UADS
- PARMLIB
- Sensitive data
  - Corporate trade secrets
  - Research results
  - Employee data
  - Customer or client lists
- Production libraries
  - PROCLIBs
  - LOADLIBs
- Application development programs and data
  - Source
  - Load libraries
  - Documentation
- User data
  - JCL
  - Documentation
  - Source
  - Load modules

## Setting RACF options

Review [Chapter 6, “Specifying RACF options,”](#) on page 105 for the RACF options you want to set:

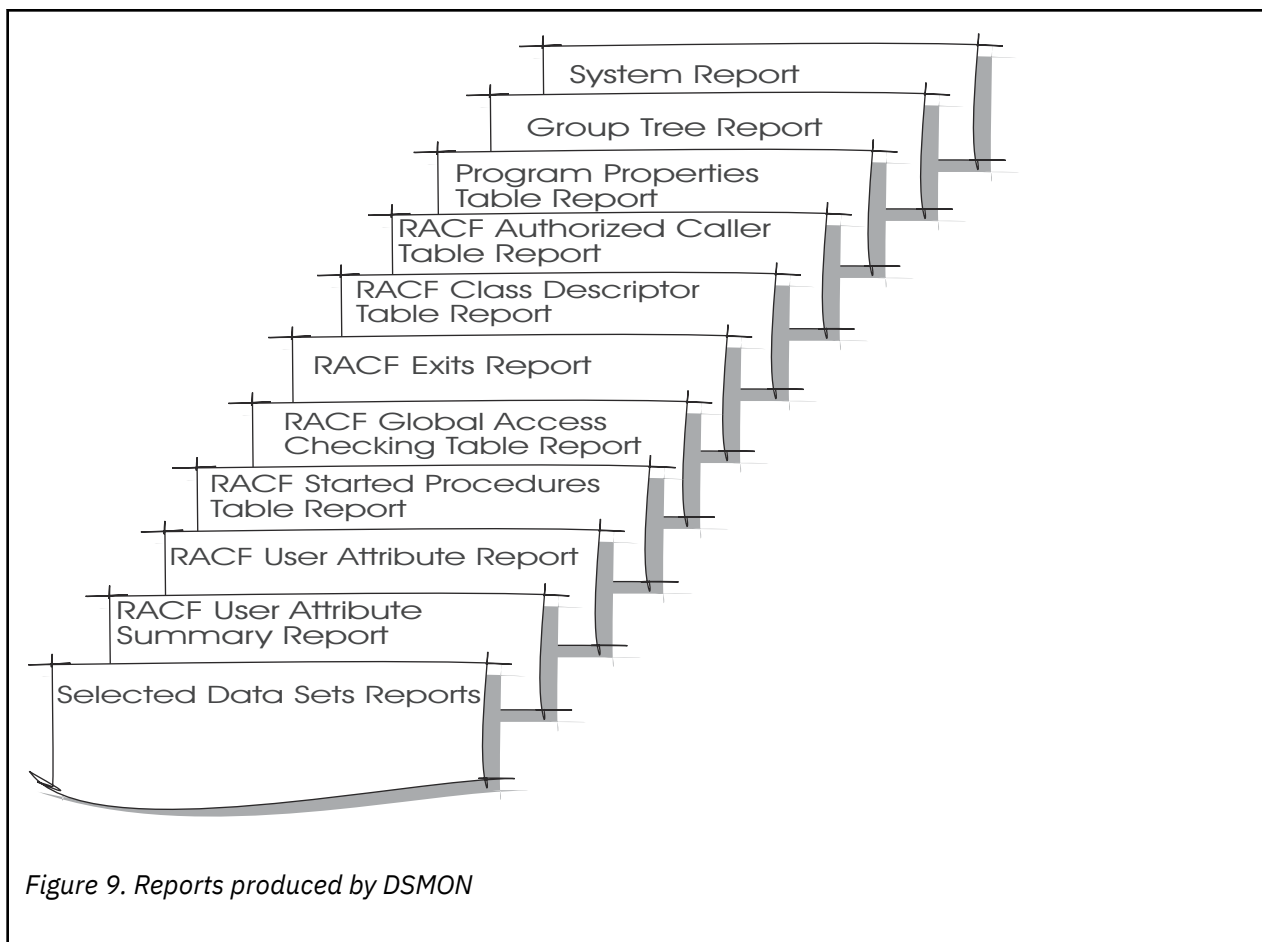
- Selecting options with the SETROPTS command (see [“Using the SETROPTS command”](#) on page 105)
- Encrypting RACF user passwords (see [“Specifying the encryption method for user passwords”](#) on page 139)
- Using started procedures (see [“Using started procedures”](#) on page 141)

## Using the data security monitor (DSMON)

---

The data security monitor (DSMON) produces a set of reports that provide information about the current status of the data security environment at your installation.

The reports DSMON can produce are:



These reports can help you (1) check the initial steps you took to establish system security, and (2) make additional security checks periodically.

A short description of each report follows. See [z/OS Security Server RACF Auditor's Guide](#) for more information on these reports and how to invoke the data security monitor.

### The system report

The system report contains information such as the identification and model of the processor complex, and the name, version, and release of the operating system. This report also specifies the RACF version and release number and whether RACF is active. If RACF is inactive, DSMON prints a message that tells you whether RACF was not activated at IPL or was deactivated by the RVARY command.

### The group tree report

This report lists, for each requested group, all of its subgroups, all of the subgroups' subgroups, and so on, as well as the owner of each group listed in the report, if the owner is not the superior group.

You can use the group tree report to examine the overall RACF group structure for your system. You can also determine the scope of the group for group-related user attributes (group-SPECIAL, group-OPERATIONS, and group-AUDITOR).

### The program properties table report

This report lists all of the programs in the MVS program properties table (PPT). The report also indicates, for each program, whether the program is authorized to bypass password protection and whether it runs in a system key.

You can use the program properties table report to verify that only those programs that the installation has authorized to bypass password protection are, in fact, able to do so. Such programs are normally communication and database control programs, and other system control programs.

You can also verify that only those programs that the installation has authorized are able to run in a system key.

### **The RACF authorized caller table report**

This report lists the names of all of the programs in the RACF authorized-caller table. The programs in this table are authorized to issue the RACROUTE REQUEST=VERIFY macro to perform user verification, or the RACROUTE REQUEST=LIST macro to load profiles into main storage.

You can use this report to verify that only those programs that are supposed to be authorized to modify an ACEE (accessor environment element) are able to issue the RACROUTE REQUEST=VERIFY. This verification is a particularly important security requirement because the ACEE contains a description of the current user. This description includes the user ID, the current connect group, the user attributes, and the group authorities. A program that is authorized to issue the RACROUTE REQUEST=VERIFY could alter the ACEE to simulate any user.

You can also use this report to verify that only those programs that are supposed to be authorized to access profiles are able to issue the RACROUTE REQUEST=LIST. Because profiles contain complete descriptions of the characteristics that are associated with RACF-defined entities, you must carefully control access to them.

### **The RACF class descriptor table report**

This report lists, for each general resource class in the class descriptor table (CDT), the class name, the default UACC, whether the class is active, whether auditing is being done, whether statistics are being kept, and whether OPERATIONS attribute users have access.

You can use the class descriptor table report to determine which classes (besides DATASET) are defined to RACF and active, and therefore can contain resources that RACF protects.

### **The RACF exits report**

This report lists the names of all of the installation-defined RACF exit routines and specifies the size of each exit routine module.

You can use the RACF exits report to verify that the only active exit routines are those that your installation has defined. The existence of any other exit routines might indicate a system security exposure, because RACF exit routines can be used to bypass RACF security checking. Similarly, if the length of an exit routine module differs from the length of the module when it was defined by your installation, the module might have unauthorized modifications.

### **The RACF global access checking table report**

This report lists, for each resource class in the global access table, all of the entry names and their associated resource access authorities.

Because global access checking allows anyone to access the resource at the associated access authority, you should verify that each entry has an appropriate level of access authority.

### **The RACF started procedures table reports**

RACF generates two reports about the started procedures table (ICHRIN03).

- If the STARTED class is active, the report uses the STARTED class profiles and contains the TRACE attribute. The trace uses module ICHDSM00.
- If the STARTED class is not active, the trace uses the installation replaceable load module, ICHRIN03.

The reports list the procedure name, the user ID and group name to be associated with the procedure, and whether the procedure is privileged or trusted.

You can use the report to determine which started procedures are defined to RACF, and which have the privileged attribute. If a started procedure is privileged or trusted, it bypasses all REQUEST=AUTH and REQUEST=FASTAUTH processing (unless the CSA or PRIVATE operand was specified on REQUEST=AUTH), including checks for security classification of users and data.

### **The selected user attribute report**

The selected user attribute report:

- Lists all RACF users with the SPECIAL, OPERATIONS, AUDITOR, ROAUDIT, or REVOKE attributes

- Specifies whether they possess these attributes on a system-wide (user) or group level
- Indicates whether they have any user ID associations

You can use this report to verify that only those users who need to be authorized to perform certain functions have been assigned the corresponding attribute.

### Selected user attribute summary report

The selected user attribute summary report shows the number of installation-defined users and totals for users with the SPECIAL, OPERATIONS, AUDITOR, ROAUDIT, and REVOKE attributes, at both the system and group level. You can use this report to verify that the number of users with each of these attributes, on either a system or group level, is the number that your installation wants. In particular, you should make sure that you have assigned the SPECIAL attribute (on a system level) to at least one user and the AUDITOR attribute (on a system level) to at least one user.

### The selected data sets report

This report lists the names of selected system data sets and, for each data set, specifies the criterion for selection, the serial number of the volume on which it resides, whether the data set is RACF-indicated or RACF-protected, and the universal access authority (UACC). If a data set meets more than one selection criterion, there is a separate entry in the report for each criterion. The selected data sets include system data sets, the MVS master catalog, user catalogs, the RACF primary and backup data sets, and user-specified data sets.

You can use the selected data sets report to determine which of these data sets are protected by RACF and which are not. You can also check whether the UACC associated with each of the data sets is compatible with your installation's resource access control requirements.

To reduce impact to system performance during the running of this report, you can limit or disable the listing of user catalogs. To do this, create a FACILITY class profile that protects the ICHDSM00.SYSCAT resource. When this resource is protected and the DSMON user does not have READ access to it, DSMON suppresses the listing of user catalogs and issues message ICH66134I, indicating the insufficient authority.

**Example:** To disable user catalog listing for the Selected Data Sets Report:

```
RDEFINE FACILITY ICHDSM00.SYSCAT UACC(NONE)
PERMIT ICHDSM00.SYSCAT CLASS(FACILITY) RESET
```

## JCL parameters related to RACF

This topic summarizes the JCL parameters that relate to RACF. For complete information, see [z/OS MVS JCL Reference](#).

- On the JOB statement:
  - USER parameter: Specify this parameter if user ID propagation is not used or if the user is submitting a job for another user.
  - PASSWORD parameter: Specify this parameter *only* when absolutely necessary. Specifying this parameter in JCL exposes the password to potential misuse.

**Note:** If a JOB statement contains a RACF password, you should establish procedures to ensure the security of the JOB statement. For example, ensure that printed job logs are kept secure.

JES suppresses the printing of passwords in output listings.

- GROUP parameter: Specify this parameter only if list-of-groups processing is *not* in effect and if the user wants the job to run with a group other than the user's default group.
- SECLABEL parameter: Specify this parameter if the job is to run with a security label other than the user's current security label.

If user ID propagation is used, all of these parameters are optional. Also, a TSO SUBMIT installation exit, TSO, or other procedures for handling batch jobs can place the RACF parameters on the JOB statement.

- On the DD statement:

- PROTECT parameter.
- LABEL parameter.
- MGMTCLAS parameter.
- STORCLAS parameter
- DSNNAME parameter: Use the DSNNAME parameter to assign a temporary data set name to an in-stream data set and to a SYSOUT data set. This name can be specified as a qualifier in JESSPOOL profile names. For more information, see [“Defining profiles for SYSIN and SYSOUT data sets”](#) on page 484.

When creating new data sets or tape volumes that require a new discrete profile, specify PROTECT=YES to automatically define the discrete profile.

**Note:** If the data set being created is adequately covered by a generic profile, do not use the PROTECT parameter because this forces the creation of a discrete profile.

- SECMODEL parameter: When creating new data sets or tape volumes that require a new discrete profile, specify the SECMODEL parameter to copy an existing data set profile to the new discrete data set profile.

**Note:** If the data set being created is adequately covered by a generic profile, do not use the SECMODEL parameter because this forces the creation of a discrete profile.

- On the OUTPUT statement:

- DPAGELBL parameter: With PSF for z/OS is installed, use the DPAGELBL parameter to indicate whether the system should print information related to the job's security label on each page of printed output.
- SYSAREA parameter: use the SYSAREA parameter to indicate whether the system should reserve an area on each page of printed output for information related to the security label.

## Restarting jobs

---

When a job automatically restarts and returns to a previous checkpoint, RACF repeats user verification and access authorization checking. If the job changed the password on the JOB statement, RACF uses the new password for user verification. But meanwhile, if the PASSWORD command or another job changes the password, RACF detects an invalid password and fails the job.

When you submit a job for a deferred restart, you can specify your current password on the JOB statement, or use JES user ID propagation and avoid the problem of exposing your password on the JOB statement.

For either an automatic or deferred restart, the user's current access authority is checked (the access authority at the time of the restart), and is used for all resources the job tries to access.

## Bypassing password protection

---

You can authorize certain programs to bypass password protection on resources such as data sets and volumes. You do this by specifying a setting in the MVS program properties table. Programs that you would normally authorize to bypass password protection include communication and database control programs, and other system control programs. RACF can be used to perform authorization checking, but RACF itself does not check the bypass password protection setting. Instead, programs that use RACF services, such as DFSMS, check the setting to determine whether to call RACF. Therefore, for information about how a product uses the bypass password protection setting, see its documentation.

## Controlling access to RACF passwords

---

Installation personnel should ensure that the security of RACF user passwords is not violated.

You should restrict the operator's use of the JES operator commands. Using JES commands, the system operator can display JES data areas that contain both the current and new RACF passwords associated

with a job, even though these passwords are in a masked format. (When a user submits a job and supplies RACF passwords on the JOB statement, JES stores them for the life of the job.)

It is also possible for the operator to display password information when displaying real storage at the console. Again, the installation should monitor the operator's activities to ensure that passwords remain secure. You can monitor the operator by creating profiles for the appropriate commands in the OPERCMDS class, and requesting auditing in the OPERCMDS profiles. If the operator has the OPERATIONS attribute, you can request additional logging by issuing the SETROPTS OPERAUDIT command. This causes logging of all activity that was allowed because the user had the OPERATIONS attribute.

Also, the JES3 dump core utility allows users to view stored passwords. You should restrict access to the JES3 dump core utility.

**Note:** JES3 allows system programmers to specify a password for the JES3 dump core utility (not a RACF password). This password is stored in clear text in a JES3 module. You should protect this module from unauthorized use.

RACF commands that contain sensitive values, such as passwords, should not be issued on the operator's console because sensitive information will appear in SYSLOG. Also, make sure you protect the GTF trace data set if you have SET TRACE active because sensitive information might appear in the trace records that are produced.

You should restrict access to SVC dumps and standalone dumps, which might contain password information.

If users need to submit jobs for other users, activate the SURROGAT class and define profiles so that users can allow other users to submit jobs for them. In this way, a user does not need to know anyone else's password. For more information, see [“Surrogate job submission” on page 467](#).

If JES support for user ID propagation is installed, batch jobs submitted by TSO users do not need any RACF identification information (user ID, group name, and password) in their JOB statements, as long as the following assumptions are true:

- The TSO user is RACF-defined.
- The job is submitted under the user's own user ID.
- If the job is submitted on a processor that is part of an NJE (networking job entry) network, the job runs on the home (user's) node.

**Note:** If a user specifies `//DD DATA` and neglects to delimit the data (with `/*` or DLM specification) when submitting a batch job through a card reader or RJE work station, subsequent jobs are read as part of the user's data until a delimiter is read. You should be aware that if this situation occurs, RACF user IDs, group names, passwords, and resource names from the following job's JCL become available to the user who failed to supply a delimiter. The installation should use SMF or JES installation-written exit routines to restrict the use of the `//DD DATA` statement to reduce this security exposure.

## Authorizing only RACF-defined users to access RACF-protected resources

---

If the universal access authority (UACC) for a RACF-protected resource is READ or higher:

- Non-RACF-defined users can access the RACF-protected resource with the specified level of universal access.
- Users who enter the system using shared RACF-defined user IDs without the RESTRICTED attribute, can access the RACF-protected resource with the specified level of universal access. These users include those who enter the system:
  1. By presenting digital certificates that are not registered to RACF, who are assigned shared user IDs based on certificate name filtering.
  2. By accessing application servers that allow users to enter the system without identifying themselves, who are assigned shared user IDs such as PUBLIC or ANONYMOS.

**Note:** For more information, see [“Defining restricted user IDs” on page 75](#).

- RACF-defined users who have access authority of NONE can access the resource with the specified level of universal access by submitting a batch job without specifying the USER operand on the JCL JOB statement.

The entry ID(\*) can be added to the access list to ensure that only RACF-defined users (who do not have the RESTRICTED attribute) can access a protected resource. For more information, see [“Using ID\(\\*\) on the access list” on page 6](#).

These accesses to RACF-protected resources can be prevented using the SETROPTS BATCHALLRACF and XBMALLRACF options, or by the REQUEST=AUTH preprocessing exit routine that fails REQUEST=AUTH processing for users who have entered the system using the RACF default user ID.

If JES user ID propagation is not in effect, this REQUEST=AUTH processing requires RACF-defined users to identify themselves (using the USER operand) on batch jobs that access RACF-protected resources and prevents non-RACF users from accessing RACF-protected resources.

## Using the TSO or ISPF editor

---

If a user edits a RACF-protected data set to which the user has only READ access authority, a failure occurs when the user attempts to save the data set. To issue the SAVE command, the user must have at least UPDATE access authority to the data set.

## Service by IBM personnel

---

If IBM support personnel require access to the system for servicing, they must be defined to RACF if they need to access RACF-protected data sets for servicing. Also, they need the appropriate access authority to these data sets.

You can define user profiles for IBM support personnel with the REVOKE attribute set. Then an authorized user can set (and reset), as needed, the REVOKE attribute in the user profile to allow IBM support personnel to enter the system. (The REVOKE and RESUME operands of the ALTUSER or CONNECT command alter the REVOKE attribute. See [z/OS Security Server RACF Command Language Reference](#) for more information.)

## Failsoft processing

---

During failsoft processing (when the RACF database is not active), RACF uses global access checking tables, REQUEST=LIST in-storage profiles, or a supplied profile, if any of these are present, to process resource access checking requests.

**Note:** RACF does not perform generic profile checking, because a generic profile might allow access to a resource that an existing discrete profile already protects. If that profile had been retrieved, RACF would not have allowed access to the resource.

RACF calls REQUEST=AUTH and REQUEST=DEFINE preprocessing installation exits during failsoft processing. (RACF does not call postprocessing exits.) This action frees the installation to define its own version of failsoft processing. By defining its own version of failsoft processing, an installation can allow or deny access to a resource or permit normal failsoft processing to continue.

During failsoft processing, the logging that your installation has specified continues as when RACF is active. In addition, RACF logs all accesses that the operator allows or denies.

If no global access checking tables are present, no REQUEST=LIST in-storage profiles are present, and no profile has been supplied, the preprocessing installation exits are called first. Then failsoft processing continues as follows:

### 1. RACROUTE REQUEST=AUTH:

- For started procedures, RACF issues an information message to the operator to describe the name and access mode of the resource. If the started procedure does not have the privileged attribute



through the RACF started procedures table, RACF issues an operator intervention message to request permission to allow access to the resource.

- For TSO sessions, RACF issues the information message and, if the high-level qualifier of the data set name matches the user's TSO user ID, RACF allows access to the resource. If the high-level qualifier does not match the user's TSO user ID, RACF also issues an operator intervention message to request permission to allow access to the resource. If the system operator gives a negative response to a request for access, the request is denied, with, in some cases, an ABEND.
- For all other environments, RACF issues the information message, followed by the operator intervention message. If the system operator gives a negative response to a request for access, the request is denied.

### 2. REQUEST=DEFINE:

RACF issues an operator message to indicate that REQUEST=DEFINE has been issued and that the request is allowed. If the user had the ADSP attribute, or if PROTECT=YES was specified on the JCL for the data set, the resource can be RACF-indicated without a RACF discrete profile being created.

You can use the operator message or SMF log records at a later time to determine whether the specified resource is in the RACF database. If it is not, use the ADDSD or RDEFINE command to create a profile for the resource.

## Failsoft processing with tape data sets

When RACF is maintaining TVTOCs, RACF checks the TVTOC during normal processing to determine the authority level that is required to define a data set or add a data set to a volume. In failsoft mode, RACF cannot make any of the normal consistency checks to ensure that the user is only writing to the last data set on the volume and is authorized to the current data on the tape volume.

## Considerations for RACF databases

---

The RACF database contains all RACF access control information. RACF databases can reside on any DASD device that is supported by the operating system. Each volume that contains a RACF database should be permanently resident. If RACF is heavily used, plan to put the database on a device accessed by a channel and control unit that is least likely to affect system performance.

## Backup RACF database

RACF allows you to provide a backup database to which you can switch should your primary RACF database fail. A backup RACF database reflects the contents of the primary. Once the installation creates the backup, RACF can maintain it automatically.

For more information about setting up a backup RACF database, maintaining RACF databases, and switching to alternate RACF databases, see [z/OS Security Server RACF System Programmer's Guide](#).

## Multiple data set support

As an alternative to maintaining all RACF profiles of your primary RACF database in a single MVS data set, RACF allows you to have up to 90 primary data sets, and an equal number of associated backup data sets. You should consider using multiple RACF data sets to reduce device contention and to reduce the number of resources that are made unavailable by the loss of one data set or device (although for installations running in data sharing mode, a single data set might provide satisfactory performance). The best number of RACF data sets for your installation depends on the extent to which you use RACF.

For more information about creating multiple RACF data sets, see [RACF database split/merge/extend utility program \(IRRUT400\)](#) in [z/OS Security Server RACF System Programmer's Guide](#).

## Protecting the RACF database

It is important that the data sets containing the primary and backup databases are properly protected. You should also ensure that data sets containing RACF database information, such as backup copies and



unloaded versions of the RACF database, are also protected. In protecting these data sets, you should ensure that only those users who have a definite job-related need to read or update the data have access. Any other users should not have access to the data sets containing your RACF databases.

- These data sets should be protected with data set profiles that specify UACC(NONE), NOWARNING, and ERASE. The profiles should not have ID(\*) in the access list. The NOTIFY user ID should be the RACF security administrator. System programmers who need to use the block update command (BLKUPD) to repair the RACF database must have UPDATE authority to the database. System programmers and others who need to run IRRUT400 or IRRUT200 to copy the database will need READ authority (or UPDATE, if using the LOCKINPUT or UNLOCKINPUT parameters of IRRUT400). Anyone who needs to run IRRDBU00 against a RACF database might also need UPDATE access, but it might be better to give them READ access and have them make a copy of the database using IRRUT200, then run IRRDBU00 against the copy. See [“Using the RACF database unload utility \(IRRDBU00\)” on page 371](#) for more information.
- If your installation uses profiles in the DASDVOL class to allow access to volumes, carefully limit the number of users who have READ access to the volumes that hold the data sets containing the RACF database. For more information, see [“DASD volume authority” on page 167](#).

**Note:** If you make a copy of the database for the purpose of running IRRDBU00, protect the copy as you would the database itself. This recommendation includes the use of ERASE and data set encryption for the copy and the IRRDBU00 output.

## Encrypting a RACF VSAM data set

As of z/OS V3R1 (or z/OS V2R5 with the installation of APAR OA62267), it is possible to encrypt a RACF VSAM data set. If you choose to encrypt a RACF data set, you must ensure that your infrastructure supports data set encryption in its disaster and recovery procedures.

The following points are key to understanding data set encryption:

- An encrypted VSAM data set must be in extended format and SMS-managed.
- The key label of the ICSF key, which is used to encrypt and decrypt the data set, is set when the data set is created. The key label value is stored in the catalog encryption cell for the data set and is permanently associated with the data set.

The key label can be specified in the following ways:

- By the security administrator, in the DFP segment of a data set profile, or in the DATACLAS ACS routine for the data set.
- At the time of data set creation, by using the DSKEYLBL JCL keyword, the DSKEYLBL keyword on the TSO ALLOCATE command, or through the IDCAMS KEYLABEL keyword.

After the key label is established for a data set, the data set label remains bound to the data set until the data set is deleted.

- The validity of the key object is not validated until the data set is opened.
- When the RACF data set is opened, in addition to an authorization check for the RACF data set, more authorization checks might be performed. These checks are intended to verify that the process that is opening the RACF data set is authorized to the following:
  - CSFKRR2 resource in the CSFSERV class
  - Key label that is associated with the RACF data set.

RACF data set OPENS that are performed during RACF initialization during IPL or during the RVARY process (including the RVARY that is performed by IRRUT200 PARM=ACTIVATE or PARM=RENAMEACTIVATE are not subject to these authorization checks.

For more information about data set encryption, see the following references:

- IBM Redbooks publication *Getting Started with z/OS Data Set Encryption*, SG24-8410-01
- The topic on [“Protecting data sets” in z/OS DFSMS Using Data Sets](#)

- Restrictions on sharing a RACF VSAM data set in [z/OS Security Server RACF System Programmer's Guide](#).

**Important:** Regarding the configuration requirements for data set encryption, IBM strongly recommends that data set encryption is enabled and tested with application data before you encrypt a live RACF data set. Your first encrypted data set should not be a live RACF data set.

Key rotation for an encrypted RACF VSAM data set is performed by copying the current RACF data set to a new data set that is encrypted with a different key. Either IRRUT200 or IRRUT400 can be used. However, using IRRUT200 with PARM=RENAMEACTIVATE allows key rotation to be performed without an IPL. For more information about the IRRUT200 and IRRUT400 utilities, see [z/OS Security Server RACF System Programmer's Guide](#).

## Using RACF data sharing

You can also reduce device contention to the RACF database by using RACF data sharing. You also need to consider the following when the system is enabled for sysplex communication:

- Each member of the sysplex data sharing group must share the same database.
- The members of the sysplex data sharing group cannot share the database with any non-member system.
- Each system must use a compatible data set name table.
- Each system must use an identical database range table.

For more information about shared or multiple RACF databases, see [z/OS Security Server RACF System Programmer's Guide](#).

## Sharing data without sharing a RACF database

You might find it useful to share RACF data between systems. The RACF remote sharing facility (RRSF) removes the restrictions of shared DASD. It allows you to configure your systems into a network of RRSF nodes and share RACF data between these nodes regardless of their physical proximity. You can:

- Give each RRSF node its own copy of the same RACF database and use remote sharing functions to keep the databases synchronized.
- Synchronize subsets of the database information, such as the user profiles.
- Administer RACF databases remotely.
- Automatically synchronize passwords for specified user IDs on systems in the RRSF network.

For more information on administering the RACF remote sharing facility, see [Chapter 17, “The RACF remote sharing facility \(RRSF\),”](#) on page 405.

## Number of resident data blocks

IBM highly recommends that you use resident data blocks to reduce the number of I/O requests made to the RACF database. See [z/OS Security Server RACF System Programmer's Guide](#) for details.

## Chapter 16. Working with the RACF database

This topic describes tasks that are related to working with and maintaining the RACF database by using the database unload utility (IRRDBU00) and the remove ID utility (IRRRID00).

For information about using the SMF data unload utility (IRRADU00), see [z/OS Security Server RACF Auditor's Guide](#).

The RACF database holds an installation's security data. This data is used to control access to resources, verify users, and generate various reports on system usage and integrity. Standard reports are provided and used to determine whether the installation's security objectives are being met.

**Note:** With z/OS V2R5, you can use a VSAM linear data set as your RACF database. For considerations, see [z/OS Security Server RACF System Programmer's Guide](#).

### Using the RACF database unload utility (IRRDBU00)

The RACF database unload utility enables installations to create a sequential file from a RACF database. The sequential file can be used in several ways: viewed directly, used as input for installation-written programs, and manipulated with sort/merge utilities. It can also be uploaded to a database manager, such as Db2z/OS, to process complex inquiries and create installation-tailored reports.

IRRDBU00 requires READ authority to the input RACF database if PARM=NOLOCKINPUT is specified. IRRDBU00 requires UPDATE authority to the input RACF database used only if PARM=LOCKINPUT or PARM=UNLOCKINPUT is specified.

**Restriction:** When you execute the IRRDBU00 utility against a database that is active on a system participating in a RACF sysplex data-sharing group, always execute IRRDBU00 from a system that participates in the RACF sysplex data-sharing group. If you do not, you might receive unpredictable results from the utility.

### Diagnosis

You can use the IRRDBU00 utility for some diagnosis functions. Because this utility reads every profile in the RACF database, it also validates such profile data as lengths and count fields that are needed to read each profile successfully.

This validation can be used to help identify a profile in error. If IRRDBU00 encounters a profile in error, it might issue message IRR67092. This message contains an ICHEINTY return and reason code and also the entry name of the profile being processed.

If a profile abends or ends in another fashion without receiving this message, you might also be able to determine the profile in error. To do this, look in the output data set (OUTDD) and find the last profile (at the bottom), that was unloaded. It is likely that this profile is okay. However, the next profile in the database (in the same class) could possibly be the profile in error, if indeed a bad profile is causing the utility to end.

The next profile in the database can be found by examining the output of an IRRUT200 utility run (specifically, INDEX FORMAT), or by using the block update command (BLKUPD) to examine an online database.

For more information on diagnosing and correcting the RACF database, see [z/OS Security Server RACF System Programmer's Guide](#) and [z/OS Security Server RACF Diagnosis Guide](#).

### Performance considerations

IRRDBU00 processes either a copy of the RACF database, a backup RACF database, or the active RACF database. You must have UPDATE authority to the database. It is *recommended* that you run the utility against a recent copy of your RACF database using the NOLOCKINPUT option.

While processing, IRRDBU00 serializes on one profile at a time (this is also the case in IRRUT100 processing). When IRRDBU00 has finished copying a profile, it releases the serialization. Consider this possible impact to performance if you select your active RACF database as input. Running IRRDBU00 against a *copy* of the database causes the least impact to system performance.

**Note:** If your system is enabled for sysplex communication RACF serializes database accesses by using global resource serialization instead of hardware RESERVEs when the system is operating in data sharing mode or in read-only mode and unloading an active database.

An installation can choose to unload its database with one utility invocation, or if it has split its database, it can unload individual pieces of its database with separate utility invocations. These utility invocations can execute concurrently.

## Operational considerations

The output records of IRRDBU00 are determined by the structure of the RACF database. The utility unloads all profiles in the database. It does not unload all fields in each profile and treats some fields in a special way.

- Encrypted and reserved fields are not unloaded.
- Fields that contain installation data are unloaded exactly as they appear in the database, with the exception of CSDATA fields.
- Fields in the CSDATA segment are unloaded according to the data type defined within each field.
- When possible, fields that contain UTF-8 data are translated to EBCDIC using the code page specified in the FACILITY class profile IRR.IDIDMAP.PROFILE.CODEPAGE if defined and contains a valid code page. Otherwise, code page IBM-1047 is used. For more details, see [“Restrictions for UTF-8 data values” on page 677](#).

The maximum length of unloaded data for most fields is 255 bytes. However, the entire length of data is unloaded for the following fields:

- Up to 1023 bytes for the HOME and PROGRAM fields in the OMVS segment of a user profile.
- Up to 1023 bytes for the HOME, PROGRAM, and FSROOT fields in the OVM segment of a user profile.
- Up to 1100 bytes for each field in the CSDATA segment of a RACF profile.

See [z/OS Security Server RACF Macros and Interfaces](#) for the conversion rules of the database unload utility.

The database unload utility uses the supplied class descriptor table (ICHRRCDX), the installation-defined class descriptor table (ICHRRCDE), and the dynamic class descriptor table, as it unloads profiles. If your database is imported from another system, you might also have to import ICHRRCDX and ICHRRCDE and create new dynamic classes.

The database unload utility unloads a class only when *both* of the following conditions are true:

1. There is an entry for the class in either the ICHRRCDE, ICHRRCDX, or the dynamic class descriptor table on the system running the utility
2. The range table on the system running the utility correctly identifies the data set where the profiles are located.

If the current range table does not match the database being unloaded, you must run the IRRDBU00 utility multiple times, specifying only one data set of the database as INDD1 for each run.

To correlate the RACF profiles with the data unloaded by the utility, see [z/OS Security Server RACF Macros and Interfaces](#).

## Using IRRDBU00 with universal groups

Using the output from IRRDBU00 is the best way to list the members of a universal group, because a complete member list for a universal group might not be obtained from a LISTGRP command. You can use

Db2z/OS to process the output of IRRDBU00 to generate a list of universal groups and to list the members of each universal group by using samples available in SYS1.SAMPLIB. See [“Using the database unload utility output with Db2”](#) on page 382 for more information. Sample RACFICE reports called GPRM and CUG\$ are also available to assist you. See [“Reports based on the database unload utility \(IRRDBU00\)”](#) on page 378.

Note that IRRDBU00 does not unload a Group Members data record (0102) for every user connected to a universal group. Only users who are listed in the group's member list (users with group-level user attributes, such as group-SPECIAL, or group authority higher than USE) have 0102 records.

For more information about universal groups, see [“Defining large groups with the UNIVERSAL attribute”](#) on page 88.

## Running the database unload utility

The following job control statements are necessary for executing IRRDBU00:

### **JOB**

Initiates the job.

### **EXEC**

Specifies the program name (PGM=IRRDBU00) or, if the job control statements are in a procedure library, the procedure name. You must request IRRDBU00 processing options by specifying a parameter in the PARM field.

### **SYSPRINT DD**

Defines a sequential message data set.

### **INDDn DD**

Defines the RACF input data set that makes up the RACF database. The input data sets must have all of the characteristics of a RACF database; that is, they must be contiguous single-extent data sets, non-VIO, with a logical record length (LRECL) of 4096 and a record format (RECFM) of fixed (F).

The *n* in INDD*n* refers to the location of the data set name in the data set name table. If you have not split your RACF database, you only have to specify INDD1. If you have split your RACF database, you can unload each part with a separate utility invocation and specify INDD1 for the input data set, or you can unload all of the parts with one utility invocation.

**Note:** When unloading all parts, specify INDD statements in the same order as they appear in the RACF data set name table. For example:

#### **INDD1**

First data set name

#### **INDD2**

Second data set name

See [“Input data set specification”](#) on page 374.

### **OUTDD DD**

Defines the single sequential output data set. The output of IRRDBU00 is a set of variable length records.

The size of the output data set is roughly estimated as twice the size of the used portion of the input data set, but you must also consider the type of profiles in your database. For example, profiles that have variable length fields, such as installation data, require more space when they are unloaded, because the maximum size of the field is unloaded (up to 255 bytes or, for the HOME and PROGRAM fields, up to 1023 bytes).

Determine the percentage of space your database is using by running the IRRUT200 utility, and use that percentage to guide you in allocating the output file. For example, if your database has 100 cylinders allocated and you are using 35% of it, you need approximately 70 cylinders for your output file.

OUTDD is a variable blocked data set (RECFM=VB). The LRECL for the output data set must be at least as large as the largest record created by IRRDBU00.

**Guideline:** Choose an LRECL value of at least 4096 so that if the size of the output records increases when new fields are added, you do not have to change your data set allocations.

## Input data set specification

Allowable DD names for the data sets that correspond to the input data sets are INDD1 through INDD255. The input data sets must be numbered consecutively. For example, if 25 input data sets are provided, they must be assigned DD names INDD1 through INDD25. The utility processes the input data sets until a number is omitted. You must provide at least one input data set (INDD1).

The DD name number must correspond to the position of the input data set name in the data set name table. That is, you must assign INDD1 to the first data set, INDD2 to the second, and so on.

### Restrictions:

- The input data sets must reside on DASD. Tape input data sets are not supported.
- You must specify input data sets using their real names. Specifying alias names for data sets of the RACF database is not supported in the RACF data set name table and is not supported for use with RACF utilities.

## IRRDBU00 example

In this example, the database unload utility processes a database that has been split into three parts. The job control language (JCL) statements that invoke the utility are:

```
//USER01  JOB  Job card...
//UNLOAD  EXEC PGM=IRRDBU00,PARM=NOLOCKINPUT
//SYSPRINT DD  SYSOUT=*
//INDD1   DD  DISP=SHR,DSN=SYS1.RACFDB.PART1.COPY
//INDD2   DD  DISP=SHR,DSN=SYS1.RACFDB.PART2.COPY
//INDD3   DD  DISP=SHR,DSN=SYS1.RACFDB.PART3.COPY
//OUTDD   DD  DISP=SHR,DSN=SYS1.RACFDB.FLATFILE
```

**Note:** You must specify a parameter in the PARM field on the EXEC statement of the step executing IRRDBU00.

## Allowable parameters

When you run the database unload utility, one of the following parameters must be specified: NOLOCKINPUT, LOCKINPUT, or UNLOCKINPUT. You can abbreviate the parameter to N, L, or U, respectively. For the *least* impact to system performance, use a copy of your RACF database as input and specify the NOLOCKINPUT parameter.

When your system is RACF is enabled for sysplex communication and RACF is running in read-only mode, the *only* parameter allowed for IRRDBU00 is NOLOCKINPUT.

Using the backup copy of the RACF database is allowed. Using an active copy of the RACF database can affect system performance and it is not recommended.

### The NOLOCKINPUT parameter

This value allows the unload to be performed and does not change the state of the input database. If the database is locked, it remains locked. If it is unlocked, it remains unlocked.

For the least impact to system performance, use a copy of your RACF database as input and specify the NOLOCKINPUT parameter.

**Important:** If you use NOLOCKINPUT on the *active* database, your unloaded database might contain inconsistencies.

### The LOCKINPUT parameter

This value ensures that the RACF database used as input is not updated by other jobs while the utility is running.

**Note:** Statistics are updated.

If you run against an active RACF database, LOCKINPUT is recommended.

Specifying LOCKINPUT means updates are no longer allowed to an input data set until the utility ends. If the RACF database is *locked* and users logging on attempt to change their user profiles, the logon might not be allowed, depending on the change. Users might be unable to:

- Change the password or password phrase
- Specify a correct password or password phrase after specifying an incorrect one
- Successfully complete the first logon of the day

If you run IRRDBU00 and use LOCKINPUT, any activity that tries to update the RACF database (such as users logging on and changing passwords or batch jobs allocating new data sets requiring the creation of RACF profiles) fails with either an ABEND483 RC50 or ABEND485 RC50.

When using LOCKINPUT against an active database, do not schedule maintenance that runs past midnight. If RACF is not running in data sharing mode and the RACF database remains locked past midnight, new jobs cannot be submitted and users cannot log on unless you disable the gathering of logon statistics by issuing a SETROPTS NOINITSTATS command. All steps that require a locked database must be performed on the same calendar day. This is because RACF updates both primary and backup logon statistics each day.

The database unload utility *unlocks* the RACF database after processing with LOCKINPUT specified if the database was unlocked when the utility started. The unload utility output is for report generation and does not replace the input database, which is your primary, active, RACF database.

This is different from the IRRUT400 utility, which keeps the input database locked and creates a new output database. This is done to maintain integrity between the input database and the output database.

## The UNLOCKINPUT parameter

UNLOCKINPUT is used to unlock a database that had previously been locked by the LOCKINPUT parameter. This action enables your input database and allows it to be updated.

No data unloading is done when this parameter is used.

## Using the database unload utility output effectively

The output file from the database unload utility can be:

- Viewed directly
- Used as input to your own programs
- Manipulated with sort/merge utilities
- Used as input to a database management system. Installations can produce reports that are tailored to their requirements.

## Sort/merge programs

The database unload utility processes all of the profiles in the input database. If you want a subset of the output records, you can use a standard utility such as DFSORT to select them. For example, the following DFSORT control statements select the Group Basic Data records (type 0100) and User Basic Data records (type 0200). All other record types are excluded.

```
SORT    FIELDS=(5,4,CH,A,10,20,CH,A)
INCLUDE COND=(5,4,CH,EQ,C'0100',OR,5,4,CH,EQ,C'0200')
OPTION  VLSHRT
```

## Using database unload utility output with the DFSORT ICETOOL

IBM's DFSORT product provides a reporting facility called ICETOOL. You can create ICETOOL reports based on output files from the RACF database unload utility (IRRDBU00) or the SMF data unload utility

(IRRADU00). The SYS1.SAMPLIB member IRRICE contains DFSORT statements for record selection and ICETOOL statements for report formatting for a wide variety of reports. The IEBUPDTE utility processes the IRRICE member and creates a partitioned data set (PDS) that contains two PDS members for each report. The two members contain:

1. The report format
2. The record selection criteria



**Attention:** For information about the RACF database unload utility (IRRDBU00), see [“Using the RACF database unload utility \(IRRDBU00\)”](#) on page 371. For information about the SMF data unload utility (IRRADU00), see [z/OS Security Server RACF Auditor's Guide](#).

### The report format

The report format has a 1 - 4 character name (for example, UGRP). It contains the ICETOOL statements that control report format and record summary information, such as SORT, COPY, DISPLAY, and OCCURS statements. An example of a report format member is shown in [Figure 10 on page 376](#). This is the report format member UGRP, which is the report format for the "Users With Extraordinary Group Authorities" report.

```
*****
* Name: UGRP                                     *
*                                               *
* Find all of the user IDs which have extraordinary RACF privileges, *
* such as SPECIAL, OPERATIONS, and AUDITOR at the group level.    *
*****
SORT FROM(DBUDATA) TO(TEMP0001) USING(RACF)
DISPLAY FROM(TEMP0001) LIST(PRINT) -
PAGE -
TITLE('UGRP: Users With Extraordinary Group Authorities') -
DATE(YMD/) -
TIME(12:) -
BLANK -
ON(10,8,CH) HEADER('User ID') -
ON(19,8,CH) HEADER('Group ID') -
ON(88,4,CH) HEADER('Group Special') -
ON(93,4,CH) HEADER('Group Operations') -
ON(113,4,CH) HEADER('Group Auditor')
```

Figure 10. Member UGRP: Users with extraordinary group authorities: report format statements

See [z/OS Security Server RACF Macros and Interfaces](#) for the conversion rules of the database unload utility.

### The record selection criteria

The name of the member containing the record selection criteria is the report member name followed by CNTL (e.g. UGRPCNTL). Record selection is performed using DFSORT control statements, such as SORT and INCLUDE. The SORT command is used to select and sort records. The INCLUDE command is used to specify conditions required for records to appear in the report.

An example of a record selection member is shown in [Figure 11 on page 376](#). This is the report selection member UGRPCNTL, which contains the selection criteria for the "Users With Extraordinary Group Authorities" report. In this example, we are including only User Connect Data records (record type 0205) when the user has the group-SPECIAL, group-OPERATIONS or group-AUDITOR attribute.

```
SORT FIELDS=(10,08,CH,A)
INCLUDE COND=(5,4,CH,EQ,C'0205',AND,
              (88,3,CH,EQ,C'YES',OR,
              93,3,CH,EQ,C'YES',OR,
              113,3,CH,EQ,C'YES'))
OPTION VLSHRT
```

Figure 11. Member UGRPCNTL: Users with extraordinary group authorities: record selection statements



See *z/OS Security Server RACF Macros and Interfaces* for record format information for the output records of the IRRADU00 and IRRDBU00 utilities. See *z/OS DFSORT Application Programming Guide* for the complete details of the DFSORT statements.

**Important note about column numbers:** Both IRRADU00 and IRRDBU00 create records that are variable-length. Variable-length records have a four-byte record descriptor word (RDW) describing the length of the record. DFSORT considers the RDW to be part of the selectable record columns. This means that you must add 4 to any of the field positions identified for the IRRADU00 and IRRDBU00 records described in *z/OS Security Server RACF Macros and Interfaces*. In the example in Figure 11 on page 376, the IRRDBU00 field for record type 0205 is defined in *z/OS Security Server RACF Macros and Interfaces* as beginning at record position 1. We add 4 to this position to get 5, the value that we must use in both the DFSORT INCLUDE statement for record selection and the ICETOOL ON operand to select the fields for the report.

### Using the RACFICE procedure to generate reports

You can invoke the ICETOOL utility with the RACFICE procedure contained in the IRRICE member of SYS1.SAMPLIB. It simplifies the JCL required to execute ICETOOL reports and contains JCL symbolic variables that represent the input to the RACFICE procedure. These variables are:

#### DBUDATA

Output of IRRDBU00 that is being used as input

#### ADUDATA

Output of IRRADU00 that is being used as input

#### REPORT

The name of the report that is being generated

See “Reports based on the database unload utility (IRRDBU00)” on page 378 for a list of the reports based on IRRDBU00 output that are shipped with this support. See *z/OS Security Server RACF Auditor's Guide* for a list of the reports based on IRRADU00 output that are shipped with this support.

See “Creating customized reports” on page 380 for information about creating your own reports.

You do not need to specify each of these variables every time you execute the RACFICE procedure. For example, if you specify the default IRRDBU00 and IRRADU00 data sets in the RACFICE procedure, you create a report (shown in Figure 12 on page 377) that lists all user IDs with extraordinary RACF group authorities with the following JCL:

```
//jobname JOB Job card...
//stepname EXEC RACFICE,REPORT=UGRP
```

If the default IRRDBU00 or IRRADU00 data sets are not correct, you can override them. For example, if the IRRDBU00 output is in the data set USER01.TEST.IRRDBU00 and the IRRADU00 output is in the data set USER01.TEST.IRRADU00, you should enter:

```
//jobname JOB Job card...
//          SET ADUDATA=USER01.TEST.IRRADU00
//          SET DBUDATA=USER01.TEST.IRRDBU00
//stepname EXEC RACFICE,REPORT=UGRP
```

```
1- 1 -   UGRP: Users With Extraordinary Group Authorities   99/04/13

  User ID      Group ID      Group Special      Group Operations      Group Auditor
  -----      -
  LCLAUDIT     GROUP1        NO                NO                YES
  LCLOPER      GROUP1        NO                YES               NO
  LCLSPEC      GROUP1        YES               NO                NO
  UCAUDR$Y     GPCONNEC      NO                NO                YES
  UCOPER$Y     GPCONNEC      NO                YES               NO
  UCSPEC$Y     GPCONNEC      YES               NO                NO
```

Figure 12. Report of all users with extraordinary group authorities

**Reports based on the database unload utility (IRRDBU00)**

The following reports are based on the output of IRRDBU00. You can find a sample of each report in SYS1.SAMPLIB.

**Name****Description****ALDS**

Data set profiles that have IDs on the standard access list with ALTER authority. **Value:** Identifies users who can alter the access list of the profile.

**ASOC**

Users who have explicit RACF remote sharing facility (RRSF) associations defined. **Value:** Identifies users who can direct commands.

**BGGR**

Discrete general resource profiles with generic characters. **Value:** Finds profiles that aren't protecting what you think that they are protecting.

**CCON**

Count of user's connections, flagging those users with more than "x" connections. **Value:** Helps find a performance bottleneck that is caused by excessive group connections.

**CGEN**

Count of general resource profiles. **Value:** Identifies basic characteristics of the RACF database.

**CKEY**

DIGTCERT class profile with key type PCICCTKN. **Value:** Identifies the certificates with private key stored in PKDS which are PCICC key types and have key sizes of 2048 bits or greater.

**CLBL**

Count of certificate labels that will be the same as the existing ones after folding to uppercase and conversion of blanks and special characters to "\_". **Value:** Identifies certificate labels that might need to change so that they can be covered by different protection profiles that are based on certificate labels in the RDATA LIB class.

**CONN**

User IDs with group authorities higher than USE. **Value:** Identifies users with additional privileges.

**CPRO**

Count of user, group, and data set profiles. **Value:** Identifies basic characteristics of the RACF database.

**CUG\$**

Group names of all universal groups, listing their owners and creation dates. **Value:** Identifies universal groups that might have members who do not appear in the group's member list.

**GIDS**

z/OS UNIX GIDs that are used more than once. **Value:** Identifies z/OS UNIX groups that are sharing authority characteristics.

**GRPM**

User IDs of all members of a group, including a universal group, listing the owner of each connection, and group-related user attributes for each member. **Value:** Provides a complete member listing for universal groups, which is not available using the LISTGRP command.

**IDSC**

Data set conditional access list entries with an ID(\*) entry of other than NONE. **Value:** Identifies data set profiles that allow any RACF-authenticated user to access data.

**IDSS**

Data set standard access list entries with an ID(\*) entry of other than NONE. **Value:** Identifies data set profiles that allow any user authenticated by RACF to access data.

**IGRC**

General resource conditional access list entries with an ID(\*) entry of other than NONE. **Value:** Identifies general resource profiles that allow any user authenticated by RACF to access data.

**IGRS**

General resource standard access list entries with an ID(\*) entry of other than NONE. **Value:** Identifies general resource profiles that allow any user authenticated by RACF to access data.

**NPWI**

User IDs that have NOINTERVAL specified as their password interval. **Value:** Identifies users who are not required to change their passwords.

**OMVS**

User IDs that have an OMVS segment defined. **Value:** Identifies users who can use z/OS UNIX with a non-default UID.

**PCAM**

Program class profiles with specific program names that have 'MAIN' or 'BASIC' for the APPLDATA. **Value:** Identifies programs that can be used as first program in ENHANCED program security mode.

**SUPU**

z/OS UNIX "superusers" (UID of zero). **Value:** Identifies users who have extraordinary privileges within the z/OS UNIX environment.

**UADS**

Data set profiles with UACCs other than NONE. **Value:** Identifies data set profiles that allow any user to access data.

**UAGR**

General resource profiles, excluding profiles in the DIGTCERT class, with UACCs other than NONE. **Value:** Identifies general resource profiles that allow any user to access data.

**UGLB**

User IDs with extraordinary global authorities. **Value:** Identifies users with extraordinary RACF authority.

**UGRP**

User IDs with extraordinary RACF group authorities. **Value:** Identifies users with extraordinary RACF authority.

**UIDS**

z/OS UNIX UIDs that are used more than once. **Value:** Identifies z/OS UNIX users who are sharing authority characteristics.

**URVK**

User IDs which are currently revoked. **Value:** Identifies users who have had a revocation performed.

**WNDS**

Data set profiles that are in WARNING mode. **Value:** Identifies data set profiles that are processing in WARNING mode.

**WNGR**

General resource profiles that are in WARNING mode. **Value:** Identifies general resource profiles that are processing in WARNING mode.

In addition, the following reports demonstrate advanced ICETOOL techniques:

**Name****Description****\$CERT01**

All DIGTCERT profiles. **Value:** Displays all certificate profiles with the following information: Owner, Label, Certificate Fingerprint, Subject Distinguished Name, Issuer Distinguished Name and Signature Algorithm.

**\$CERT02**

All DIGTCERT profiles. **Value:** Displays all certificate profiles with the following information: Certificate Fingerprint, Subject Distinguished Name, Issuer Distinguished Name, Start Date/Time, End Date/Time, Signature Algorithm, Private Key Type, Private Key Size and Last Serial Number Issued.

**\$CFQG**

A count of the number of fully qualified generic profiles that are defined for each high-level qualifier (HLQ). **Value:** Identifies users who have defined an excessive number of fully qualified generic profiles.

**\$CHLQ**

A count of the number of generic profiles that are defined for each high-level qualifier (HLQ). **Value:** Identifies a potential performance bottleneck.

**\$CKEY**

DIGTCERT class profile with key type PCICCTKN. **Value:** Identifies the certificates with private keystored in PKDS which are PCICC key types and have key sizes of 2048 bits or greater.

**\$ULAST90**

Identifies the user profiles which have been created within the past 90 days. **Value:** Shows recent administrative activity.

Note that these reports (\$CERT01, \$CERT02, \$CFQG, \$CHLQ, \$CKEY, and \$ULAST90) are standalone reports and are not run using the RACFICE PROC.

***Creating customized reports***

You can create your own reports using the RACFICE procedure by following these steps:

1. Identify the records that you want for the report.
  - a. Define the DFSORT statements for the record selection criteria.
  - b. Place them in the RACFICE data set with a unique member name consisting of a 1 - 4 character report identifier followed by CNTL.

If there is an existing report that has similar selection criteria, use it as a model. For example, if you want to report all the access records created when users PATTY, MAXINE, and LAVERNE accessed resources, you need to create DFSORT selection statements that look like [Figure 13 on page 380](#) and store them in your RACFICE report data set as the PMLCNTL record selection criteria.

```
INCLUDE COND=(63,8,CH,EQ,C'PATTY',OR,
              63,8,CH,EQ,C'MAXINE',OR,
              63,8,CH,EQ,C'LAVERNE')
OPTION VLSHRT
```

*Figure 13. Customized record selection criteria*

Note the similarity of this record selection criteria to the "Users With Extraordinary Group Authorities Report" record selection criteria shown in [Figure 11 on page 376](#).

See [z/OS DFSORT Application Programming Guide](#) for the complete details of the DFSORT statements.

2. Identify the report format you want to use.
  - a. Define the ICETOOL statements for the report format.
  - b. Place them in the RACFICE data set with a 1 - 4 character report identifier that you chose.

If there is an existing report that has similar report format, use it as a model. For example, if you wanted your report to contain the user ID, job name, date, time, and status of the access you could use the ICETOOL report statements shown in [Figure 14 on page 381](#), and store them in your RACFICE report data set as the PML report format.

```

COPY      FROM(ADUDATA) TO(TEMP0001) USING(RACF)
DISPLAY   FROM(TEMP0001) LIST(PRINT) -
          PAGE -
          TITLE('Patty, Maxine, and Laverne's Accesses') -
          DATE(YMD/) -
          TIME(12:) -
          BLANK -
          ON(63,8,CH)  HEADER('User ID') -
          ON(5,8,CH)   HEADER('Event') -
          ON(12,8,CH)  HEADER('Qualifier') -
          ON(23,8,CH)  HEADER('Time') -
          ON(32,10,CH) HEADER('Date') -
          ON(184,8,CH) HEADER('Job Name')

```

Figure 14. Customized report format

Note the similarity of this report format to the "Users With Extraordinary Group Authorities" report format shown in [Figure 10 on page 376](#).

For complete details on the ICETOOL statements, see [z/OS DFSORT Application Programming Guide](#).

3. Update the report JCL to invoke the RACFICE procedure with the 1 - 4 character report identifier you chose, as shown in [Figure 15 on page 381](#).

```

//jobname JOB Job card...
//stepname EXEC RACFICE,REPORT=PML

```

Figure 15. Customized report JCL

#### Creating a RACFVARS member report

Because the output of the RLIST command lists the members of a general resource profile in alphabetical order, it might be helpful to list the members in the order in which they occur in the profile. To do this, you might use the ICETOOL to format the report. The following statement samples can be used to list the members of a RACFVARS class profile called &PAYTAPE.

The following DFSORT statements select the needed record (type 0503) from the IRRDBU00 output records:

```

INCLUDE COND=(10,3,CH,EQ,C'&PAYTAPE',AND,
              257,8,CH,EQ,C'RACFVARS',AND,
              5,4,CH,EQ,C'0503')
OPTION VLSHRT

```

The following ICETOOL statements control the report format:

```

COPY      FROM(DBUDATA) TO(TEMP0001) USING(RACF)
DISPLAY   FROM(TEMP0001) LIST(PRINT) -
          ON(266,255,CH) HEADER('MEMBER NAME')

```

Here is a sample output report:

```

MEMBER NAME
-----
D
C
A
B

```

## Relational databases

Much of the function of the database unload utility is not realized until the data it creates is loaded into a relational database management system (DBMS) such as Db2z/OS.

## Using the database unload utility output with Db2

You can use the Db2z/OS load utility or its equivalent to process the records produced by the database unload utility. The definition and control statements for a Db2z/OS utilization of the output, all of which are contained in SYS1.SAMPLIB, are as follows:

- Sample data definition language (DDL) statements to define the relational presentation of the RACF database and sample Db2z/OS definitions which perform database and index creation. These are contained in member RACDBUTB.
- Sample control statements for the Db2z/OS load utility that map the output from the database unload utility (IRRDBU00). These are contained in member RACDBULD.
- Sample structured query language (SQL) queries that perform the following queries. These are contained in member RACDBUQR.
  - Listing all of the members of a group, including a universal group
  - Listing all of the users with the SPECIAL attribute
  - Finding all of the groups to which a user is connected
  - Finding all of the data set access lists that contain user IDs that are no longer valid
  - Listing of z/OS UNIX user identifiers (UIDs) with associated user ID and programmer name
  - Listing of z/OS UNIX group identifiers (GIDs) with associated group name
  - Listing of UIDs including only those users who are connected to a group that has a GID (for each UID, the user ID and programmer name are listed)

For more information on Db2z/OS, visit [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims).

### Steps for using IRRDBU00 output with Db2

To create and manage a Db2z/OS database that contains the output from the database unload utility, you must:

1. Create one or more Db2z/OS databases.
2. Create one or more Db2z/OS table spaces.
3. Create Db2z/OS tables.
4. Create the Db2z/OS indexes.
5. Load data into the tables.
6. Reorganize the data in the tables (optional).
7. Create performance statistics (optional).
8. Delete table data (optional).

The first three steps are initial setup, and you can choose to run them once. When you get new data to import into the Db2z/OS database, you delete your current table data. You then reload and reorganize your tables and create the performance statistics.

The following sections show examples of the Db2z/OS utility input for these functions.

### Creating a Db2 database for unloaded RACF data

A Db2z/OS database names a collection of table spaces. The following SQL statement creates a Db2z/OS database for the output of the database unload utility:

```
CREATE DATABASE databasename
```

where *databasename* is supplied by the user.

## Creating a Db2 table space

A table space is one or more data sets in which one or more tables are stored. [Figure 16 on page 383](#) contains examples of SQL statements that create a table space. There are other methods of allocating a table space. For details, see the Db2z/OS documentation.

Member RACDBUTB in SYS1.SAMPLIB contains statements that create a table space.

```
CREATE TABLESPACE tablespacename IN databasename
  LOCKSIZE TABLESPACE
  SEGSIZE 64
  PCTFREE 0
  USING STOGROUP storagegroup
  PRIQTY 2000
  SECQTY 500
  CLOSE NO
  ;
```

Figure 16. Sample SQL utility statements: Defining a table space

The user must supply the name of the table space (*tablespacename*) and the storage group (*storagegroup*). The sample shows a value of 64 for SEGSIZE, 2000 for PRIQTY, and 500 for SECQTY.

The samples in RACDBUTB put all of the tables into one table space. The sample also suggests using a segment size because segmented table spaces improve performance. Users might want to define their own table spaces rather than use table spaces that are defined by the storage group.

Installations have a number of other options, such as the number of table spaces to use, the type of spaces, and the security for the data. They might want to keep the number of tables per table space fairly small for better performance and might want to consider putting the larger tables into separate table spaces.

## Creating the Db2 tables

After the database and the table space are created, SQL statements that define the tables are executed. [Figure 17 on page 383](#) contains an example of the SQL statements that are required to create a table for the Group Basic Data record of the database unload utility.

Member RACDBUTB in SYS1.SAMPLIB contains examples that create separate tables for each record type that is produced by the database unload utility. The user must supply the user ID (*userid*).

```
CREATE TABLE userid.GROUP_BD (
  GPBD_NAME          CHAR(8)    NOT NULL,
  GPBD_SUPGRP_ID     CHAR(8),
  GPBD_CREATE_DATE   DATE,
  GPBD_OWNER_ID      CHAR(8)    NOT NULL,
  GPBD_UACC          CHAR(8)    NOT NULL,
  GPBD_NOTERMUACC    CHAR(1)    NOT NULL,
  GPBD_INSTALL_DATA  CHAR(254),
  GPBD_MODEL         CHAR(44)
)
IN databasename.tablespacename
;
```

Figure 17. Sample SQL utility statements: Creating a table

## Creating the Db2 indexes

Db2z/OS performance improves with the use of indexes. Member RACDBUTB in SYS1.SAMPLIB creates an index for every primary key and every foreign key identified in the record types. [Figure 18 on page 384](#) contains sample statements to create the indexes for the Group Basic Data record.

```

CREATE UNIQUE INDEX userid.GROUP_BD_IX1
ON userid.GROUP_BD
  (GPBD_NAME)
  USING STOGROUP storagegroup
  PRIQTY 100
  SECQTY 50
  CLUSTER
  PCTFREE 0
  CLOSE NO
;

CREATE UNIQUE INDEX userid.GROUP_BD_IX2
ON userid.GROUP_BD
  (GPBD_NAME, GPBD_SUPGRP_ID)
  USING STOGROUP storagegroup
  PRIQTY 100
  SECQTY 50
  PCTFREE 0
  CLOSE NO
;

CREATE INDEX userid.GROUP_BD_IX3
ON userid.GROUP_BD
  (GPBD_OWNER_ID)
  USING STOGROUP storagegroup
  PRIQTY 100
  SECQTY 50
  PCTFREE 0
  CLOSE NO
;

CREATE INDEX userid.GROUP_BD_IX4
ON userid.GROUP_BD
  (GPBD_MODEL)
  USING STOGROUP storagegroup
  PRIQTY 100
  SECQTY 50
  PCTFREE 0
  CLOSE NO
;

```

Figure 18. Sample SQL utility statements: Creating indexes

### Loading the Db2 tables

Figure 19 on page 384 shows the statements that are required to load the Group Basic Data record. The RACDBULD member of SYS1.SAMPLIB contains statements that load all of the record types produced by the database unload utility.

```

LOAD DATA
  INDDN tablespacename
  RESUME YES
  LOG NO
  INTO TABLE userid.GROUP_BD
  WHEN(1:4)='0100' (
    GPBD_NAME          POSITION(006:013)  CHAR(8),
    GPBD_SUPGRP_ID     POSITION(015:022)  CHAR(8),
    GPBD_CREATE_DATE   POSITION(024:033)  DATE EXTERNAL(10),
    GPBD_OWNER_ID      POSITION(035:042)  CHAR(8),
    GPBD_UACC           POSITION(044:051)  CHAR(8),
    GPBD_NOTERMUACC     POSITION(053:053)  CHAR(1),
    GPBD_INSTALL_DATA   POSITION(058:311)  CHAR(254),
    GPBD_MODEL          POSITION(314:357)  CHAR(44)
  )

```

Figure 19. Db2 utility statements required to load the tables

**Note:** You can choose not to load some of the tables.

### Reorganizing the unloaded RACF data in the Db2 database

Queries are processed faster if they are performed against an organized database. The Db2z/OS utility statement required to reorganize the database is:

```
REORG TABLESPACE databasename.tablespaceName
```



## Creating optimization statistics for the Db2 database

Queries are processed faster if they are performed against an organized database for which Db2z/OS has collected performance statistics. The Db2z/OS utility statement required to create these statistics is:

```
RUNSTATS TABLESPACE databasename.tablespace
```

## Deleting data from the Db2 database

Before you reload the database with new data, you should delete the old data. This can be done in several ways:

1. Use the DROP TABLE statement for each table you want to delete.
2. Use the DROP TABLESPACE statement for each tablespace.
3. Delete all of the records in each table.

Figure 20 on page 385 shows the sample SQL statements that delete the group record data from the tables.

```
DELETE FROM userid.GROUP_BD ;
DELETE FROM userid.GROUP_DFP_DATA ;
DELETE FROM userid.GROUP_INSTALL_DATA ;
DELETE FROM userid.GROUP_SUBGROUPS ;
DELETE FROM userid.GROUP_MEMBERS ;
```

Figure 20. Db2 utility statements required to delete the group records

## Db2 table names

Member RACDBULD in SYS1.SAMPLIB creates Db2z/OS tables for each record type. Table 28 on page 385 provides a useful reference of record type, record name, and Db2z/OS table name.

For more information, see “Database unload utility output samples” on page 390.

Table 28. Correlation of record type, record name, and Db2 table name

Record type	Record name	Db2z/OS table name
0100	Group Basic Data	GROUP_BD
0101	Group Subgroups	GROUP_SUBGROUPS
0102	Group Members	GROUP_MEMBERS
0103	Group Installation Data	GROUP_INSTALL_DATA
0110	Group DFP Data	GROUP_DFP_DATA
0120	Group OMVS Data	GROUP_OMVS_DATA
0130	Group OVM Data	GROUP_OVM_DATA
0140	Reserved	—
0141	Group TME Role	GROUP_TME_ROLE
0150	Reserved	—
0151	Group CSDATA Custom fields	GROUP_CSDATA_CUST
0200	User Basic Data	USER_BD
0201	User Categories	USER_CATEGORIES
0202	User Classes	USER_CLASSES

Table 28. Correlation of record type, record name, and Db2 table name (continued)

<b>Record type</b>	<b>Record name</b>	<b>Db2z/OS table name</b>
0203	User Group Connections	USER_GROUPS
0204	User Installation Data	USER_INSTALL_DATA
0205	User Connect Data	USER_CONNECT_DATA
0206	User RRSF Data	USER_RRSF_DATA
0207	User Certificate Data	USER_CERT_DATA
0208	User Associated Mappings	USER_NMAP_DATA
0209 <sup>1</sup>	User Associated Distributed Mappings	USER_DISTR_MAPPING
020A	User MFA Factor Data Record	USER_MFA_FACTOR_DATA_RECORD
020B	User MFA Policies Record	USER_MFA_POLICIES_RECORD
0210	User DFP Data	USER_DFP_DATA
0220	User TSO Data	USER_TSO_DATA
0230	User CICS Data	USER_CICS_DATA
0231	User CICS Operation Classes	USER_CICS_OPCLASS
0232	User CICS RSL Keys	USER_CICS_RSLKEY
0233	User CICS TSL Keys	USER_CICS_TSLKEY
0240	User LANGUAGE Data	USER_LANGUAGE_DATA
0250	User OPERPARM Data	USER_OPERPARM_DATA
0251	User OPERPARM Scope	USER_OPERPARM_SCOP
0260	User WORKATTR Data	USER_WORKATTR_DATA
0270	User OMVS Data	USER_OMVS_DATA
0280	User NETVIEW Data	USER_NETV_DATA
0281	User NETVIEW Operation Classes	USER_NETV_OPCLASS
0282	User NETVIEW Domains	USER_NETV_DOMAINS
0290	User DCE Data	USER_DCE_DATA
02A0	User OVM Data	USER_OVM_DATA
02B0	User LNOTES Data	USER_LNOTES_DATA
02C0	User NDS Data	USER_NDS_DATA
02D0	User KERB Data	USER_KERB_DATA
02E0	User PROXY Data	USER_PROXY_DATA
02F0	User EIM Data	USER_EIM_DATA
02G1	User CSDATA Custom fields	USER_CSDATA_CUST
0400	Data Set Basic Data	DS_BD
0401	Data Set Categories	DS_CATEGORIES
0402	Data Set Conditional Access	DS_COND_ACCESS

Table 28. Correlation of record type, record name, and Db2 table name (continued)

<b>Record type</b>	<b>Record name</b>	<b>Db2z/OS table name</b>
0403	Data Set Volumes	DS_VOLUMES
0404	Data Set Access	DS_ACCESS
0405	Data Set Installation Data	DS_INSTALL_DATA
0410	Data Set DFP Data	DS_DFP_DATA
0420	Reserved	—
0421	Data Set TME Role	DS_TME_ROLE
0431	Data Set CSDATA Record	DS_CSDATA_CUST
0500 <sup>2</sup>	General Resource Basic Data	GENR_BD
0501	General Resource Tape Volumes	GENR_TAPE_VOLUMES
0502	General Resource Categories	GENR_CATEGORIES
0503	General Resource Members	GENR_MEMBERS
0504	General Resource Volumes	GENR_VOLUMES
0505	General Resource Access	GENR_ACCESS
0506	General Resource Installation Data	GENR_INSTALL_DATA
0507	General Resource Conditional Access	GENR_COND_ACCESS
0508	General Resource Filter Data	GENR_FILTER_DATA
0509 <sup>3</sup>	General Resource Distributed Identity Mapping Data	GENR_DISTR_MAPPING
0510	General Resource Session Data	GENR_SESSION_DATA
0511	General Resource Session Entities	GENR_SESSION_ENT
0520	General Resource DLF Data	GENR_DLF_DATA
0521	General Resource DLF Job Names	GENR_DLF_JOBNames
0530	General Resource SSIGNON Data Record	GENR_SSIGNON_DATA
0540	General Resource STDATA Data	GENR_STDATA_DATA
0550	General Resource SVFMR Data	GENR_SVFMR_DATA
0560	General Resource Certificate Data	GENR_GRCERT_DATA
0561	General Resource Certificate References Data	GENR_CERTR_DATA
0562	General Resource Key Ring Data	GENR_KEYR_DATA
0570	General Resource TME Data	GENR_TME_DATA
0571	General Resource TME Children	GENR_TME_CHILDREN
0572	General Resource TME Resource	GENR_TME_RESOURCE
0573	General Resource TME Group	GENR_TME_GROUP
0574	General Resource TME Role	GENR_TME_ROLE

Table 28. Correlation of record type, record name, and Db2 table name (continued)

Record type	Record name	Db2z/OS table name
0580	General Resource KERB Data	GENR_KERB_DATA
0590	General Resource PROXY Data	GENR_PROXY_DATA
05A0	General Resource EIM Data	GENR_EIM_DATA
05B0	General Resource ALIAS Data	GENR_ALIAS_DATA
05C0	General Resource CDTINFO Data	GENR_CDTINFO_DATA
05D0	General Resource ICTX Data	GENR_GRICTX_DATA
05E0	General Resource CFDEF Data	GENR_CFDEF_DATA
05F0	General Resource SIGVER Data	GENR_GRSIG_DATA
05G0	General Resource ICSF Data	GENR_ICSF_DATA
05G1	General Resource ICSF Key Label	GENR_ICSF_KEY_DATA
05G2	General Resource ICSF Certificate Identifier	GENR_ICSF_CERT_DATA
05H0	General Resource MFA Factor Definition Record	GENR_MFA_FACTOR_DEFINITION_RECORD
05I0	General Resource MFPOLICY Definition Record	GENR_MFA_DEFINITION_RECORD
05I1	General Resource MFA Policy Factors Record	GENR_MFA_POLICY_FACTORS_RECORD
05J1	General Resource CSDATA Record	GENR_CSDATA_CUST
05L0	General Resource JES Data Record	GENR_JES_DATA
1210	User MFA Factor Tags Data Record	USER_MFA_FACTOR_TAGS_DATA_RECORD
1560	General Resource Certificate Information Record (1560)	GENR_CERT_INFORMATION_RECORD

**Note:** When possible, fields that contain UTF-8 data are translated to EBCDIC using the code page specified in the FACILITY class profile IRR.IDIDMAP.PROFILE.CODEPAGE if defined and contains a valid code page. Otherwise, code page IBM-1047 is used. For more details, see [“Restrictions for UTF-8 data values”](#) on page 677.

1. The USDMAP\_MAP\_NAME field in record type 0209.
2. The GRBD\_NAME field in record type 0500 when GRBD\_CLASS\_NAME is IDIDMAP.
3. The GRDMAP\_NAME and GRDMAP\_DIDREG fields in record type 0509.

## Comparing LISTUSER and LISTGRP output with IRRDBU00

The RACF list commands, such as LISTUSER and LISTGRP, do not necessarily produce equivalent output information as the IRRDBU00 utility. IRRDBU00 creates as output a representation of actual data in the RACF database and does only a minimal amount of interpretation of the data. This differs from some commands, such as LISTUSER and LISTGRP, that interpret the data they find in the RACF database before displaying it. Therefore, you might notice this difference when reviewing the command and IRRDBU00 output related to certain user information, such as password intervals and user revocation status.

See *z/OS Security Server RACF Macros and Interfaces* for the description of each field in the group basic data (type 100) and user basic data (type 200) records.

See *z/OS Security Server RACF Command Language Reference* for information about the output listings for the LISTUSER and LISTGRP commands.

### ***Processing password intervals for protected users***

When you issue the LISTUSER command for a protected user, the PROTECTED attribute is listed and the user's password interval is listed as N/A. (These values reflect that a protected user need not supply a password or password phrase to enter the system.)

Protected users are identified in IRRDBU00 output by the PRO value in the USBD\_NOPWD field of the user basic data (record type 200). However, the PASSINT field for the protected user might contain a password interval value, such as 060. Ignore the contents of the PASSINT field for protected users.

### ***Processing user revocation information***

For users and their connections to groups, RACF stores several pieces of information related to the user's revocation status:

#### ***revoke\_date***

The date from which the user is revoked

#### ***resume\_date***

The date on which the user is no longer revoked

#### ***revoked\_flag***

A flag indicating if the user has been revoked

At logon or job initiation, RACF compares the current date with the *revoke\_date* and *resume\_date*. If the current date falls between them, the logon or job initiation is not allowed or the connection with the group is not considered valid.

LISTUSER and LISTGRP perform similar checks. For example, if the date on which the LISTUSER command is issued falls between the *revoke\_date* and the *resume\_date*, LISTUSER reports that the user is revoked. If the date on which the LISTUSER command is issued does not fall between the *revoke\_date* and the *resume\_date*, LISTUSER indicates that the user is not revoked even if the *revoke\_date*, *resume\_date*, and *revoked\_flag* are set in the RACF database.

**Note:** It is possible to have no data for the *revoke\_date* and *resume\_date*.

Because IRRDBU00 does not have a reference date such as the current date, it cannot interpret the *revoke\_date*, *resume\_date*, and *revoked\_flag* information with a reference date. IRRDBU00 unloads the values as they are specified in the RACF database. This means that if you write a query that just checks the *revoked\_flag*, the results differ from LISTUSER and LISTGRP.

You can incorporate a date check into your queries that performs the same checks as the LISTUSER and LISTGRP commands. [Figure 21 on page 389](#) shows a sample of structured query language (SQL) that does this test for the user revoke status. Note that CURRENT DATE can be replaced with any valid Db2z/OS date value.

```
SELECT * FROM USER01.USER_BD
WHERE
    (CURRENT_DATE >= USBD_REVOKE_DATE AND
      (USBD_RESUME_DATE IS NULL OR
       USBD_RESUME_DATE <= USBD_REVOKE_DATE OR
       USBD_RESUME_DATE > CURRENT_DATE))
OR
    (USBD_REVOKE = 'Y' AND
      (USBD_RESUME_DATE IS NULL OR
       NOT (CURRENT_DATE >= USBD_RESUME_DATE AND
            (USBD_REVOKE_DATE IS NULL OR
             USBD_REVOKE_DATE < USBD_RESUME_DATE OR
             USBD_REVOKE_DATE > CURRENT_DATE))))
```

Figure 21. Sample SQL to process revoke and resume dates

## Database unload utility output samples

A relational database management system such as Db2z/OS can be used with the Query Management Facility (QMF) to create reports.

A report many installations find useful is a list of all of the data set profiles that contain an non-valid ID in the access list. This situation occurs when you delete a user ID or group name without deleting the authorities the ID might have had in the RACF database.

To search the data set access list for user IDs and group names that do not have user or group profiles in the database, perform the following steps:

1. Create a query that compares the entries in the access list with a list of valid user IDs or group names.  
A sample SQL query is provided in [Figure 22 on page 390](#).
2. Format the results of the query as provided by the QMF form in [Figure 23 on page 391](#).

The resulting report is shown in [Figure 24 on page 391](#).

**Note:** If you use the IRRUT100 utility to check the references to a user or group name, IRRUT100 requires that the user or group name be known. The sample query shown here does not have such a requirement. It finds *all* user IDs or group names that are not valid.

When your RACF database is unloaded, the IRRDBU00 utility creates a Data Set Access Record (record type 404) for each user ID or group name in the access list of each data set.

When you load your IRRDBU00 output into Db2z/OS, an AUTH\_IDS table is created that contains the name of every valid user ID and group name.

### SQL query

The sample SQL query compares the ID in the data set access record (DSACC\_AUTH\_ID) with the list of valid user and group names (in AUTH\_IDS). When a user ID is found that is not a valid user ID or group name, it is listed. The query also lists the data set profile name, the authority that the user has, and the access count.

```
-----
-- Description: Check all of the data set standard access lists and --
--               verify that each user ID is a valid user or      --
--               group name                                         --
--                                                                 --
-- Tables Accessed: SQL                                           --
--               "DS_ACCESS"   - A list of data set authorities    --
--               "AUTH_IDS"   - A list of valid users/groups      --
--                                                                 --
-----

SELECT
    DSACC_NAME
    ,DSACC_AUTH_ID
    ,DSACC_ACCESS
    ,DSACC_ACCESS_CNT
FROM
    USER01.DS_ACCESS X
WHERE NOT EXISTS
    ( SELECT *
      FROM
          USER01.AUTH_IDS
        WHERE
          X.DSACC_AUTH_ID=AUTHID_NAME
        )
AND
    X.DSACC_AUTH_ID~='*'
ORDER BY 1
;
```

Figure 22. A sample SQL query

### QMF form

If the SQL query shown in [Figure 22 on page 390](#) is processed using QMF, the data that is returned can be processed into a report. [Figure 23 on page 391](#) shows a report or forms definition. It creates the report

shown in [Figure 24 on page 391](#) entitled "Data Set Profiles With Access Lists Containing User IDs or Group Names That Are Not Valid".

COLUMNS:		Total Width of Report Columns: 80				
NUM	COLUMN HEADING	USAGE	INDENT	WIDTH	EDIT	SEQ
1	DSACC_NAME	BREAK1	2	44	C	1
2	DSACC_AUTH_ID		2	8	C	2
3	DSACC_ACCESS		2	9	C	3
4	DSACC_ACCESS_CNT		2	11	L	4

PAGE:	HEADING	===>	DATA SET PROFILES WITH ACCESS LISTS CONTAINING			
			USER IDS OR GROUP NAMES THAT ARE NOT VALID			
	FOOTING	===>				
FINAL:	TEXT	===>				
BREAK1:	NEW PAGE FOR BREAK?	===>	NO			
	FOOTING	===>				
BREAK2:	NEW PAGE FOR BREAK?	===>	NO			
	FOOTING	===>				
OPTIONS:	OUTLINE?	===>	YES	DEFAULT BREAK TEXT?	===>	NO

Figure 23. A sample QMF form

## Report output

Figure 24 on page 391 shows the report that results from the SQL query shown in [Figure 22 on page 390](#) and the QMF form shown in [Figure 23 on page 391](#). Not all of the resulting rows are shown.

DATA SET PROFILES WITH ACCESS LISTS CONTAINING USER IDS OR GROUP NAMES THAT ARE NOT VALID			
DSACC_NAME	DSACC_AUT	DSACC_ACC	DSACC_ACCES
JAS.WORK.CNTL	WILLIEC	READ	3
	JEFFK	READ	2
	LIEN	READ	2
	DIANE	READ	4
	SLICK	READ	5
	KOFI	READ	2
	CHUCKY	READ	1
	AERON	READ	2
	LING	READ	3
	TONYL	READ	3
	JOES	READ	6
SINEAD.DOC.TEXT	SIMONE	READ	4
	HAN	READ	2
FRANTI.TEST.DATA	MIKEF	READ	10
MARLEY.DESIGNS.DATA	ZIGS	READ	3
HUMAN.*	FATIMA	UPDATE	6
	STING	READ	1

04/31/2004 04:26 PM	PAGE 10
---------------------	---------

Figure 24. A sample report

## Using the RACF remove ID (IRRRID00) utility

The RACF remove ID (IRRRID00) utility can help you keep your RACF database current. You can use this utility to remove all references to group IDs and user IDs that no longer exist in or are about to be removed from the RACF database. Also, you can specify a replacement ID for those IDs that will be removed.

The remove ID utility processes the output of the RACF database unload (IRRDBU00) utility. You need to have read access to this output. To get this output, run the database unload utility against a copy of

the RACF database. See [“Using the RACF database unload utility \(IRRDBU00\)”](#) on page 371 for more information.

The remove ID utility:

- Uses the DFSORT utility (or an equivalent program) to create lists of IDs from the output of the database unload (IRRDBU00) utility or from user input in a SYSIN file.
- Compares these IDs to the user IDs and group names contained in such RACF data fields as:
  - Standard access list
  - Conditional access list
  - Profile names in the FACILITY class and certain general resource member classes
  - OWNER fields
  - NOTIFY fields
  - APPLDATA fields of certain general resource profiles.

**Note:** See [“Finding residual IDs”](#) on page 396 for more information about the fields searched by the remove ID utility.

- Generates as output a TSO/E CLIST consisting of commands that change or remove each reference to residual IDs that no longer exist or to the IDs you specify in the SYSIN data set. For example, the output could include:
  - PERMIT commands to delete references on an access list
  - DELDSD or RDELETE commands to delete all data set and general resource profiles when the profile name contains the reference as a qualifier
  - ALTDSO and RALTER commands to change references to other values when an ID value is required.

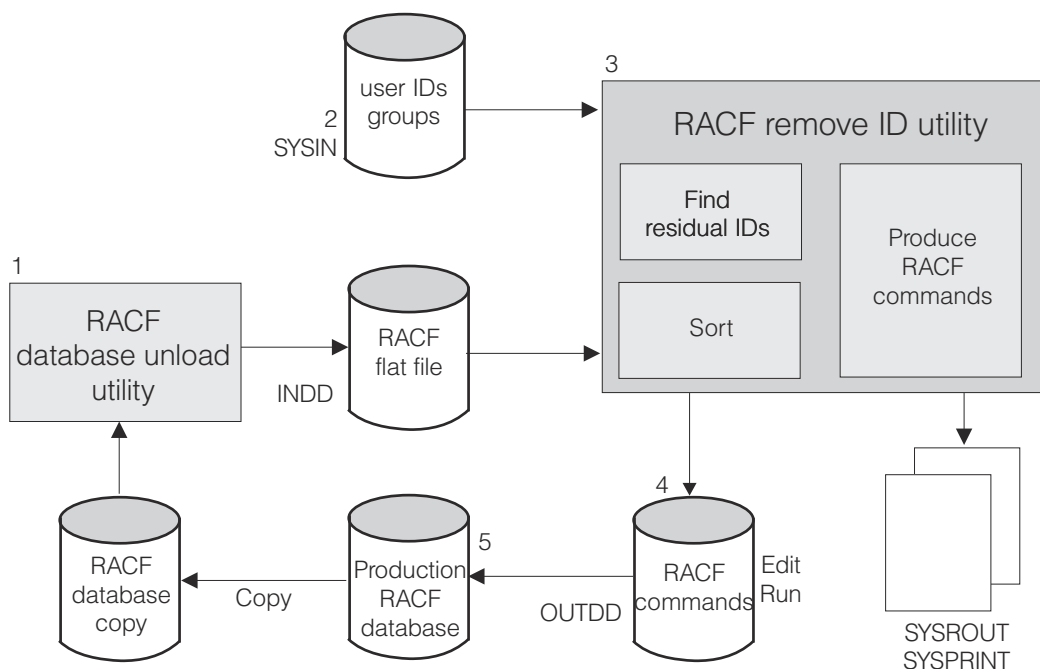


Figure 25. Using the remove ID utility

As shown in [Figure 25](#) on page 392, here's how to use the remove ID utility:

1. Use the database unload utility to produce a *flat file*. This file is the main input to the remove ID utility. You should use a *copy* of the production RACF database as input to the database unload utility.
2. Optionally, you can specify a SYSIN file.



If this file is empty or does not contain any valid input, or if it is allocated as DUMMY, the remove ID utility searches for residual references to user IDs or group names that do not exist as a user profile or a group profile. See [“Running IRRRID00 with an empty SYSIN” on page 397](#) for an example.

3. The remove ID utility does one of the following:
  - a. Finds the residual IDs, sorts them, and then uses this list of IDs to produce output that contains the appropriate RACF commands.  
See [“Finding residual IDs” on page 396](#) for more information about this step.
  - b. Uses a list of user IDs and group names that are specified in the SYSIN file to produce output that contains the appropriate RACF commands.  
See [“Creating commands to remove IDs” on page 397](#) for more information about this step.
4. The remove ID utility creates an OUTDD file, which contains commands to change or remove the occurrences of these IDs.

You should review the commands the remove ID utility generates and, if necessary, edit them.

If you run the remove ID utility with no SYSIN file, or do not specify a replacement ID, the output shows any references to an ID that requires a replacement as *?id*. This might be the case, for example, in places where a residual user ID was the owner of other profiles. You should change all occurrences of *?id*, if any, to an *existing* user ID or group name.

5. As long as you have sufficient authority, you can now run these commands on the production RACF database. See [z/OS Security Server RACF Command Language Reference](#) for the specific authority requirements for RACF commands.

**Note:**

1. The remove ID utility deals with profiles in the RACF database. So, keep in mind that the remove ID utility does not produce any commands to delete, or rename the resources these profiles protect. You must delete, rename, or make sure other profiles protect those resources that were once protected.  
You can use DFSMSdss to rename data sets for IDs that you will be removing from the RACF database.
2. If you delete profiles before you delete or rename the data sets themselves and PROTECTALL is in effect, you might need some extra authority to remove these data sets.
3. If you remove a user ID that had been cross-linked with a DCE principal, contact the cell's DCE administrator to determine whether the DCE principal should be deleted from the cell.
4. If a residual ID is found in a NOTELINK or NDSLINK profile, an RDELETE command will be produced to delete the profile. However, if the profile name contains lower case characters, the RDELETE command cannot be executed successfully. To delete the profile, you must issue an ADDUSER command for the user ID specifying the corresponding LNOTES SNAME or NDS UNAME. Then, a DELUSER can be issued to delete the user profile and the NOTELINK or NDSLINK profile.
5. If a user ID that you specified in the SYSIN file is found in the name of a user profile containing an LNOTES, NDS, or DCE segment, IRRRID00 will produce a DELUSER command to delete the user profile, but it will not produce RDELETE commands to delete the corresponding NOTELINK, NDSLINK, or DCEUIDS profiles. Deletion of the user ID through DELUSER processing will cause the deletion of the corresponding general resource profiles.
6. If a residual user ID or a user ID that you specified in the SYSIN file is found in an IDIDMAP profile, IRRRID00 does not produce an RDELETE command to delete the IDIDMAP profile. Instead, it produces a RACMAP DELMAP command, specifying the user ID and label name of the distributed identity filter contained in the IDIDMAP profile, to delete the filter.

A residual user ID might be found in an IDIDMAP profile if a user ID that is mapped by distributed identity filter is subsequently deleted by issuing a DELUSER command from a downlevel system that does not support distributed identity filters.

**Performance consideration:** When you issue the RACMAP DELMAP command specifying both the label and a user ID that has no user profile (such as a residual user ID), distributed identity filter RACF searches all profiles in the IDIDMAP class to locate and delete all matching filters. This search might take an extended period of time.

## IRRRID00 job control statements

The following job control statements are needed to run the remove ID utility:

### JOB

Initiates the job.

It is recommended that you run IRRRID00 with a region size of 25M:

```
EXEC PGM=IRRRID00,REGION=25M
```

**Note:** The storage required to run IRRRID00 successfully depends on the size of the RACF database and other factors that are controlled by the sort utility your installation uses. If the job does not run because there is not enough storage available, try increasing the region size. If your installation has a large RACF database, a region size of 0M might be required:

```
EXEC PGM=IRRRID00,REGION=0M
```

### EXEC

Specifies the program name (PGM=IRRRID00) or the procedure name if the job control statements are in a procedure library.

### SYSPRINT DD

Defines a sequential message data set for the messages produced by IRRRID00.

### SYSOUT DD

Defines a sequential message data set for the messages produced by the DFSORT utility or its equivalent.

### SORTOUT DD

Defines a work data set that contains final list records. This data set should be approximately the same size as the data set allocated to INDD.

### SYSUT1 DD

Defines a work data set that contains intermediary records. This data set should be approximately the same size as the data set allocated to INDD.

### INDD DD

Defines the sequential input data set that contains the IRRDBU00 output being processed. This statement should refer to the same data set as the OUTDD statement does in the IRRDBU00 job.

### OUTDD DD

Defines the single sequential output data set. The output of IRRRID00 is a set of variable length records that contain the commands needed to delete or alter the references to the IDs. This data set must be allocated as a variable length data set, with a logical record length (LRECL) of at least 259. If a shorter LRECL is supplied, IRRRID00 changes the LRECL to 259.

When IRRRID00 opens the OUTDD data set, it verifies that the block size of the data set is at least 4 greater than the LRECL.

### SYSIN DD

Defines the sequential input data set that contains the list of user IDs or group names to search for. Each ID must be on its own record. The ID can be up to 8 characters in length. It will be truncated if longer than 8 characters.

Optionally, you can specify a replacement ID. The replacement ID must be on the same input record, separated from the original ID by at least 1 blank. The replacement ID can be up to 8 characters in length. It will be truncated if longer than 8 characters.

IRRRID00 accepts both fixed length records (RECFM=F or RECFM=FB) and variable length records (RECFM=V or RECFM=VB) either with or without record numbers. To allow for record numbers, IRRRID00 always ignores columns 1–8 if the SYSIN records are variable length, and ignores the last eight columns if the records are fixed length. In addition, IRRRID00 ignores blank records.

### Note:

1. The SYSIN DD is optional.

If SYSIN is specified, IRRRID00 does not validate the ID being searched for or the replacement ID.

If it is not specified or if it points to a data set that does not contain a list of user IDs or group names, a message is issued to SYSPRINT and a search is performed for all references to IDs that no longer exist.

2. The percent (%) and asterisk (\*) are processed as regular characters; no generic processing will be performed. A period (.) should not be used but will be accepted; a match of IDs will not occur in most cases.
3. Some of the DD names shown are for DFSORT only. If you are using an equivalent product, refer to that product's documentation for the DD names to use.

## Searching for all residual references

To search for all references to IDs that no longer exist, run IRRRID00 with no IDs specified. You can do this either by not allocating the SYSIN DD statement or by allocating it to DUMMY. [Figure 26 on page 395](#) shows the sample JCL used to run RACF remove ID utility.

```
//USER01 JOB Job card...
//CLEANUP EXEC PGM=IRRRID00,REGION=25M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTOUT DD UNIT=SYSALLDA,SPACE=(CYL,(5,5))
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(3,5))
//INDD DD DISP=OLD,DSN=USER01.IRRDBU00.DATA
//OUTDD DD DISP=OLD,DSN=USER01.IRRRID00.CLIST
//SYSIN DD DUMMY
/*
```

Figure 26. Searching for all residual references

## Searching for a list of IDs

IRRRID00 can be used to find specific user IDs and group names. [Figure 27 on page 395](#) shows the sample JCL used to run RACF remove ID utility and search for the IDs MARK, BRUCE, and JUNO.

```
//USER01 JOB Job card...
//CLEANUP EXEC PGM=IRRRID00,REGION=25M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTOUT DD UNIT=SYSALLDA,SPACE=(CYL,(5,5))
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(3,5))
//INDD DD DISP=OLD,DSN=USER01.IRRDBU00.DATA
//OUTDD DD DISP=OLD,DSN=USER01.IRRRID00.CLIST
//SYSIN DD *
MARK
BRUCE
JUNO
/*
```

Figure 27. Searching for specific references

## Specifying a replacement ID

[Figure 28 on page 396](#) shows the sample JCL used to run the RACF remove ID utility and search for the IDs MARK, BRUCE, and JUNO, with a replacement ID for MARK.

```
//USER01 JOB Job card...
//CLEANUP EXEC PGM=IRRRID00,REGION=25M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SORTOUT DD UNIT=SYSALLDA,SPACE=(CYL,(5,5))
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,(3,5))
//INDD DD DISP=OLD,DSN=USER01.IRRDBU00.DATA
//OUTDD DD DISP=OLD,DSN=USER01.IRRRID00.CLIST
//SYSIN DD *
MARK ELVIS
BRUCE
JUNO
/*
```

Figure 28. Specifying a replacement ID

## IRRRID00 return codes

Table 29 on page 396 describes the IRRRID00 return codes. For message explanations, see [RACF remove ID utility \(IRRRID00\) messages in z/OS Security Server RACF Messages and Codes](#).

Table 29. Return codes for the remove ID utility (IRRRID00)

Hex (decimal)	Explanation
<b>0 (0)</b>	Function successful. Output generated.
<b>4 (4)</b>	Function completed. Output is truncated. (See <b>Note</b> .)
<b>10 (16)</b>	Terminating error. Contact IBM service. One of the following occurred: <ul style="list-style-type: none"> <li>• ESTAE error</li> <li>• DBU record error</li> <li>• OPEN error</li> <li>• SORT error</li> <li>• Internal name index error</li> <li>• Internal message error</li> </ul>
<b>14 (20)</b>	Open of SYSPRINT DCB failed. Process ends.
<b>20 (32)</b>	RACF is not enabled. Process ends.

**Note:** For information about truncated output, see [“Lengthy commands” on page 399](#).

## Finding residual IDs

The remove ID (IRRRID00) utility searches the following profile fields for any references to residual user IDs or group names that do not exist as a user profile or a group profile.

**Note:** IRRRID00 searches for IDs within profile names without regard to any generic characters within the profile name.

- Owner
- Superior group
- Subgroup
- Default group
- Connections
- Connection owner
- Notify
- Standard access list
- Conditional access list

- DFP(RESOWNER)
- STUSER
- STGROUP
- GROUPS field of the TME segment for general resource profiles in the ROLE class
- APPLDATA field of general resource profiles in the following classes:
  - DCEUUIDS
  - DIGTCERT
  - DIGTCRIT
  - DIGTNMAP
  - KERBLINK
  - NDSLINK
  - NOTELINK
  - TMEADMIN

**Note:** RDELETE commands created for profiles in the NDSLINK and NOTELINK classes might not be executed successfully if the profile name contains lowercase characters. See [z/OS Security Server RACF System Programmer's Guide](#) for information about recovering from these failures.

- Data set high-level qualifier
- GLOBAL DATASET high-level qualifier
- Profile names in the FACILITY class and certain general resource member classes
- DIDUSER field of general resource profiles in the IDIDMAP class.

A list of user IDs and group names is generated from this processing. This list of IDs is used to create the appropriate RACF commands.

## Running IRRRID00 with an empty SYSIN

You might want to run the remove ID utility regularly, as part of a RACF database cleanup task that you perform periodically, for example. When you run the utility with a SYSIN DD DUMMY statement, the output show all occurrences of residual IDs in your RACF database.

```
//CLEANUP EXEC PGM=IRRRID00,REGION=25M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
:
//INDD DD ...
//OUTDD DD ...
//SYSIN DD DUMMY
```

Figure 29. Running IRRRID00 with an empty SYSIN: Sample input

```
PERMIT 'A.B.**' ID(MARK) DELETE
PERMIT 'UPROC1' CLASS(TSOPROC) ID(MARK) DELETE
PERMIT '12345' CLASS(ACCTNUM) ID(MARK) DELETE
PERMIT 'A.B.**' ID(JUNO) DELETE
PERMIT 'A.B.**' OWNER(?JUNO)
```

Figure 30. Running IRRRID00 with an empty SYSIN: Sample output

As shown in [Figure 30 on page 397](#), IRRRID00 found several references to MARK and JUNO, even though these users did not have a profile in the USER class. IRRRID00 produces the commands you would need to change or remove these references in the various fields.

## Creating commands to remove IDs

A list of IDs from the SYSIN file or from the residual search processing is used to create the appropriate RACF commands. While creating the RACF commands, the remove ID utility searches these fields:

- All fields that were searched in finding residual IDs
- All data set qualifiers
- All general resource qualifiers
- Selected member data

See [“Processing general resource profiles” on page 403](#) for more information about general resources and member data.

## Running IRRRID00 with data in SYSIN

If you run the remove ID utility with *oldid newid* as input, *newid* replaces all references to *oldid* in these fields:

- DFLTGRP
- OWNER
- RESOWNER
- SUPGROUP

The *newid* value must follow the *oldid* value on the same input record, separated from it by at least 1 blank.

In this example, MARK was specified in SYSIN. IRRRID00 now produces only RACF commands relative to this user ID. In addition, when you run IRRRID00 with a list of IDs, delete commands (DELDSD or RDELETE) are created for all data set and general resource profiles that have one of the IDs as a qualifier. The search for IDs within profile names is done without regard to any generic character in the profile name.

```
//CLEANUP EXEC PGM=IRRRID00,REGION=25M
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
.
//INDD DD ...
//OUTDD DD ...
//SYSIN DD *
MARK
/*
```

Figure 31. Running IRRRID00 with data in SYSIN: Sample input

```
ALTDSD 'A.B.**' OWNER(?MARK)
ALTDSD 'A.B.**' NONOTIFY
ALTDSD 'A.B.**' DFP(RESOWNER(?MARK))
PERMIT 'A.B.**' ID(MARK) DELETE
PERMIT 'A.B.**' WHEN(PROGRAM(ABCD)) ID(MARK) DELETE
```

Figure 32. Running IRRRID00 with data in SYSIN: Sample output

## Using IRRRID00 output

The output from the remove ID utility is a set of commands intended to be processed as a TSO/E CLIST. Before you process the CLIST, or extract portions to issue as TSO/E commands, you need to review the IRRRID00 output and in some cases edit the results. (For a sample of IRRRID00 output, see [“Sample output” on page 400.](#))

## Replacement IDs

In some cases, such as when IRRRID00 finds an ID in an access list or in a NOTIFY field, it generates a command to simply remove the ID. In other cases, such as an OWNER field, the ID cannot simply be removed. In these cases, IRRRID00 creates a value that is the replacement ID value. The default replacement ID value is *?id*, where *id* is the ID that is being replaced. For example, if the utility was

searching for the ID MARK, which is the owner of profile IRRDBU00.JCL.\*. IRRRID00 generates this command:

```
ALTDS DA('IRRDBU00.JCL.*') GENERIC OWNER(?MARK)
```

Because ?MARK is not a syntactically valid ID, you cannot run this command. Use the editor of your choice to change ?MARK to the ID you want to own this profile. If MARK owned more than one profile, you can globally change ?MARK to the new ID field for all of its occurrences with a global edit command. For example, to change all of MARK's ownership to ELVIS using ISPF, enter:

```
C ?MARK ELVIS ALL
```

## EXIT commands

In the CLIST, IRRRID00 places the commands that alter existing profiles before the commands that delete profiles. IRRRID00 places an EXIT command between these two sets of commands to cause TSO/E to stop before running the second set of commands. This prevents the deletion of profiles until after you have reviewed the commands generated by IRRRID00. After you review the output, you must remove the EXIT statement to run the delete commands that follow it.

## Ampersand characters

When an ampersand character occurs in a command in IRRRID00 output, the utility inserts a second ampersand to prevent CLIST processing from performing symbolic substitution and allow the CLIST to properly process the command. If you execute the command as a TSO/E command, instead of as a CLIST, remove the second ampersand.

## Lengthy commands

The maximum length for lines of IRRRID00 output is 255 characters. If a command is longer than 255 characters, such as when ADDMEM, DELMEM, and WHEN(CRITERIA) operands contain lengthy values, the utility splits command lines at blank characters into shorter pieces. If a piece is too long to fit on one line, the utility puts a question mark in the first column, the piece is truncated, and if a continuation mark is needed, it appears in column 255.

Before you issue a truncated command, copy the command pieces to a longer length record and add the truncated data to the command image. To locate the truncated data, search your input data set (the IRRDBU00 utility output).

## Sample output

```

/*****
/*
/* The RACF Remove ID Utility (IRRRID00) was executed on
/* 1998-03-23 at 09:00:01.
/*
/*
/* This file contains RACF commands that can be used to
/* identify references to user IDs and group names. Residual
/* references on an access list are deleted with the PERMIT
/* command. For all other references, commands are created to
/* change the reference to another value. The default value
/* is ?id. This allows all references to a particular ID to
/* be easily changed to another value using a text editor.
/*
/*
/* Commands to alter ROLE definitions will be created within
/* comments for informational purposes, though the actual
/* updates should be made from Tivoli. The ROLE will not be
/* updated with a replacement value for a group name.
*****/

/*****
/* The INDD data set has been scanned for all names that do
/* not have a user or group name defined for them in INDD. This
/* list of names has been formatted and sorted into the
/* SORTOUT data set.
*****/

CONNECT BILL GROUP(RACFDEV ) OWNER(?ELVIS )
ALTDSD 'DASDDEF.VCE313S' GENERIC OWNER(?JUNO )
PERMIT D12* CLASS(DASDVOL ) ID(MARK ) DELETE
PERMIT 111111 CLASS(DASDVOL ) ID(BRUCE ) DELETE
PERMIT 222222 CLASS(DASDVOL ) ID(JUNO ) DELETE
RALTER FACILITY IRR.LISTUSER NONOTIFY
RALTER FACILITY IRR.PASSWORD.RESET OWNER(?TERRY )
/* RALTER ROLE DEVELOPMENT TME(DELGROUPS(RACFDEV )) */

/*****
/* The following commands delete profiles. You must review
/* these commands, editing them if necessary, and then remove
/* the EXIT statement to allow the execution of the commands.
*****/

EXIT

RDELETE TMEADMIN TERRY@TWINPEAKS.COM
DELDSD 'D69A.BRUCE.TEXT' VOLUME(TSO018) NOSET
DELDSD 'D69A.MARK.*'
DELUSER BRUCE
DELUSER JUNO
DELUSER MARK
DELGROUP RACFDEV

/*****
/* IRRRID00 has successfully completed
*****/

```

Figure 33. Sample output from the IRRRID00 utility

## Running the output CLIST as a batch job

You can run the CLIST that is generated by IRRRID00 as a batch job. To do this, execute the TSO/E terminal monitor (TMP) program.

For a sample of the JCL statements needed to run the CLIST in batch using TMP, see [Figure 34 on page 400](#).

```

//STEP01 EXEC PGM=IKJEFT01,REGION=0M
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
EXEC 'USER01.IRRRID00.CLIST'
/*

```

Figure 34. Running IRRRID00 CLIST using TMP: Sample JCL statements



## Processing profiles and resources

IRRRID00 creates commands that change the protection of your resources. You should make sure that the resources protected by the RACF profiles that are being altered or deleted have been properly renamed, deleted, or protected by other RACF profiles.

A resource that was protected by a profile that IRRRID00 has deleted is now protected by another less specific profile, your installation's PROTECTALL value (for data sets only), or any installation exits.

When IRRRID00 generates a DELDSD command to remove a profile for a discrete data set, it uses the NOSET operand, which leaves the RACF-indicated bit on in the VTOC.

Any data sets that have a high-level qualifier (HLQ) of a user ID or a group name that no longer exists should be archived or assigned new high-level qualifiers. You should consider renaming the data sets to another HLQ to ensure that they have proper protection and ownership.

DFSMSdss (or equivalent) can be used to delete or rename data sets. With appropriate profiles in the RACF FACILITY class, you can use the ADMIN option on DFSMSdss commands:

- COPY with delete and rename unconditional
- DUMP with delete followed by RESTORE with rename unconditional.

DFSMSdss also provides the following special patch flags, which are effective only when ADMIN is used:

- Changing Default Protection Status During Restore

### **Offset 13**

Turns off the RACF indicator in the volume table of contents

- Bypass Storage and Management Class Authorization Checking During Restore

### **Offset 16**

Bypass failures due to the owner of the resource being a revoked user ID

- Bypass Storage and Management Class Authorization Checking During Copy

### **Offset 3C**

Bypasses failures due to the owner of the resource being a revoked user ID

- Allow COPY with DELETE of RACF Indicated Data Sets and No Discrete Profile

### **Offset 3D**

Requests a warning instead of an error condition

For more information about DFSMSdss commands, patch flags, the ADMIN option, and use of appropriate FACILITY profiles, see [z/OS DFSMSdss Storage Administration](#).

## What IRRRID00 verifies

When doing a residual search for ID values that are no longer valid, IRRRID00 verifies that the ID value is correct in the context that it is used. For example, a NOTIFY field can only have a user ID as its value. If IRRRID00 determines that a NOTIFY field contains a group name, IRRRID00 then searches that IRRDBU00 output for all occurrences of that ID and creates commands to delete those occurrences, allowing you to ensure that the ID has the correct access authorities.

You should always review the SYSPRINT output from IRRRID00 for occurrence of messages IRR68017I and IRR68018I. If these messages are found, the profiles in error should be corrected and IRRRID00 should be rerun.

Because IRRRID00 searches for all occurrences of user IDs and group names that are referenced incorrectly anywhere in the RACF database, you might find IRRRID00 attempting to delete access list entries or profiles that should not be deleted. This might be the case if you erroneously specified a group name for a command operand that requires a user ID, or if a user ID is deleted leaving residual data and the same ID is then used to create a group name.

For example, USER1 has the correct default group of GROUP1 as a result of executing the following commands.

```
AG GROUP1
AU USER1 DFLTGRP(GROUP1)
```

If the following command is then executed, a STARTED profile is defined with GROUP1 erroneously specified as the USER value in the STDATA field. Message IRR52144I is issued warning that GROUP1 is not a user ID. However, the profile is added.

```
RALTER STARTED AJB.* STDATA(USER(GROUP1))
```

When IRRRID00 is executed, the following messages appear in SYSPRINT.

### **IRR68017I**

The ID GROUP1 in the STARTED profile AJB.\* is not correct.

### **IRR68018I**

The record number is 48. The ID value should be a USER profile.

These messages indicate that GROUP1 is used incorrectly. IRRRID00 searches for all references to GROUP1 and creates the following commands to remove those references.

```
/******
/* The INDD data set has been scanned for all names that do
/* not have a user or group id defined for them in INDD.
/* This list of names has been formatted and sorted into
/* the SORTOUT data set.
/******

CONNECT  USER1      GROUP(?GROUP1 )
ALTUSER  USER1      DFLTGRP(?GROUP1 )
CONNECT  ?GROUP1    GROUP(GROUPB )
RALTER   STARTED    AJB.* STDATA( USER(?GROUP1 ))
REMOVE   USER1      GROUP(GROUP1 )

/******
/* The following commands delete profiles. You must review
/* these commands, editing them if necessary, and then
/* remove the EXIT statement to allow the execution
/* of the commands.
/******

EXIT

DELGROUP GROUP1

/******
/* IRRRID00 has successfully completed
/******
```

In this example, the output commands to remove all references to GROUP1 should not be executed. Instead, you can alter the AJB.\* profile to reference an existing valid user ID. Then, when IRRRID00 is rerun, these output commands will not appear.

## Database objects that are not processed

RACF requires that several key RACF objects always exist. IRRRID00 does not create commands that delete these items:

- the user IDs: IBMUSER, irrcerta, irrmulti, and irrsitec
- the group name: SYS1
- connection between IBMUSER and SYS1
- the SECLABEL profiles SYSLOW, SYSHIGH, SYSNONE and SYSMULTI.

## Processing a hierarchy of groups

When IRRRID00 encounters a hierarchy of groups in which a group is a superior group to a second group and both of the groups are being deleted, IRRRID00 temporarily changes the superior group to SYS1 and deletes the groups.

## Processing global profiles

GLOBAL data set profiles are processed like data set profiles. If no SYSIN data is specified, IRRRID00 searches the database looking for potential residual IDs. During this search, general resource profiles are not searched, with the exception of GLOBAL data set profiles. GLOBAL data set profile names are processed just like data set profile names: the HLQs of profiles are examined for residual IDs.

The special qualifier names that are allowed in a GLOBAL data profile, such as &RACUID and &RACGPID, are not considered residual IDs.

## Processing general resource profiles

IRRRID00 does not search the names of general resource profiles that do not contain user ID or group name values. Classes in which profiles names are not searched for ID values include TAPEVOL, DASDVOL, RACGLIST, SECLABEL, and SECDATA classes. Member data in these classes or their GLOBAL equivalents is not searched.

The profile names of VMEVENT and VMXEVENT profiles are searched, but member data in these classes or their GLOBAL equivalents is not searched.

## Processing MEMBER data

IRRRID00 only processes the first 252 characters of data for an ADDMEM or DELMEM operand. Commands with truncated data are preceded by a question mark (?). VMEVENT/VMXEVENT member records are not processed.

## Processing universal groups

If you want to delete a universal group, you should run IRRRID00, specifying the group name, to delete the group and remove all member connections. The DELGROUP command can successfully delete a UNIVERSAL group that has members connected to it because universal group profiles might not contain all connected user IDs in the member list. However, when you delete a universal group, you will receive the ICH05008I informational warning message that the group is a universal group, reminding you to run IRRRID00. For more information about universal groups, see [“Defining large groups with the UNIVERSAL attribute” on page 88](#).

Be sure to execute the resulting REMOVE commands to remove all users from the universal group. If you do not, the profiles of those users will contain residual data regarding the deleted group connection. You should also execute any resulting PERMIT DELETE commands to remove residual entries from access lists that contain the deleted universal group.

## IRRRID00 and Tivoli

The RACF remove ID utility detects profiles in the TMEADMIN class when the input user ID is in the APPLDATA field of the TMEADMIN class profile.

IRRRID00 also finds occurrences of group names in the GROUPS field of the TME segment for general resource profiles in the ROLE class. You should make updates to ROLE profiles by changing the role definition from the Tivoli desktop and distributing the change to the z/OS system. The commands generated by IRRRID00 to remove the group references are commented out in the IRRRID00 output data set. If Tivoli has left a residual group reference in this field, you can uncomment the command and run the output EXEC.

If a replacement group name is specified in the SYSIN data set, IRRRID00 does not generate the command to add the new group name to the GROUPS field in the TME segment. Again, this is because the updates should be performed from the Tivoli desktop.

A change made locally to RACF does not have any effect on resource access due to role membership. If this change is not also made to the Tivoli database, the local RACF modification will be overridden the next time the role is distributed from Tivoli.

## Time required to run IRRRID00

The amount of processing time IRRRID00 requires and how much I/O it performs depends on:

- The size of the database it is processing
- The number of IDs it is searching for
- The number of commands it will create
- Whether it is performing a residual search.

Periodically, IRRRID00 displays the number of IRRDBU00 records it has processed. The message numbers are IRR68019I and IRR68020I. These messages describe the number of records that have been searched (if a residual processing search was specified) and the number of records that have been processed looking for references to IDs.

---

## Chapter 17. The RACF remote sharing facility (RRSF)

This topic describes aspects of the RACF remote sharing facility (RRSF) that security administrators should be aware of.

The RACF remote sharing facility allows RACF to communicate with other MVS systems that use RACF, allowing you to maintain remote RACF databases. RRSF extends the RACF operating environment beyond the traditional single host and shared DASD environments, to an environment made up of RRSF nodes that are capable of communicating with one another. This support provides administration of multiple RACF databases from anywhere in the RRSF network.

Benefits of RRSF support for the security administrator include:

- Administration from anywhere in the RRSF network.

With RRSF, a security administrator logged on to one system in the RRSF network can direct allowed RACF TSO commands to remote RRSF nodes in the RRSF network. Administration of all the RACF systems in the RRSF network can take place from a single point of control.

- User ID associations.

By supporting user ID associations and password synchronization, RRSF gives users with multiple user IDs the option of keeping their user ID passwords automatically synchronized across multiple systems.

- Automatic synchronization of databases. With automatic direction, RACF can keep databases synchronized automatically. When a command or application updates a database, RACF can automatically make the change to other databases.

RACF supports APPC and TCP/IP as network protocols that are used to connect one RRSF node to another. Your programmer determines which protocols are used. Your RRSF network might consist of any combination of APPC and TCP/IP node connections.

Using TCP/IP connections for RRSF nodes provides advantages over APPC such as improved overall security, including the availability of stronger encryption levels.

When your programmer implements TCP/IP for RRSF node connections, you must issue RACF commands to allow TCP/IP communication to take place between RRSF nodes. For instructions, see [“Establishing RACF security for RRSF TCP/IP connections”](#) on page 441.

---

### The RRSF network

The *RRSF network*, the foundation of the RRSF environment, is the structure through which RRSF functions operate. An RRSF network is made up of RRSF nodes. An *RRSF node* is made up of one or more z/OS systems running with active RACF subsystems. [Figure 35 on page 406](#) illustrates an RRSF network.

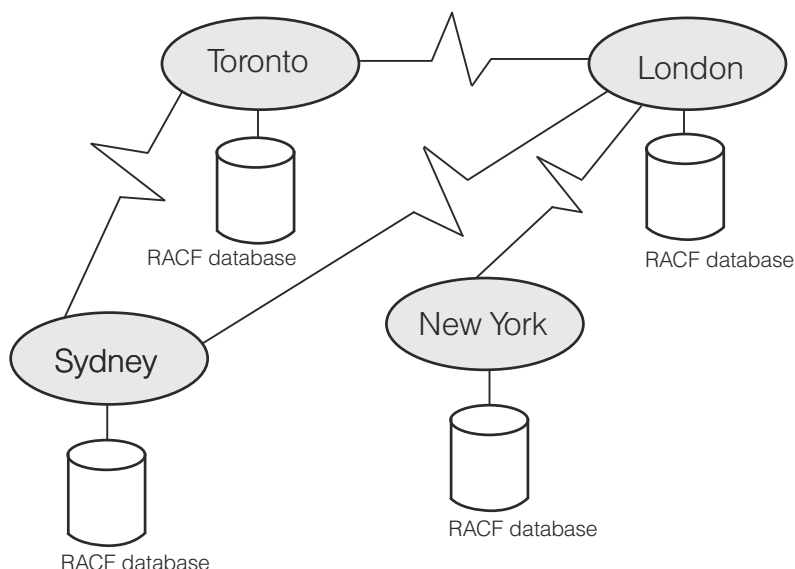


Figure 35. An RRSF network

## RRSF nodes

An RRSF node is an MVS system image that has been defined as an RRSF node to RACF by a TARGET command. For information about defining RRSF nodes with the TARGET command, see [z/OS Security Server RACF System Programmer's Guide](#).

To direct commands or application updates from one MVS system image to another or synchronize passwords between users on two or more MVS system images, both system images must first be defined to RACF as RRSF nodes that can communicate with each other.

### Local and remote RRSF nodes

In an RRSF network, the *local node* is the node whose viewpoint you are speaking from. Its *remote nodes* are the other nodes in the network the local node communicates with. For example, in the network shown in Figure 35 on page 406:

- From the Toronto node's point of view, the Toronto node is the local node and the Sydney and London nodes are remote nodes. The Toronto node cannot communicate with the New York node.
- From the London node's point of view the London node is the local node and the Toronto, Sydney, and New York nodes are remote nodes.
- From the Sydney node's point of view, the Sydney node is the local node and the Toronto and London nodes are remote nodes. The Sydney node cannot communicate with the New York node.
- From the New York node's point of view, the New York node is the local node and the London node is a remote node. The New York node cannot communicate with the Sydney and Toronto nodes.

### Single-system and multisystem RRSF nodes

In an RRSF network, each local or remote node can also be a *single-system node* or a *multisystem node*.

A *single-system RRSF node* is an RRSF node that consists of one MVS system image that does not share its RACF database. For example, in the network shown in Figure 35 on page 406, the Toronto node uses its own RACF database, which it shares with no other system.

A *multisystem RRSF node* is an RRSF node that consists of multiple MVS system images that share the same RACF database. One of the systems is designated as the main system and receives most of the RRSF communications sent to the node. For example, in the network shown in Figure 35 on page 406, the New York node might consist of two systems, Bronx and Brooklyn, that share a RACF database. One of the systems, Bronx, is designated as the main system. This does not affect its status as a local or remote node.

## Operating in local and remote mode

An RRSF node can operate in either local mode or remote mode.

When an RRSF node operates in *local mode*, it cannot communicate with other RRSF nodes. A node operating in local mode provides some remote sharing functions:

- Users with multiple user IDs on the node can synchronize passwords between those user IDs.
- Users with multiple user IDs on the node can direct commands to run under the other user IDs.
- Users can direct commands from their user IDs on the node to the same user ID. This allows you to run commands asynchronously in the RACF subsystem address space.

When an RRSF node operates in *remote mode*, it can communicate with other RRSF nodes. A node operating in remote mode provides all remote sharing features, so you can perform RACF functions across a network.

## Unidirectional or Bidirectional remote connections

A remote connection can allow updates to flow in both directions, or can allow updates to flow in only one direction. When a unidirectional connection is established, one system defines that inbound work requests from the other are denied:

- User ID associations may only be established from the denying node. They may be approved on the denied node.
- Password synchronization flows only from the denying node to the denied node.
- Automatic updates flow only from the denying node to the denied node.
- Commands can only be directed from the denying node to the denied node.
- Output and notifications may be returned from the denied node to the denying node.

If two nodes are denying inbound work from each other, output and notifications may still be sent in either direction.

## Establishing user ID associations in the RRSF network

---

Once the RRSF environment has been configured, users can establish associations (called user ID associations) between two RACF user IDs on nodes in the network. The user ID associations are used to:

- Link (associate) two user profiles on the same or different nodes
- Enable password synchronization to occur
  - Between a user's user IDs on the same node
  - Between a user's user IDs on different nodes.
- Allow a user to manually direct most RACF commands to execute on other user IDs with which the user's user ID has an appropriate association.

A RACF user ID can have multiple user ID associations. User ID associations can be set up by each user or by the security administrator using the RACLINK command. To enable the use of RACLINK, you need to create a profile in the RRSFDATA class. Once you've created the profile, you can restrict it to a subset of users. See [“Controlling access to the RACLINK command” on page 435](#) for more information on enabling the use of the RACLINK command.

## Types of user ID associations

User ID associations can be either peer or managed associations.

A *peer user ID association* occurs between two user IDs and enables the user of either user ID to run allowed RACF commands under the authority of the other using two-way command direction. Password synchronization is allowed in peer associations and these associations can be deleted by either user.

A *managed user ID association* enables the managing user ID to run allowed RACF commands under the authority of the managed user ID using one-way command direction. Users of the managed user ID cannot run RACF commands under the authority of the managing user ID. Password synchronization is not allowed in managed associations and these associations can be deleted by either user.

## Password synchronization

With RRSF, users with multiple user IDs can keep their passwords and password phrases synchronized across RACF databases. Password synchronization (for passwords and password phrases) can be requested between user IDs when a peer user ID association is established with the RACLINK command and the PWSYNC option is specified.

The passwords and password phrases of the user IDs need not to be synchronized at the time the association is requested, nor are they synchronized when the association is established. They are synchronized when either of the associated user IDs initiates a password or password phrase change. The password and password phrase history lists are updated on all systems where the change occurs.

Password synchronization can occur for password and password phrase changes initiated by:

- Logon processing
- The PASSWORD (or PHRASE) command
- The ALTUSER command
- Application programs that use the ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE macro to supply the user's new password or password phrase in clear text form.
- Application programs that use the ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE macro to change:
  - Both the password and the last password change date information, *or*
  - Both the password phrase and the last password phrase change date information.
- Application programs that use the ICHEINTY or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE macro to change the last password or password phrase change date information, not the password or password phrase itself.

**Note:** Password and password phrase changes initiated by the ADDUSER command do *not* result in password synchronization because the new user ID is not yet part of a user ID association.

The security administrator can enable or disable password synchronization for user IDs that have established a peer user ID association with password synchronization requested. See [“Controlling password synchronization”](#) on page 435 for more information.

## Message processing

Messages about the status of a password change and the password synchronization request can be viewed by editing the user's RRSFLIST data set, depending on the option specified on the SET PWSYNC command. See [“Capturing command output”](#) on page 413 for information about output handling for RACF commands that execute in the RACF subsystem address space. TSO end users might receive notification via the TSO SEND command that a password change request has been processed on their behalf by the command output handling components of RRSF.

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



The user who originates the password change is the source user, and the user to whom the source user directs a password change is the target user.

Messages issued by RRSF are returned as specified on the SET PWSYNC command.

The password history of the target user is updated with the user's old password by RRSF password synchronization support.

If the RACF database manager is unable to communicate a password change request to the RACF subsystem address space, message IRR417I is issued to the operator's console via WTO.

## Output capturing

Figure 36 on page 409 shows captured output from a successful password synchronization request.

```

Password synchronization request issued at 15:03:58 on 04/28/98 was
processed at NODE1.TSOUSER on 04/28/98 at 15:04:00
REQUEST ISSUED: From user TSOUSR3 at NODE1 to user TSOUSER at NODE1.

REQUEST OUTPUT:
IRRC013I Password synchronized successfully for TSOUSR3 at NODE1 and
TSOUSER at NODE1.
```

Figure 36. Captured output from a password synchronization request

## User ID associations

Using the RACLINK command, you can define, approve, delete, and list user ID associations.

### Defining user ID associations

You can define these types of user ID associations:

- For your own user ID, with password synchronization
- For your own user ID, without password synchronization
- For other users, with password synchronization
- For other users, without password synchronization
- Managed user ID associations

**Important:** In the following defining user ID sub-topics, if the password or phrase contains special characters that cause problems with TSO/E, the entire string ([node].userid2[/password-or-phrase]) must be enclosed in single quotation marks. Additionally, if the phrase contains blanks, or special characters such as the comma, parenthesis, or comment delimiter (/), the string must be enclosed in quotes. Likewise, when a password or phrase starts with an asterisk, the string must be enclosed in quotes. For example:

```
RACLINK ID(COREY) DEFINE('MVS03.TREVOR/We got your back') PEER(PWSYNC)
```

**Note:** Although, it is possible to define associations to users with 8 character IDs on systems that do not support TSO 8 character user IDs, care must be taken when doing so. Errors may appear on the operator console due to a failed SEND command when the association is defined.

Certain commands directed to 8 character users on systems where 8 character user IDs are not supported or enabled by TSO might fail. Examples include dataset commands (addsd, altdsd, listdsd, deldsd) when the dataset name is not specified in quotes. Also, certain rdefine and ralter commands might fail if the addmem() or delmem() operands contain dataset names that are not specified in quotes.

## Defining user ID associations for your own user ID

To define a user ID association for your own user ID, the syntax of the RACLINK command is:

```
RACLINK DEFINE([node].userid/[password-or-phrase]) type
```

### *With password synchronization*

To define a user ID association with password synchronization between your two user IDs, WILLIE on MVS01 and WONKA on MVS03, enter the following command from user ID WILLIE on MVS01:

```
RACLINK DEFINE('MVS03.WONKA/Ch*c0late') PEER(PWSYNC)
```

### *Without password synchronization*

To define a user ID association without password synchronization between your two user IDs, WILLIE on MVS01 and WONKA on MVS03, enter the following command from user ID WILLIE on MVS01:

```
RACLINK DEFINE('MVS03.WONKA/Ch*c0late') PEER(NOPWSYNC)
```

## Defining user ID associations for other users

To define user ID associations for other users, the syntax of the RACLINK command is:

```
RACLINK ID(userid) DEFINE([node].userid/[password-or-phrase]) type
```

### *With password synchronization*

To define a user ID association with password synchronization between VIOLET on MVS01 and VERUCA on MVS03, enter the following command on MVS01:

```
RACLINK ID(VIOLET) DEFINE(MVS03.VERUCA) PEER(PWSYNC)
```

### *Without password synchronization*

To define a user ID association without password synchronization between VIOLET on MVS01 and VERUCA on MVS03, enter the following command on MVS01:

```
RACLINK ID(VIOLET) DEFINE(MVS03.VERUCA) PEER(NOPWSYNC)
```

## Managed user ID associations

To define a managed user ID association, the syntax of the RACLINK command is:

```
RACLINK DEFINE([node].userid/[password-or-phrase]) MANAGED
```

For example, to define a managed user ID association between VIOLET on MVS01 and VERUCA on MVS03, enter the following command on MVS01:

```
RACLINK ID(VIOLET) DEFINE(MVS03.VERUCA) MANAGED
```

## Approving user ID associations

To approve a pending user ID association, the syntax of the RACLINK command is:

```
RACLINK ID(userid) APPROVE(node.userid)
```

For example, to approve a pending user ID association between VIOLET on MVS01 and VERUCA on MVS03, enter the following command on MVS01:

```
RACLINK ID(VIOLET) APPROVE(MVS03.VERUCA)
```

### Deleting user ID associations

To reject a pending association or to delete an existing association, the syntax of the RACLINK command is:

```
RACLINK ID(userid) UNDEFINE(node.userid)
```

For example, to reject a pending user ID association between VIOLET on MVS01 and VERUCA on MVS03, enter the following command on MVS01:

```
RACLINK ID(VIOLET) UNDEFINE(MVS03.VERUCA)
```

### Listing user ID associations

To list user ID associations, the syntax of the RACLINK command is:

```
RACLINK ID(userid) LIST(node.userid)
```

The default is `RACLINK LIST(*.*)`.

For example, to see all the associations defined for user ID CHARLIE, you could issue the following command:

```
RACLINK ID(CHARLIE) LIST(*.*)
```

and receive the following output:

ASSOCIATION information for user ID CHARLIE on node MVS01  
at 9:27:12 on 04/18/98:

Association Type	Node.userid	Password Sync	Association Status
PEER OF	MVS01.VIOLET	YES	ESTABLISHED
PEER OF	MVS03.VERUCA	YES	ESTABLISHED
PEER OF	MVS01.WILLIE	NO	ESTABLISHED
MANAGER OF	MVS03.MIKE	N/A	ESTABLISHED
MANAGER OF	MVS03.AGLOOP	N/A	ESTABLISHED

Figure 37. `RACLINK ID(userid) LIST(*.*)` output

## Command direction

Command direction can be used to perform security administration for multiple data centers from a central site without submitting batch jobs or logging on to the remote systems when the AT option is specified. Command direction using the AT option is not usually used when the remote databases are kept synchronized. In that case, automatic direction is used. Most RACF TSO commands can be manually directed on the local node or to a remote node within an RRSF network.

Manual command direction extends the user's RACF TSO command execution environment to include any of the RRSF nodes defined as targets of the node the user is logged on to, provided the user has an associated user ID on the target node and the associated user ID has the appropriate authority to run the command.

You can enable the use of command direction by creating a profile in the RRSFDATA class. You can restrict the use of command direction by not creating the profile (so no one can direct commands) or by creating

a profile and restricting access to it. See [“Controlling the use of the AT operand” on page 436](#) for more information on restricting the use of the AT option.

Certain commands directed using the AT or ONLYAT options to 8 character users on systems where 8 characters user IDs are not supported or enabled by TSO may fail. Examples include dataset commands (addsd, altdsd, listdsd, deldsd) when the dataset name is not specified in quotes. Also, certain rdefine and ralter commands may fail if the addmem() or delmem() operands contain dataset names that are not specified in quotes.

## Commands that are not eligible for command direction

The following commands are not eligible for command direction:

- BLKUPD
- IRRDPI00
- RACDCERT
- RACLINK
- RACMAP
- RACF operator commands
- RACF TSO commands, when they are issued as operator commands

## Directing commands using the AT option

Once a peer user ID association is established, either user in the association can use the AT option to direct allowed RACF commands to run under the other user's authority. The commands run in the RACF subsystem address space at the other user's node.

The user specifies as the target node a node in an association and a node the user is allowed to direct to via RRSFDATA profiles. Commands can be directed only to a node with which the user has a RACLINK association. In addition, the user must have access to the *DIRECT.nodename* profile. If this is not true, the command cannot be directed unless the commands can be directed to your own ID on the local node only without any RACLINK association.

The target user ID specified in the AT option becomes the user ID that RRSF uses to determine if the requested command can be executed. That is, the user ID effectively becomes the command issuer at the target node and RRSF checks to see if that user ID has the proper authority to run the requested command.

When the command arrives, RRSF creates a subtask in the RACF subsystem address space for the specified user ID and performs authority checking while processing the requested command.

RACF TSO commands that specify command direction run asynchronously, that is, the command issuer does not wait until the command completes processing, and the command output is *not* automatically displayed at the command issuer's terminal. When the command completes processing, the command issuer might receive a TSO SEND message.

Any command output created via the PUTLINE service is captured by RRSF and saved in the issuing user's RRSFLIST data set.

## Directing commands on the local node

If a request specifies that it is to be handled by the local node, it runs in the RACF subsystem address space on the MVS system the request originates from.

Suppose you are USER2 on NODEC. To make RRSF operating in the RACF subsystem address space on NODEC process your LISTUSER request and run it from within the RACF address space, enter:

```
LISTUSER USER2 AT(NODEC.USER2)
```

This request makes RRSF start a subtask in the NODEC RACF subsystem address space for USER2 and invoke the LISTUSER command processor. The output is captured and put into a data set as described in [“Capturing command output” on page 413](#).

If the target user ID on the local node is the same, no user ID association is needed. You only need a user ID association if the target user ID is different from the issuer's user ID when only one system is involved. Also, you need authority to the `DIRECT.nodename` RRSFDATA profile for the local node.

## Directing commands on a remote node

If a request specifies that it is to be handled by a remote node, the local node sends the request to the target node specified by the AT value. The request is handled by the RRSF at that remote node.

For example, USER1 on NODED could direct a RACF TSO command to run in either the RACF subsystem address space on NODED under his own user ID authority, or direct the request to NODEC to run under the authority of USER2 (because USER1 on NODED has an approved user ID association with USER2 on NODEC.)

The request to have a LISTUSER for USER2 on NODEC issued by USER1 on NODED would look like:

```
LISTUSER USER2 AT(NODEC.USER2)
```

This command causes RRSF on NODED to send the request to NODEC. RRSF running in the RACF subsystem address space on NODEC creates a subtask for USER2 in the RACF subsystem address space and runs the LISTUSER command. The output is captured and sent back to NODED, where RRSF places it into USER1's RRSFLIST data set, as specified in [“Capturing command output” on page 413](#).

## Capturing command output

When you direct a command, the results are returned to you and are appended to the end of your RRSFLIST user data set. If you do not have an RRSFLIST user data set, RRSF allocates one for you and adds the results.

The RRSFLIST user data set name is made up of the user's prefix as specified by the user through the TSO PROFILE command, the user ID, and RRSFLIST. When the prefix and the user ID are the same, the duplicate qualifier is dropped. Thus, the data set name is either *prefix.userid.RRSFLIST* or *userid.RRSFLIST*.

You receive a TSO SEND message when the results are ready for viewing, for example:

```
LISTUSER was successful at node NODEC. Output written to USER2.RRSFLIST.
```

You do not receive a TSO SEND message if you had the TSO PROFILE NOINTERCOM setting in effect when you directed the command.

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

Users are responsible for maintaining their own RRSFLIST data sets. If a user's data set becomes full, or otherwise unusable, RRSF performs one of the following actions:

- If the user does not have READ access to profile RRSFLIST.FAILOVER.TRANSMIT in the RRSFDATA class:
  - RRSF appends the command output to the user's RRSFLIST.OVERFLOW data set. If the user does not have an RRSFLIST.OVERFLOW data set, RRSF allocates one and adds the results.
  - RRSF continues to save all output, if any, to the user's RRSFLIST.OVERFLOW data set for a 20-second time window.

- After 20 seconds, RRSF will revert to saving to the user's RRSFLIST data set. If the RRSFLIST data set is still full or unusable, the process repeats.
- If the user's RRSFLIST.OVERFLOW data set is also full or unusable, RRSF discards all output for a 20-second time window.
- After 20 seconds, RRSF will revert to saving to the user's RRSFLIST data set. If the RRSFLIST data set is still full, the process repeats.
- TSO SEND messages are issued to explain why output is being routed to the user's RRSFLIST.OVERFLOW data set or is being discarded. The message is issued only once per each 20-second time window. The first output that is written to the user's RRSFLIST.OVERFLOW data set is preceded by a message that explains why it was saved there instead of the user's RRSFLIST data set.
- The overflow and discard time windows are independent of one another. It is possible to be outside the overflow window, but still within the discard window. Given the limitation of one error notice per window, this behavior can lead to situations in which a new overflow window is entered when the previous discard window is still in effect. For example:

```
IRRR024I User data set IBMUSER.RRSFLIST is full.
IRRR022I IBMUSER.RRSFLIST is not usable. Switching to IBMUSER.RRSFLIST.OVERFLOW for 20
seconds.
IRRR011I LG was successful at node NODE1. Output was lost.
```

Here, RRSF attempts to write to the user's RRSFLIST data set, discovers that it is full, and triggers the 20-second overflow window. However, a previously triggered 20-second discard window is still in effect. Because RRSF notifies the user only at the initial triggering of the window, no additional message is sent to indicate that the user's RRSFLIST.OVERFLOW data set is also unavailable. This situation leads to the command output ultimately being discarded, as indicated by the IRRR011I message.

- The user's RRSFLIST or RRSFLIST.OVERFLOW data set is considered unusable if it is in use by another program, such as being browsed in ISPF.
- Users should not simultaneously allocate both RRSFLIST and RRSFLIST.OVERFLOW unless they are certain that no output is expected from RRSF.
- If the user has READ access to profile RRSFLIST.FAILOVER.TRANSMIT in the RRSFDATA class, or if the RRSFLIST.FAILOVER.TRANSMIT profile does not exist, note the following:
  - RRSF uses TSO TRANSMIT to send the command output to the user. The output begins with a message that indicates that the user's RRSFLIST data set was full at the time the output was received.
  - If the user's RRSFLIST data set is in use, RRSF waits a brief time and tries again. If the RRSFLIST data set remains in use for a long period, and the user has many output records to receive, RRSF might encounter constraint issues with storage or the INMSG VSAM workspace data set.

Users are responsible for preventing their RRSFLIST data sets from becoming full. It is recommended that users periodically delete data from RRSFLIST when it is no longer needed.

If users want more control over the characteristics of their RRSFLIST data sets, they can allocate their own RRSFLIST data set before RRSF writes any data. The data set must have the following attributes:

#### **Record format (RECFM)**

Fixed block (FB)

#### **Logical record length (LRECL)**

80

#### **Data set organization (DSORG)**

Physical sequential (PS).

The contents of the data that is captured and appended to the RRSFLIST data set varies, but generally it contains:

- A brief description or summary of the event

- A reproduction, but not necessarily an exact replica, of the command issued. Command options that are not specified but defaulted by RACF might be included: security-sensitive data such as passwords or key codes are suppressed.

The command is reproduced up to 255 characters, including the command options that are defaulted by RACF, and is truncated at this point. If it is truncated, the last three characters are replaced by a set of ellipses (...) to indicate that the remaining letters or options of the command are omitted.

- The output produced by the command. This output is truncated after 4096 lines.

The following examples show the format of the captured output that is produced by commands that are running in the RACF subsystem address space. The format of the output that is shown is the same for the user's RRSFLIST or RRSFLIST.OVERFLOW data set, or the TRANSMIT that is issued when the user's data sets are full.

Figure 38 on page 415 shows the format of captured output for a directed LISTGRP command.

```
LG issued at 07:50:39 on 04/21/98 was processed at MVS03.SMITHJ on
04/21/98 at 07:50:41

COMMAND ISSUED: LG          (SYS1)

COMMAND OUTPUT:
INFORMATION FOR GROUP SYS1
SUPERIOR GROUP=NONE          OWNER=SMITHJ
NO INSTALLATION DATA
NO MODEL DATA SET
TERMUACC
NO SUBGROUPS
USER(S)=      ACCESS=      ACCESS COUNT=      UNIVERSAL ACCESS=
  IBMUSER      JOIN      000017      READ
CONNECT ATTRIBUTES=NONE
REVOKE DATE=NONE          RESUME DATE=NONE
```

Figure 38. Captured output from a directed LISTGRP command

Figure 39 on page 415 shows the format of captured output for a directed ADDSD command.

```
ADDSD issued at 11:34:29 on 04/21/98 was processed at MVS02.JWS on
04/21/98 at 11:34:31

COMMAND ISSUED: ADDSD      'JWS.DEV*'

COMMAND OUTPUT:
IRRR008I Command succeeded. There are no messages.
```

Figure 39. Captured output from a directed ADDSD command

All timestamps that are shown in the RRSFLIST data set are initially recorded as Greenwich Mean Time (GMT). These timestamps are meant to show the relative sequence in which the commands were entered and processed. When output or notify information is written into the RRSFLIST data set, these times are converted from GMT into local times. The timestamps are as accurate as possible, but they are not intended to give the exact, precise times of events. In addition, the accuracy of the timestamps depends on how accurately you have set your system clocks.

**Note:** RRSF assumes that either all nodes in the RRSF network have their clocks set to GMT and have appropriate local time offsets in SYS1.Parmlib, or that all nodes have their clock set to local time in the same time zone. Any other configuration can cause errors in the timestamps that are shown in an RRSFLIST data set.

## Directing commands using the ONLYAT option

The following information pertains only to automatic command direction.

Because automatic command direction provides a facility to keep RACF database profiles synchronized between RRSF nodes with respect to RACF TSO commands, you might need to fix a situation that has caused the RACF profiles to become unsynchronized. The ONLYAT option addresses this situation.

The ONLYAT option is restricted to SPECIAL users because it can potentially cause unsynchronized conditions if used improperly. It is a mechanism to direct RACF TSO commands to the same or other nodes in the same manner as the AT option, except that the command is not automatically directed. That is, it runs only on the node it is directed to. The command is processed in the RACF subsystem address space under the authority of the specified user ID provided the following requirements are met:

- Both the command issuer and the target user ID must be SPECIAL.
- If the target user ID is the same as the command issuer (although nodes can be different), no user ID association is required.
- If the target user ID is different from the command issuer, a user ID association between command issuer and target user ID is required. (This prevents a SPECIAL user from unauthorized use of another remote SPECIAL user ID.)

## Order considerations for directed commands and application updates

RACF ensures that RRSF requests directed to a remote node by a given user are executed by the remote node in the same order that they were issued. However, when multiple users are directing commands to a remote node, commands directed by different users might not be received in the order they were issued, and might not be executed in the order they are received. Similarly, if multiple users are taking actions that cause applications to issue update requests that RACF is directing to a remote node, updates directed for different users might not be received in the order they were issued, and might not be executed in the order they are received.

The RACF subsystem address space is a multitasking address space and can run many RRSF requests at the same time. To ensure that the commands issued by a user and the application updates initiated by a user run in the same order they are issued, RACF runs only one command or application update at a time for a given user.

For example, PAT and LAUREN send commands from NODEA to NODEB in the following order:

```
PAT      sends  command_1
LAUREN   sends  command_A
PAT      sends  command_2
PAT      sends  command_3
LAUREN   sends  command_B
```

The commands might be received on NODEB in the following order:

```
command_1
command_2
command_A
command_B
command_3
```

The commands might be executed in the following order:

```
command_1
command_A
command_2
command_B
command_3
```

The commands sent by PAT are received and run in the order that PAT sent them, and the commands sent by LAUREN are received and run in the order that LAUREN sent them, but command\_B is sent after command\_3 from NODEA, and runs before command\_3 on NODEB. This is similar to what happens when an RRSF network is not configured and multiple TSO users issue RACF commands simultaneously. The order in which MVS services the TSO users determines the order in which the commands run.

RACLINK functions have a higher priority than directed commands or application updates so it is possible for a RACLINK command issued after a directed command or application update to take effect before the directed command or application update runs. For example, if you direct a command to a user ID you have an association with, and then issue a RACLINK UNDEFINE to delete the association, the association might be deleted before the directed command runs, causing the directed command to fail.



## Directing commands to incompatible systems

If you direct a command to an incompatible system, you might encounter an error if the directed command includes a keyword that is unknown on *either* the local system or the remote system. For example, if you include a new keyword with a command issued on a higher level system and direct it to a lower level system, the command will fail on the *lower* level system where the keyword is unknown. In addition, if you include a new command keyword with a command issued on a lower level system and attempt to direct it to a higher level system, the command will also fail on the *lower* level system where the keyword is unknown, and the command will not be sent to the higher level system.

Similarly, if you use custom fields and you attempt to direct a command that includes a custom field keyword, the custom field must be defined on *both* the local and remote systems. If the custom field is not defined one of the systems, the command will fail on the system where the custom field is undefined because the custom field keyword is unknown. (For information about custom fields, see [Chapter 26](#), “Defining and using custom fields,” on page 637.)

## Automatic direction

---

Automatic direction is an extension of command direction and password synchronization that allows some administrative tasks and application updates to be automated between RRSF nodes. Automatic direction keeps already synchronized RACF profiles synchronized between two or more remote nodes.

Automatic direction includes:

- Automatic direction of commands, which allows RACF TSO commands that update the RACF database to be automatically directed to remote nodes in order to keep profiles synchronized between the nodes. Commands issued with automatic direction of commands run asynchronously. The results and output from the commands are returned to specified users (not necessarily the command issuer).
- Automatic password direction, which keeps already synchronized RACF user profiles synchronized between two nodes, with respect to RACF passwords and password phrases.
- Automatic direction of application updates, which allows updates made by RACF macros to be propagated to the RACF databases of other systems. Updates to the RACF database can be made using:
  - ICHEINTY
  - RACDEF
  - RACROUTE REQUEST=DEFINE
  - RACROUTE REQUEST=EXTRACT,TYPE=REPLACE
  - RACXTRT

Automatic direction of application updates allows these changes to be automatically sent to selected remote nodes. These updates to remote target nodes take place only after the update has successfully completed on the local node where it is executing and the macro completes with a return code of 0.

**Note:** RACF database updates made by the RACDCERT command are candidates for propagation under the control of automatic direction of application updates, even though the RACDCERT command itself is not eligible for automatic command direction. In this case, the individual updates made by RACDCERT might be successful, and propagated to other nodes, even though the RACDCERT command as a whole might fail.

The essential elements of automatic direction are:

- Activation and deactivation, which are done with SET command options. See “Preparing to use automatic direction” on page 418 and [z/OS Security Server RACF Command Language Reference](#) for more information.
- Controlling which updates get automatically directed to which nodes.
 

Application updates, password changes, and commands can be controlled by RRSFDATA profiles.
- Notification of appropriate users of results and output from automatically directed updates.

An installation must decide who should be notified of results and output from automatically directed commands, application updates, and passwords. This is done with the SET command options OUTPUT and NOTIFY. See [“Output processing” on page 421](#) and [z/OS Security Server RACF Command Language Reference](#) for more information.

**Example:** Automatic direction of commands

Automatic direction of commands works as follows: Suppose NODEA, NODEB, and NODEC have equivalent profiles in the USER and GROUP classes. All RACF TSO commands that affect USER and GROUP profiles can be automatically directed between the nodes. When an ADDUSER command is issued on NODEA, it can be automatically directed to execute on NODEB and NODEC. When a DELGROUP command is issued on NODEC, it can be automatically directed to NODEB and NODEA.

There might also be special situations where automatic direction can be used to facilitate administrative updates to multiple RACF databases. For example, suppose a university has a production MVS system and a test MVS system, each with its own RACF database. At the beginning of each semester, each new student gets a user ID on each MVS system. By temporarily using automatic direction of commands, an administrator can enter ADDUSER commands on the production system and have them automatically directed to the test MVS system.

**Example:** Automatic password direction

Automatic password direction does the following: NODEA, NODEB, and NODEC have equivalent profiles in the USER class. Password and password phrase changes can be automatically directed to RACF user profiles without the need for established RACLINK PEER PWSYNC associations. When a user's password or password phrase changes on NODEA, a password synchronization request can be automatically directed to execute on NODEB and NODEC.

**Example:** Automatic direction of application updates

Automatic direction of application updates does the following: An installation-written application that creates profiles in installation-defined class using RACROUTE REQUEST=DEFINE can execute on NODEA and cause profiles to be created on NODEB and NODEC as well as NODEA.

## Preparing to use automatic direction

1. Check your RACF exits, naming conventions table, templates, and dynamic parse tables.

If you are keeping profiles synchronized between multiple nodes, the preceding items on these nodes should be examined to see if they could cause unsynchronized conditions. See [z/OS Security Server RACF System Programmer's Guide](#) for more information.

2. Decide which updates to automatically direct and which profiles to keep synchronized. Updates can be made by command, password changes, and applications. Decide which ones you want to send to target nodes.

There are many considerations to synchronizing profiles because many profiles in the RACF database are related in important ways. An installation's goal should be to have RACF updates execute with the same results on each node (to minimize error conditions and unsynchronized conditions). Here are some guidelines to help prevent unsynchronized conditions.

**Guidelines:**

- a. Use automatic direction to keep USER and GROUP profiles synchronized:

- The same user IDs and groups need to exist on each node, because updates are automatically directed to the same user IDs.
- The RACF authorities for each user ID authorized to automatically direct updates on a node should be equivalent to that same user ID's authorities on each node. This includes user authorities (such as SPECIAL and CLAUTH) and group authorities (which users are connected to which groups).

For example, if SYSPROG on NODEA is SPECIAL and automatically directs commands to SYSPROG on NODEB, SYSPROG on NODEB should have SPECIAL authority also.

- For automatic direction of commands:
    - If commands such as PERMIT or other commands that specify a group name or a user ID as an operand are being kept synchronized, the specified group names or user IDs must exist on both nodes.
    - If the CONNECT command is automatically directed, the GROUP class should be kept synchronized. Otherwise, the automatically directed CONNECT commands are likely to fail.
  - If the USER class is being kept synchronized, it should be noted that the RACLINK command is not eligible for automatic direction. Because the RACLINK command adds information to the USER profiles, an installation should consider whether to manually establish the user ID associations on each node; otherwise the USER profiles do not remain synchronized.
  - If the USER class is kept synchronized and you use a distributed identity filter, be aware that the RACMAP command is not eligible for automatic direction. Because the RACMAP command adds information to USER profiles and affects profiles in the IDIDMAP class, you should implement automatic direction of application updates. For details, see [“RRSF considerations for distributed identity filters” on page 432](#).
  - If the USER or GROUP class is kept synchronized and you use CSDATA segments in USER or GROUP profiles to store data in custom fields, you should also synchronize profiles in the CFIELD class. (For information about custom fields, see [Chapter 26, “Defining and using custom fields,” on page 637](#) and [“RRSF considerations for custom fields” on page 653](#).)
- b. Synchronize profiles by class rather than by command:
- If RDEFINE DASDVOL is automatically directed and RALTER DASDVOL is not, the DASDVOL profile becomes unsynchronized.
  - If ADDSD is automatically directed, but PERMIT DATASET is not, the DATASET profile becomes unsynchronized.
  - If RDEFINE TAPEVOL is automatically directed, but application updates for the TAPEVOL class are not, the TAPEVOL profiles become unsynchronized.
- c. Keep SETROPTS command settings synchronized:
- This can be done manually (using command direction) if SETROPTS is issued infrequently.
  - Make sure that general resource classes on both nodes are active if updates are being automatically directed for the class.
- d. Considerations for running the IRRRID00 utility:
- Because the DELUSER and DELGROUP commands do not automatically delete access list entries from resource profiles, you can use IRRRID00 to generate commands to clean up the profiles for a user or group that has been deleted. If the PERMIT command is being automatically directed, the user who runs the generated list of commands from IRRRID00 must be authorized to automatically direct the PERMIT commands. Otherwise, the resource profiles on the remote system will not be cleaned up.
- e. When synchronizing a particular kind of profile between nodes, it is better to give all users who can update the profiles controlling those profiles the authority to direct updates automatically. To do this, you can give them READ authority to the appropriate RRSFDATA profiles, for example.
- If there are users who can cause updates to occur locally but cannot direct them automatically, some updates are not automatically directed, and the profiles do not remain synchronized. There are special considerations for automatic direction of application updates for the DATASET class. For these, see [“Considerations for the DATASET class” on page 431](#).
3. Synchronize specified profiles:
- Run IRRRID00 on each RACF database to remove references to user IDs and group names that no longer exist.
  - Synchronize databases. You can do this by manually copying the database from one system to the other, or by using a tool that can help you synchronize databases. For information on how to get this

tool and others from the RACF home page or via anonymous FTP, see [“Internet sources”](#) on page xxviii.

4. Create AUTODIRECT profiles in the RRSFDATA class as desired to specify which updates should be automatically directed. Remember to create the profiles on each node from which automatic direction should occur. For example, the following commands cause all password and password phrase updates, commands, and application updates for the user class to be automatically directed to all remote nodes:

```
SETOPTS GENERIC(RRSFDATA)    (required if you use generic profiles)
RDEFINE RRSFDATA AUTODIRECT.*.USER.* UACC(READ)
```

For automatic password direction only, RDEFINE commands are:

```
RDEFINE RRSFDATA AUTODIRECT.*.USER.PWSYNC UACC(READ) (for passwords)
RDEFINE RRSFDATA AUTODIRECT.*.USER.PHRSSYNC UACC(READ) (for password phrases)
```

**Note:** The RRSFDATA profile PWSYNC.*nodename* is not required for using automatic password direction.

5. Activate the RRSFDATA class on each node, if it has not already been activated. It is recommended that the class be RACLISTed for performance reasons. If the class has already been RACLISTed, refresh the profiles. For example, use one of the following commands:

```
SETOPTS CLASSACT(RRSFDATA) RACLIST(RRSFDATA)
SETOPTS RACLIST(RRSFDATA) REFRESH
```

6. Decide who should be notified of the automatic direction results and output.
7. Activate automatic direction when you are ready by issuing the SET command with the options corresponding to automatic command direction, automatic password direction, and automatic direction of application updates, with output and notification options appropriate to your installation's needs:

```
SET AUTODIRECT(...)
SET AUTOPWD(...)
SET AUTOAPPL(...)
```

**Guideline:** Use a RACF parameter library as specified for RACF subsystem initialization. The SET command should be in the default IRROPTxx member of this library so that these options are reinstated when the subsystem is restarted or the entire system is reIPLed.

8. At a later date, to temporarily or permanently deactivate one or more automatic direction functions, issue the SET command with the options corresponding to automatic command direction, automatic password direction, and automatic direction of application updates:

```
SET NOAUTODIRECT
SET NOAUTOPWD
SET NOAUTOAPPL
```

See the note in Step [“7”](#) on page 420 regarding the RACF parameter library.

9. It is possible to fail while attempting to execute a command issued on an uplevel system and manually or automatically directed to a downlevel system through RACF remote sharing. This can occur if the command references a class unknown to the target system (class descriptor tables are different), if it references a segment or field unknown to the target system (templates or dynamic parse definition are different), if it uses a command keyword unknown to the target (dynamic parse definitions or command processor code is different), or if it specifies a profile or member name that is unacceptable to the target system (class descriptor tables have different syntax requirements for profile name length or syntax).

**Guideline:** Be sure that your class descriptor tables, including dynamic classes, are compatible across all systems participating in automatic direction.

If an unsynchronized condition occurs while using automatic command direction, a RACF TSO command can be directed with the ONLYAT option to fix the condition. The command runs on the node specified on the ONLYAT option and is not propagated to any other node. (Note that if the AT keyword is used, the command can be propagated by automatic command direction to other nodes.) For information on the ONLYAT option, see [“Directing commands using the ONLYAT option” on page 415](#). For a complete list of RACF commands that are eligible for automatic command direction, see [z/OS Security Server RACF Command Language Reference](#).

## Output processing

For automatic direction, the installation has many options concerning where the output and notification information resulting from an RRSF request can be sent. This flexibility is provided by operands of the SET command. For example, some installations might choose to notify only one or two administrators when errors are encountered during automatic direction.

The OUTPUT operand specifies that the output from automatic direction should be put in the RRSFLIST data set for the specified users. If the output cannot be put in the RRSFLIST data set for some reason, the output is transmitted to the users or saved to the RRSFLIST.OVERFLOW data set. The output usually contains messages issued during execution, such as informational, warning, or error messages. After RACF determines the result of execution, it sends back either a message saying it succeeded or a message giving diagnostic information.

The NOTIFY operand specifies that TSO SEND commands are issued to the specified users with the results of automatically directed updates. The information sent indicates whether the update was successful or unsuccessful, but does not include other details about the execution.

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

On both the OUTPUT and NOTIFY operands, you can specify whether the output or messages should be sent for all or some updates. The ALWAYS option specifies that results or output from all automatically directed updates are returned to the specified users. This should be used if the users are interested in the results of every automatically directed update.

Output includes informational, warning, and error messages. For automatic direction of commands:

- WARN specifies that results or output from an automatically directed update are returned to the specified users only when the return code from the update is at least four. This should be used if the users are interested in those automatically directed updates that complete with error conditions or warning conditions.
- FAIL specifies that results or output from an automatically directed update be returned to the specified users only when the return code from the update is at least 8. This should be used if the users are interested in only those automatically directed updates which complete with error conditions.

For automatic direction of application updates and passwords:

- WARN and FAIL have the same meaning. Either WARN or FAIL result in returned output or notification when a directed application update or password fails to completely take effect on a remote node.

If an automatically directed update error occurs, the RACF profiles become unsynchronized. An installation should specify at least one person (probably an administrator who is familiar with required profiles) on the OUTPUT and NOTIFY options to receive error results and output (FAIL option). If the initial values of NOOUTPUT and NONOTIFY are used, no one is notified when automatic direction errors occur. Also, once an update is automatically directed to a remote node, all output and messages associated with that update are discarded.

For more information and examples on using the SET command to specify options for automatic direction of updates, see [z/OS Security Server RACF Command Language Reference](#).

## Effects of using OUTPUT and NOTIFY

After the OUTPUT and NOTIFY operands have been specified on the SET AUTODIRECT command, they are used in the processing for automatic direction in the following ways.

Suppose for the examples that the following commands are in effect on NODE1:

```
SET AUTODIRECT (OUTPUT(WARN(NODE1.ANN)) NOTIFY(FAIL(NODE1.ANN)))
SET AUTOPWD (OUTPUT(WARN(NODE1.ANN)) NOTIFY(FAIL(NODE1.ANN)))
SET AUTOAPPL (OUTPUT(WARN(NODE1.ANN)) NOTIFY(FAIL(NODE1.ANN)))
```

and the following commands are in effect on NODE2:

```
SET AUTODIRECT (OUTPUT(WARN(NODE2.SAM)) NOTIFY(FAIL(NODE2.SAM)))
SET AUTOPWD (OUTPUT(WARN(NODE2.SAM)) NOTIFY(FAIL(NODE2.SAM)))
SET AUTOAPPL (OUTPUT(WARN(NODE2.SAM)) NOTIFY(FAIL(NODE2.SAM)))
```

1. When an update has been automatically directed to a remote node and execution is completed, the OUTPUT and NOTIFY settings are used on the node at which the automatically directed update executes. The OUTPUT and NOTIFY settings on the node at which the update originally executed are not relevant.

For example, a user on NODE1 runs an application that performs an update, and the update is automatically directed to NODE2. On NODE2, the update runs with a return code of 8. SAM on NODE2 gets a NOTIFY message and the OUTPUT containing the error messages. Even though the update originally executed on NODE1, the OUTPUT and NOTIFY settings on NODE2 are used instead of the OUTPUT and NOTIFY settings on NODE1.

The update has successfully executed on one node and is automatically directed to another node. The NOTIFY and OUTPUT settings on that second node determine who is notified of the update completion.

2. When an error occurs while attempting to automatically direct an update to a remote node, the OUTPUT and NOTIFY settings are used on the node at which the update originally executes.

For example, a user on NODE1 runs an application that performs an update, and the update is to be automatically directed to NODE2. As the update is placed in the OUTMSG workspace data set (which contains work items to be sent to NODE2), the data set becomes full; therefore, the update is not automatically directed to NODE2. ANN on NODE1 gets a NOTIFY message and the OUTPUT containing the error messages.

The update has successfully executed on one node and is to be automatically directed to another node. An error occurs before the request arrives at the other node. The NOTIFY and OUTPUT settings on the first node determine who is notified of the error. This is treated as an error case (like an update which executed with return code 8), so any user with the FAIL, WARN, or ALWAYS setting would be notified or would receive output.

### Guidelines the use of OUTPUT and NOTIFY

To make sure that the appropriate users are notified or receive output, specify the same users on the OUTPUT and NOTIFY operands on the SET command issued on each node with automatic direction active.

The OUTPUT and NOTIFY lists should include users from at least 2 different nodes, if possible. This is important in the case of a workspace data set problem. For example, suppose a command executes successfully on NODE1 but cannot be sent to NODE2. If all the NOTIFY and OUTPUT users are also on NODE2, it is possible that RRSF might experience the same problem trying to notify the users as it experienced trying to send the command. By also specifying a user on NODE1 or by specifying one local user on each node to get the output, you ensure that RRSF can find someone to notify.

## Output data set names for automatic direction

When output from automatic direction is returned to the command issuer (&RACUID was specified on the OUTPUT operand), the RRSFLIST data set is allocated exactly as it is for directed commands, either *prefix.userid.RRSFLIST* or *userid.RRSFLIST*. The user's prefix is the one in effect at the time that

the RACF update was originally issued (as specified by the user through the TSO PROFILE command). Avoid sending application update output to &RACUID.

When output from automatic direction is returned to a user other than the issuer, the output is placed into a data set named either *prefix.userid.RRSFLIST* or *userid.RRSFLIST*. However, the prefix that is used is taken from the user's TSO segment at the time the output is returned. This prefix is the same as the prefix specified by the user through the TSO PROFILE command, except if the user is logged on when the output is received and the user has changed their TSO prefix during that logon session.

If the automatically directed command originated from the operator's console (even if &RACUID is specified on the OUTPUT operand), the TSO prefix is also taken from the user's TSO segment at the time the output is returned. The same is true for automatic direction of application updates that originate from batch jobs, started procedures, and other non-TSO environments.

The preceding discussion also applies to the RRSFLIST.OVERFLOW data set that might be allocated if the RRSFLIST data set cannot be used.

Application update output should not be sent to &RACUID.

## Notify messages for automatic direction

The TSO SEND messages for automatic direction are the same messages used for command direction.

If the NOTIFY operand specifies &RACUID, and the command issuer has the TSO PROFILE NOINTERCOM setting in effect at the time the command is issued, the command issuer does not receive a TSO SEND message. This is similar to processing for a directed command.

For automatic direction of application updates the messages are similar except that they substitute, for example, Application update has completed successfully... for Command succeeded.

## Sample output from automatic direction

Some sample output from automatic command direction follows:

```
ADDUSER issued at 10:42:33 on 4/1/98 was processed at NODE1.LAURIE
on 4/1/98 at 10:43:45
Command was propagated by automatic direction from NODE2.LAURIE
```

```
COMMAND ISSUED: ADDUSER (ANDREW) PASSWORD()
NAME('#####') AUTHORITY(USE) NOSPECIAL UACC(NONE)
NOOPERATIONS NOADSP NOGRPACC NOAUDIT
```

```
COMMAND OUTPUT:
IRRR008I Command succeeded. There are no messages.
```

```
RDEFINE issued at 12:33:41 on 4/1/98 was processed at NODE1.LAURIE
on 4/1/98 at 12:35:02
Command was propagated by automatic direction from NODE2.LAURIE
```

```
COMMAND ISSUED: RDEFINE RRSFDATA AUTODIRECT.** UACC(NONE)
```

```
COMMAND OUTPUT:
ICH10102I AUTODIRECT.** ALREADY DEFINED TO CLASS RRSFDATA.
```

When errors occur during automatic direction of commands, the command output appropriately reflects what happened.

If an unexpected error occurred while trying to automatically direct a command to a remote node, the output might be similar to the following:

```
RDEFINE issued at 12:33:41 on 4/1/98 was *not* processed at NODEA.LAURIE
Command was *not* propagated by automatic direction from NODEB.LAURIE
```

```
COMMAND ISSUED: RDEFINE RRSFDATA AUTODIRECT.** UACC(NONE)
```

```
ERROR INFORMATION:
IRRR016I Command was not sent. Processing code is 502.
```

If an unexpected error occurs after an automatically directed command arrives at a remote node, the output might look as follows:

```
RDEFINE issued at 12:33:41 on 4/1/98 was *not* processed at NODEA.LAURIE
Command was propagated by automatic direction from NODEB.LAURIE

COMMAND ISSUED:  RDEFINE RRSFDATA AUTODIRECT.**  UACC(NONE)

ERROR INFORMATION:
IRRC010I UNABLE TO ESTABLISH RACF ENVIRONMENT FOR COMMAND RDEFINE.
IRRC012I TARGET USER ID NODEA.LAURIE DOES NOT EXIST.
```

All time stamps shown in the RRSFLIST data set are initially recorded as Greenwich Mean Time (GMT). These time stamps are meant to show the relative sequence in which the commands were entered and processed. When output or notify information is written into the RRSFLIST data set, these times are converted from GMT into local times. The time stamps are as accurate as possible, but they are not intended to give the exact, precise times of events. In addition, the accuracy of the time stamps depends on how accurately you have set your system clocks.

**Note:** RRSF assumes that either all nodes in the RRSF network have their clocks set to GMT and have appropriate local time offsets in SYS1.PARMLIB, or that all nodes have their clock set to local time in the same time zone. Any other configuration will cause errors in the timestamps shown in an RRSFLIST data set.

Some sample output from automatic direction of application updates follows.

Sample output for a successful RACDEF request:

```
Application update request issued at 15:42:33 on 4/1/98
was processed at NODE2.RACFU01
on 4/1/98 at 15:43:45
Request was propagated by automatic direction from NODE1.RACFU01

REQUEST ISSUED: RACDEF TYPE=DEFINE, NEWNAME FROM NODE1.RACFU01

REQUEST OUTPUT:
IRRR101I Application update request completed successfully
        for class DATASET, profile name MYNEW.PROFILE.
```

**Note:** NEWNAME is for DATASET or FILE class only (RENAMES).

Sample output for a successful ICHEINTY request:

```
Application update request issued at 15:51:33 on 4/11/98
was processed at NODE2.RACFU01
on 4/11/98 at 15:53:45
Request was propagated by automatic direction from NODE1.RACFU01

REQUEST ISSUED: ICHEINTY ALTER operation from NODE1.RACFU01

REQUEST OUTPUT:
IRRR101I Application update request completed successfully
        for class $MYCLASS, profile name MYNEW.PROFILE.
```

Sample output for a successful RACROUTE request:

```
Application update request issued at 15:51:33 on 4/13/98
was processed at NODE2.RACFU01
on 4/13/98 at 15:53:45
Request was propagated by automatic direction from NODE1.RACFU01

REQUEST ISSUED: RACROUTE REQUEST=EXTRACT from NODE1.RACFU01

REQUEST OUTPUT:
IRRR101I Application update request completed successfully
        for class $MYCLASS, profile name MYNEW.PROFILE.
```

If a request is unsuccessful or if an abend occurs because of the status of the RACF database on the target, one of three messages is issued:



- IRRR102I if the request type was an ICHEINTY or RACDEF or RACXTRT
- IRRR103I if the request type was a RACROUTE
- IRRR104I if an abend occurred processing a RACROUTE, ICHEINTY, RACDEF, or RACXTRT.

Password synchronization requests initiated by automatic password direction return the following output:

```

Password synchronization request issued at 15:03:58 on 04/18/98 was
processed at NODE2.TSOUSER on 04/18/98 at 15:04:00
Request was propagated by automatic direction from NODE1.TSOUSER

REQUEST ISSUED: From user TSOUSER at NODE1

REQUEST OUTPUT:
IRRC013I Password synchronized successfully for TSOUSER at NODE2 and
TSOUSER at NODE1.
```

The general output format of error messages is the same.

## Interactions among automatic direction functions and password synchronization

Each of the automatic direction functions is controlled independently of the other two. Details of these types of interactions follow:

- Between automatic direction of commands and automatic direction of application updates
- Between automatic password direction and automatic direction of application updates
- Among password synchronization, automatic direction of commands, and automatic password direction

### Interaction between automatic direction of commands and automatic direction of application updates

These two automatic directions work independently of each other. If automatic direction of commands is active and automatic direction of application updates is not active, only RACF command updates are propagated to remote nodes. If automatic direction of application updates is active and automatic direction of commands is not active, only application updates are propagated to remote nodes. If both automatic direction of application updates and automatic direction of commands are active, application updates and commands are propagated to remote nodes.

### Interaction between automatic password direction and automatic direction of application updates

Automatic password direction propagates user password and password phrase updates. Automatic direction of application updates does *not* propagate user password and password phrase changes. If updates to fields unrelated to the password (or password phrase) are made with the same ICHEINTY macro execution that updates the password (or password phrase), the propagation of the unrelated fields is controlled by automatic direction of application updates.

A single ICHEINTY macro TYPE= 'USR' with ACTIONS= that specifies both password and non-password user information will result in the propagation of two requests to the target node: one request (to define the user) is propagated by automatic direction of application updates, and the other (to specify password information for the same user) is propagated by automatic password direction. Requests propagated by automatic direction of application updates execute at the target node using the authority of the user ID associated with the application that issued the ICHEINTY to define the user. Requests propagated by automatic password direction execute at the target node using the authority of the user whose password information is to be changed. Because these two requests execute using the authority of different user IDs, they can execute concurrently with unpredictable results.

Unpredictable results might occur with propagation of password and non-password user information through any combination of ICHEINTY macro executions, such as a program executing a single

ICHEINTY, or multiple ICHEINTY executions within the same or different programs. For this reason, the recommended methods for defining RACF users are:

1. Execute the ADDUSER command
2. Invoke the R\_admin callable service from an application program

Automatic password direction can be used to propagate a password update for a user only when that user is defined to RACF on both the source and target nodes.

## Interaction among password synchronization, automatic direction of commands, and automatic password direction

Password synchronization, automatic direction of commands, and automatic password direction can be active at the same time. These functions interact as follows:

- When a password or password phrase change is made at logon:
  - Password synchronization sends the change to users with approved PEER PWSYNC associations, if the user changing the password or password phrase is authorized to the appropriate password synchronization resource.
  - If automatic password direction is enabled on the system, and the user who changed the password or password phrase is authorized to one or more profiles, the change is automatically directed to the same user IDs on the nodes defined by the RRSFDATA profiles that protect automatic password direction.
- A password or password phrase changed by the PASSWORD command is *not* directed by automatic password direction. It is directed by automatic direction of commands based on RRSFDATA profile setup. Also, if the user is authorized to the appropriate password synchronization resource, password and password phrase changes are sent to users with approved PEER PWSYNC associations with the user whose password or password phrase was changed.
- Password synchronization and automatic password direction do *not* handle updates initiated by the ADDUSER command. Users who participate in password synchronization must be initially defined to RACF before automatic direction can occur. The ADDUSER command can be directed by automatic direction of commands based on RRSFDATA profile setup.
- The ALTUSER and PASSWORD commands must be automatically directed to maintain the synchronization of user passwords and password phrases for the same user IDs across RRSF nodes. The automatic direction of commands RRSFDATA profiles that protect AUTODIRECT . node . USER . ALTUSER and AUTODIRECT . node . USER . PASSWORD control this automatic direction. The RRSFDATA profiles that protect automatic password direction are not checked for the automatic direction of the ALTUSER and PASSWORD commands.
- A password or password phrase change by other methods, such as at logon or by an installation-written application, is *not* directed by automatic direction of commands. These changes are sent to users with the same name if automatic password direction is in effect. Also, if the user is authorized to the appropriate profile in the RRSFDATA class, the changes are sent to users with approved PEER PWSYNC associations with the user whose password or password phrase was changed.

## RRSF considerations for mixed-case passwords

The following rules apply when RRSF nodes do not have the same mixed-case option of the SETROPTS PASSWORD command in effect:

### Rules:

1. When passwords are propagated from a system with MIXEDCASE in effect to a system with NOMIXEDCASE in effect, the resulting passwords are translated to uppercase on the target system.
2. When passwords are propagated from a system with NOMIXEDCASE in effect to a system with MIXEDCASE in effect, the results differ depending on the type of propagation:
  - With password synchronization or automatic password direction, when an application action such as TSO LOGON translates the new password that is entered by a user to uppercase before the

application passes it to RACF, the resulting password is in uppercase on both the propagating and target systems. However, if an application uses ICHEINTY to set a lowercase password on the system with NOMIXEDCASE in effect, the password is in lowercase on both systems.

- With command direction or automatic command direction, commands are sent to the target systems as they were entered by the user. If a lowercase password is entered, it is translated to uppercase on the NOMIXEDCASE system but the command propagates with a lowercase password on the MIXEDCASE system.

#### **Guideline:**

You can administer your RRSF network from any system. However, instruct users to change their passwords from a system with MIXEDCASE in effect to more easily maintain their mixed-case passwords.

For more information, see [“Allowing mixed-case passwords \(PASSWORD option\)” on page 106](#).

## **Using automatic direction of commands**

Automatic direction of commands has unique features. These are listed as follows.

### **Commands not eligible for automatic direction of commands**

In general, commands that are ineligible for command direction are also ineligible for automatic direction of commands. Also, commands that do not update the RACF database are ineligible for automatic direction of commands. The following commands are not eligible for automatic direction of commands:

- BLKUPD
- DISPLAY
- IRRDPI00
- LISTDSD
- LISTGRP
- LISTUSER
- RACDCERT
- RACLINK
- RACMAP
- RESTART
- RLIST
- RVARV
- SEARCH
- SET
- SETROPTS LIST (with no other operands specified)
- SIGNOFF
- STOP
- TARGET

RACF commands can be automatically directed when they are issued in any way except from the RACF parameter library. For example, if they are issued from a TSO session, from a batch job, or even as MVS operator commands, they are still eligible for automatic direction of commands.

### **How automatic direction of commands works**

Automatic direction of commands:

- Can take place only if SET AUTODIRECT has been used to activate it.
- Does not use or require user ID associations.

- Requires proper authority as defined in the RRSFDATA class.
- Begins after a command has successfully completed with a return code that is less than 8.
- Is relative to where a command runs, as opposed to where it originated.
- Only occurs *from* the node where the command originally runs. Although a command can be automatically directed to multiple nodes, it is not directed or propagated *beyond* those nodes.

Automatic command direction is activated using the SET command and the AUTODIRECT operand to specify output options. See [z/OS Security Server RACF Command Language Reference](#) for more information about the SET AUTODIRECT command.

When automatic command direction is active, and a command runs successfully (with a return code that is less than 8), the command becomes a candidate for automatic direction. Before it is automatically directed to any other nodes, authorization checks are made against profiles in the RRSFDATA class. The authorization checks determine whether the command should be automatically directed, and which nodes it should be directed to.

**Note:** Certain commands automatically directed to systems where 8 character user IDs are not supported or enabled by TSO may fail. Examples include dataset commands (addsd, altdsd, listdsd, deldsd) when the dataset name is not specified in quotes. Also, certain rdefine and ralter commands may fail if the addmem() or delmem() operands contain dataset names that are not specified in quotes.

## Automatic command direction authorization checks

The following example illustrates how automatic direction of commands works:

1. Suppose:

- NODE1, NODE2, and NODE3 are RRSF nodes that are operative targets of each other.
- NODE2 and NODE3 have automatic command direction activated between them with the following RRSFDATA class profiles:
  - On NODE2: AUTODIRECT .NODE3 . \* with UACC(READ)
  - On NODE3: AUTODIRECT .NODE2 . \* with UACC(READ)
- CHARLIE2 exists on NODE2 and NODE3, but with no user ID association between nodes.

2. CHARLIE2 on NODE2 issues the following command:

```
ADDUSER PREMA
```

3. On NODE2, the ADDUSER PREMA command runs under the authority of CHARLIE2.
4. After the ADDUSER PREMA command runs successfully (under the authority of CHARLIE2 at NODE2), it is automatically directed to NODE3.CHARLIE2.
5. At NODE3, the ADDUSER PREMA command runs under the authority of CHARLIE2.

### Note:

1. The ADDUSER PREMA command is not automatically directed to NODE1.CHARLIE2 because there is no profile protecting the resource AUTODIRECT.NODE1.USER.ADDUSER.
2. The destination of notification and output from the ADDUSER PREMA command that ran on NODE3 is determined by what was specified on SET AUTODIRECT command issued on NODE3.
3. Once the ADDUSER PREMA command runs on NODE3, it is not automatically directed back to NODE2. RACF detects that the command was already automatically directed, and does not further send it to any other nodes.

## The AT option and automatic command direction

The AT option is used to explicitly direct a command to a user ID on a particular RRSF node. Although explicit command direction using the AT option and automatic command direction are two distinct forms of command direction, they can both occur when a command is issued within an RRSF network, as shown in the following examples.

## 1. Suppose:

- NODE1, NODE2, and NODE3 are RRSF nodes that are operative targets of each other.
- NODE2 and NODE3 have automatic command direction activated between them with the following RRSFDATA class profiles:
  - On NODE2: AUTODIRECT . NODE3 . \* with UACC(READ)
  - On NODE3: AUTODIRECT . NODE2 . \* with UACC(READ)
- CHARLIE1 on NODE1 has a peer user ID association with CHARLIE2 on NODE2.
- CHARLIE2 exists on NODE2 and NODE3, but with no user ID association between nodes.

## 2. CHARLIE1 on NODE1 issues the following command:

```
ADDUSER PREMA AT(NODE2.CHARLIE2)
```

## 3. Because the AT option was specified, the command is explicitly directed to NODE2.CHARLIE2.

## 4. At NODE2, the ADDUSER PREMA command runs under the authority of CHARLIE2.

## 5. After the ADDUSER PREMA command runs successfully (under the authority of CHARLIE2 at NODE2), it is automatically directed to NODE3.CHARLIE2.

**Note:**

1. The ADDUSER PREMA command does not run on NODE1.
2. NODE1.CHARLIE1 receives output via the RRSFLIST data set for the ADDUSER PREMA command that ran on NODE2.
3. The destination of notification and output from the ADDUSER PREMA command that ran on NODE3 is determined by what was specified on SET AUTODIRECT command issued on NODE3.

**Summary of rules for automatic direction of commands**

Here is the processing flow of checks that are made to determine whether or not a command should be automatically directed:

1. When a command is issued:
  - If AT is not specified, the command runs on the local node in the user's address space.
  - If AT is specified, the command runs in the RACF subsystem address space of the specified local or remote node.
  - If appropriate, automatic direction of commands occurs from the node where the command executed.
2. The command is not automatically directed if any of these is true:
  - a. Automatic direction of commands is inactive.
  - b. The command return code is greater than 4.
  - c. The command has already been automatically directed.
  - d. The command is ineligible for automatic direction of commands. See [“Using automatic direction of commands” on page 427](#) for more information.
  - e. The RRSFDATA class is INACTIVE.
  - f. The RRSFDATA class is ACTIVE and an AUTODIRECT profile covering that command does not exist.
3. For each remote target node, the following occurs:
  - a. If the command issuer does not have at least READ authorization to RRSFDATA profile `AUTODIRECT . target-node . classname . command-name`, the command is not automatically directed to this node
  - b. If the command has passed all checks so far, it is sent to execute on the remote node under the authority of the same-named user ID on the remote node. The user ID is the user ID under

which the command executed, which is not necessarily the command issuer (if the command was directed and then automatically directed, for example).

For example, a command issued by and executed on LAURIE at NODE1 is automatically directed to LAURIE at NODE2. A command issued by LAURIE specifying AT (NODE2 . ANDREW) is automatically directed to ANDREW at NODE1. No authorization check with the AUTODIRECT profiles is made on the receiving node.

## Using automatic direction of application updates

See [“Controlling automatic direction of passwords” on page 439](#) for information about the profiles needed for automatic direction of application updates.

### Summary of rules for automatic direction of application updates

1. When automatic direction of application updates is active, RACF automatically directs selected application updates made by the following commands and macros to selected remote nodes:
  - ICHEINTY ADD, ALTER, DELETE, DELETEA, and RENAME requests
  - The RACDCERT command
  - The RACMAP command
  - RACDEF
  - RACROUTE REQUEST=DEFINE
  - RACROUTE REQUEST=EXTRACT,TYPE=REPLACE
  - RACXTRT specifying TYPE=REPLACE
2. If profiles on two or more RRSF nodes are already synchronized, you can use automatic direction of application updates to keep the profiles synchronized with respect to application updates.
3. RACF directs an application update only after the update has successfully completed on the node where the application is executing.
4. Not all RACROUTE REQUEST=DEFINE and RACDEF requests update the RACF database, and RACF does not automatically direct requests that do not update the database. RACROUTE REQUEST=DEFINE and RACDEF are not automatically directed if:
  - ENVIR=VERIFY is specified
  - RACFIND=NO is specified and DSTYPE=T is not specified
5. RACROUTE REQUEST=VERIFY and RACINIT update the RACF database by issuing ICHEINTY macros. RACF automatically directs the following ICHEINTY requests made by RACROUTE REQUEST=VERIFY and RACINIT:
  - The ICHEINTY setting the revoke flag in the user profile when a user is being revoked due to inactivity or password attempts that are not valid
  - The ICHEINTY that increments the revoke count when a user enters a password that is not valid
  - The ICHEINTY that resets the revoke count to 0 when a user enters a valid password, if the revoke count for the user was nonzero before the update was made
6. Automatic direction of the ICHEINTY that RACROUTE REQUEST=VERIFY issues to change the password in the user's profile is controlled by automatic password direction, and not by automatic direction of application updates.
7. When a RACROUTE REQUEST=DEFINE, or a RACDEF, issues an ICHEINTY, RACF does not direct the ICHEINTY separately.
8. Automatic command direction determines whether a RACF command is directed. If a command issues a RACROUTE or ICHEINTY, that RACROUTE or ICHEINTY is not directed by automatic direction of application updates.
9. Use the AUTOAPPL and NOAUTOAPPL options on the RACF SET command to activate and deactivate automatic direction of application updates. Use the OUTPUT and NOTIFY values of SET AUTOAPPL

to specify which users will be notified of results and receive output from automatically directed application updates.

10. Profiles in the RRSFDATA class control which application updates are automatically directed to which nodes.

## Considerations for the DATASET class

Because discrete DATASET profiles are closely tied to the data set they protect and DATASET profiles for tape data sets are closely tied to profiles in the TAPEVOL class, the following are important:

- If a discrete profile is created without turning on the RACF indicator bit for the data set in the VTOC or catalog entry, the profile is not found during authority checking.
- If a discrete data set profile is renamed and the data set itself is not renamed, the discrete profile no longer is used for data set protection.
- If a discrete data set profile is deleted and the data set itself is not deleted, its protection is likely to be changed to an existing generic profile.
- Because RACROUTE REQUEST=DEFINE and RACDEF only manipulate the RACF profiles and do not modify the data set itself or its RACF indicator bit, it is not desirable to propagate these changes to a remote system.

Because these changes are controlled by the AUTODASD and AUTOTAPE profiles, it is not necessary to turn off propagation of AUTODIRECT .node .DATASET .APPL unless your own applications require this.

## RRSF considerations for digital certificates

Updates can be made to digital certificate information in the RACF database by the RACDCERT, RDEFINE, RALTER and RDELETE commands, and by applications that invoke the R\_data1ib and initACEE callable services. (While the RACDCERT command itself is not eligible for command direction and cannot be used with the AT or ONLY keyword, it might be executed on remote nodes using other methods.) These updates can affect profiles in DIGTCERT, DIGTCRIT, DIGTNMAP, DIGTRING and USER classes. Updates to profiles in these classes are eligible for automatic direction of application updates. Therefore, you must ensure consistent propagation across these classes.

### ***Suppression of private key information propagation***

When automatic direction of application updates is enabled, RACF database changes initiated by RACDCERT are propagated to other systems. These changes include additions, alterations, and deletions of certificates. However, private key information contained in the following fields of general resource profiles in DIGTCERT class is *not* propagated:

#### **CERTPRVK**

Private key or key label (whether stored in the RACF database or in the ICSF PKA key data set (PKDS))

#### **CERTPRVS**

Private key size

#### **CERTPRVT**

Private key type

Possession of a private key allows authorized certificate issuers to generate signed certificates using the SIGNWITH keyword of the RACDCERT GENCERT command. If a given private key is propagated to multiple systems, RACF cannot properly serialize the certificate serial numbers across multiple systems. By not propagating private key information, RACF prevents you from inadvertently reusing certificate serial numbers and creating multiple conflicting certificates with the same issuer, same serial number, but different certificate content.

### ***Guidelines for propagation of command and application updates***

- If you want certificates and certificate information to be propagated through your RRSF network, even though private keys are not propagated, you can define the RRSFDATA resources shown listed in [Table 30 on page 432](#). These resources are most useful when your certificates have no associated private

keys. (If you do not want to propagate certificates, you need not define the RRSFDATA resource for the DIGTCERT class.)

- In general, a certificate label should be unique within an RRSF network. This prevents an application update to delete a certificate on one RRSF node from deleting certificates on other nodes.
- If you want to create a copy of an existing certificate (and its non-ICSF private key) on a different system, use the RACDCERT EXPORT command to create a PKCS #12 format data set on the system where the certificate resides, and send the data set to the other system where you can use it as input with the RACDCERT ADD command to recreate the same certificate.

**Restriction:** If the private key is stored in the ICSF PKA key data set (PKDS), a PKCS #12 data set cannot be created.

To copy or replicate a certificate with a private key that is stored in the ICSF PKDS, see [“Migrating an ICSF private key in the PKDS from one system to another”](#) on page 582.

To ensure that RACF database updates are propagated in a consistent manner across the DIGTCERT, DIGTCRIT, DIGTNMAP, DIGTRING and USER classes, you should create a single RRSFDATA profile called AUTODIRECT . *target-node* . DIGT\*, or you should ensure that the access lists for the RRSFDATA resources that are shown in [Table 30 on page 432](#) are kept identical:

*Table 30. RRSFDATA resources to control propagation of certificate information*

Type of automatic direction	RRSFDATA resource
Automatic direction of application updates	AUTODIRECT . <i>target-node</i> . DIGTCERT . APPL
	AUTODIRECT . <i>target-node</i> . DIGTNMAP . APPL
	AUTODIRECT . <i>target-node</i> . DIGTRING . APPL
Automatic direction of commands	AUTODIRECT . <i>target-node</i> . DIGTCRIT . <i>command-name</i>

**Note:** If you want to propagate all digital certificate-related updates, the best way to ensure that RACF database updates are propagated consistently is to define a single profile that is called AUTODIRECT . *target-node* . DIGT\* and use it to control the RRSFDATA resources shown in this table.

Observe the following considerations:

- When updates are made in the USER class that pertain to digital certificates, the RRSFDATA resource AUTODIRECT . *target-node* . DIGTCERT . APPL is used to determine the authority to propagate.
- When updates are made in the USER class that pertain to certificate name filters, the RRSFDATA resource AUTODIRECT . *target-node* . DIGTNMAP . APPL is used to determine the authority to propagate.
- The resource AUTODIRECT . *target-node* . USER . APPL is not checked to determine authority to propagate USER class updates that are made as a result of processing digital certificates or certificate name filters.

## RRSF considerations for distributed identity filters

The RACMAP command updates profiles in the USER and IDIDMAP classes. (While the RACMAP command itself is not eligible for command direction and cannot be used with the AT or ONLY keyword, it might be executed on remote nodes using other methods.) Updates to profiles in these classes by the RACMAP command are eligible for automatic direction of application updates.

To ensure that RACF database updates are propagated in a consistent manner across the IDIDMAP and USER classes, you should define an RRSFDATA resource called AUTODIRECT . *target-node* . IDIDMAP . APPL. Note that the AUTODIRECT . *target-node* . USER . APPL resource is not checked to determine authority to propagate USER class updates that are made as a result of processing distributed identity filters

For information about distributed identity filters, see [Chapter 28, “Distributed identity filters,”](#) on page 671.



## Using automatic password direction

When PEER PWSYNC changes a password, the change is not propagated to other nodes by automatic password direction. However, when automatic password direction changes a password, that change results in PEER PWSYNC changing the password on all of the user IDs that have the proper association and profile access. Automatic password direction and PWSYNC work best together when peer associations exist only with other user IDs on the local system and the same set of user IDs are associated with each other on each other system. For example, NODE1.JOE1 and NODE1.JOE2 have peer PWSYNC and NODE2.JOE1 and NODE2.JOE2 have peer PWSYNC with each other (but not with Joe's user IDs on node1). When Joe changes his password on NODE1.JOE1, peer PWSYNC carries it on to JOE2. This keeps both network traffic and complexity of associations to a minimum, while keeping all of Joe's passwords the same.

If you are using automatic password direction between same named users on your system and another RRSF node, do not establish PEER PWSYNC user associations between the same user IDs across RRSF systems that use automatic password direction. Doing so would result in duplication of password synchronization requests.

## Relationship to user ID associations

No user ID associations are required for automatic password direction. If user ID associations are present, passwords are synchronized for the users with approved PEER PWSYNC associations to the user who initiated the password change.

PWSYNC associations can be used in environments with automatic password direction. In this environment, the passwords of users who have PEER PWSYNC associations with the originator of the password change are synchronized with the originator's password. Automatic password direction only synchronizes passwords between the same user IDs on multiple RRSF nodes.

## Synchronizing passwords and password phrases

You can use automatic password direction to automatically synchronize passwords and password phrases when changes are initiated by:

- Application programs that use the ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE macro to supply the user's new password or password phrase in clear text form.
- Application programs that use the ICHEINTY, RACROUTE REQUEST=VERIFY, or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE macro to change:
  - Both the password and the last password change date information, *or*
  - Both the password phrase and the last password phrase change date information.
- Application programs that use the ICHEINTY or RACROUTE REQUEST=EXTRACT,TYPE=REPLACE macro to change the last password or password phrase change date information, not the password or password phrase itself.

**Note:** Password and password phrase changes initiated using the following commands do *not* result in password or password phrase synchronization:

- ADDUSER
- ALTUSER
- PASSWORD or PHRASE.

## RRSF considerations for JES security

RRSF functions can be invoked by a batch job. For instance, RACF TSO commands that are subject to automatic command direction can be issued from within a batch job. If your JES security approach uses the &RACLNDE profile in the RACFVARS class, all JES nodes (not RRSF nodes) that you want to be treated as local nodes must be defined as members in the &RACLNDE profile. For an example, see [“Defining nodes as local input sources” on page 482](#).

You must define the JES node where the batch job is submitted as a member of &RACLNDE because there are no default members in this profile. If you do not define the submitting JES node as a member of &RACLNDE, the RRSF authority check for the function invoked by the job, such as the authority check for the RRSFDATA profile protecting the propagation of the issued command, might fail and prevent the command from being propagated to remote RRSF nodes.

## RRSF considerations for z/OS Network Authentication Service

If your installation has implemented automatic direction and you want to define multiple realms, you should review your current RRSF implementation in view of these important considerations:

1. The KERB segment of the RACF user profile defines a user as a local principal. If KERB segment information is directed to a remote RRSF node, users will be defined as local principals on all z/OS Network Authentication Service servers that share that RACF database.
2. RACF does not distinguish between user passwords and passwords assigned to local principals for key generation. The same is true for password phrases. If user passwords and password phrases are synchronized with a remote RRSF node, keys will be generated for those users on the remote node and they will be recognized as local principals by all z/OS Network Authentication Service servers that share that RACF database.
3. REALM class profiles define information about local and foreign realms. If these profiles are propagated to a remote RRSF node, all z/OS Network Authentication Service servers that share that RACF database will have duplicate local and foreign realm definitions.
4. KERBLINK class profiles map foreign principals to local RACF user IDs, and control which users are authorized to use the SKRBKDC started procedure to decrypt service tickets for a given principal. If KERBLINK profiles are propagated to a remote RRSF node, all z/OS Network Authentication Service servers sharing that RACF database will attempt to map those foreign principals to the same RACF user IDs, and allow the users authorized by the KERBLINK profiles to use SKRBKDC to decrypt service tickets for the given principal.

For more information, see [z/OS Integrated Security Services Network Authentication Service Administration](#).

## Synchronizing database profiles

You can use automatic direction to maintain synchronization of RACF database profiles that are already synchronized, but you must synchronize the profiles before you activate RRSF functions. You can do this synchronization manually, but it can be a time-consuming process. You can also run IRRDBU00 against the databases that you want to synchronize. Then use a program or REXX EXEC to compare the IRRDBU00 output for the databases and generate the commands that are needed to synchronize them. IBM provides a sample REXX EXEC, DBSYNC, to help you do this work. IBM does not support the DBSYNC EXEC.

For information on how to get this tool and others, see the link for the **RACF download** page in [“Internet sources”](#) on page xxviii.

## Controlling the use of remote sharing functions

You can control the use of most RACF remote sharing facility (RRSF) functions through profiles in the RRSFDATA class.

When you set up RRSF, you need to think on a network level, rather than on a node level. For example, suppose users can't define associations or synchronize passwords on NODE1, but they can on NODE2. Susan has user IDs on NODE1 and NODE2. On NODE2, she can create an association with her NODE1 user ID and synchronize password changes she makes on NODE2 with her NODE1 user ID. She gets an association on NODE1 even though NODE1 doesn't allow her to set them up, and gets one-way password synchronization to her NODE1 user ID, even though NODE1 doesn't allow password synchronization.

**Note:** The following sections assume that remote nodes are accepting updates from the local node. However, if the remote node uses the DENYINBOUND keyword on the **TARGET** command that defines the

local node, then updates will be not accepted, regardless of the RRSFDATA profiles defined on the local node.

## Controlling access to the RACLINK command

You can control whether users can issue the RACLINK command to establish user ID associations and enable password synchronization.

### Controlling the use of the RACLINK DEFINE operand

RACLINK commands that specify the DEFINE operand are subject to a security check to determine if the command issuer is authorized to issue RACLINK DEFINE to the specified node. Because use of the DEFINE operand is controlled at a node level, the local node and all target nodes defined to it must have appropriate profiles.

To allow a user ID association to be made to a specific node, create a profile in the RRSFDATA class to protect a resource called RACLINK.DEFINE.*node*, where *node* is the node name.

The request issuer needs to have READ access to the resource. The request will fail if:

1. The RRSFDATA class is inactive.
2. There is no RRSFDATA profile protecting RACLINK.DEFINE.*node* for the specified node.
3. The command issuer is not properly authorized to the resource.

### Controlling the use of the RACLINK PWSYNC operand

RACLINK commands that specify the PWSYNC operand are subject to a security check to determine if the command issuer is authorized to request password synchronization with user IDs that are on the specified node. Because password synchronization is controlled at a node level, the local node and all target nodes defined to it must have appropriate profiles.

To allow password synchronization to occur with user IDs on a specific node, create a profile in the RRSFDATA class to protect a resource called RACLINK.PWSYNC.*node*, where *node* is the node name.

When you issue a request to create an association with password synchronization, you need to have:

- The authority to issue the RACLINK DEFINE command with PWSYNC specified
- READ access to the RACLINK.DEFINE.*node* and RACLINK.PWSYNC.*node* resources

The request will fail if:

1. The RRSFDATA class is inactive.
2. There is no RRSFDATA resource protecting RACLINK.PWSYNC.*node* for the specified node.
3. You do not have READ access to the RACLINK.DEFINE.*node* and RACLINK.PWSYNC.*node* resources.

## Controlling password synchronization

To enable synchronization of passwords and password phrases, issue the SET PWSYNC command. (For syntax information, see [z/OS Security Server RACF Command Language Reference](#) for more information.)

For users with RACLINK PEER PWSYNC associations on an RRSF node, you can use the following resources in the RRSFDATA class to further control synchronization:

- The PWSYNC resource, to authorize users for password synchronization.

#### Example:

```
RDEFINE RRSFDATA PWSYNC UACC(NONE)
PERMIT PWSYNC CLASS(RRSFDATA) ID(SYSGRP ADMGRP) ACCESS(READ)
```

- The PHRASESYNC resource, to authorize users for password phrase synchronization.

**Example:**

```
RDEFINE RRSFDATA PHRASESYNC UACC(NONE)
PERMIT PHRASESYNC CLASS(RRSFDATA) ID(*) ACCESS(READ)
```

**Important:** When you define the PWSYNC or PHRASESYNC resources, you do not *initiate* synchronization for authorized users. For synchronization to occur, each user must have an approved RACLINK PEER association with password synchronization (PWSYNC) enabled *and* have sufficient authority for either the PWSYNC resource, the PHRASESYNC resource, or both resources. For more information, see [“Password synchronization” on page 408](#).

To be authorized for synchronization, a user must be permitted with at least READ access to the appropriate RRSFDATA resource. This allows PWSYNC requests for the user to be processed successfully. Alternatively, you can define a UACC of READ for the PWSYNC resource or the PHRASESYNC resource, or both, to authorize synchronization for all users who have approved PEER associations with PWSYNC enabled.

**Examples:**

```
RALTER RRSFDATA PWSYNC UACC(READ)
RDEFINE RRSFDATA PHRASESYNC UACC(READ)
```

**Important:** If the RACF RRSFDATA class is not active or the PWSYNC resource is not defined, password synchronization will not occur even for users with established associations. Similarly, if the RACF RRSFDATA class is not active or the PHRASESYNC resource is not defined, password phrase synchronization will not occur even for users with established associations.

To enable synchronization for users with RACLINK PEER PWSYNC associations and disable automatic password direction, issue:

```
SET PWSYNC NOAUTOPWD
```

To disable synchronization, issue:

```
SET NOPWSYNC
```

You can also use the RRSFDATA resources to control synchronization at a system level. For example, you can turn off synchronization without having to delete all of the existing user ID associations by deleting the PWSYNC or PHRASESYNC resource, or by changing the UACC to NONE with no users on the access list.

**Examples:**

```
RALTER RRSFDATA PWSYNC UACC(NONE)
RDELETE RRSFDATA PHRASESYNC
```

## Controlling the use of the AT operand

Commands that specify the AT operand are subject to a security check to determine if the command issuer is authorized to direct RACF TSO commands to the specified node. Because command direction is controlled at a node level, the local node and all target nodes defined to it must have appropriate profiles.

To allow command direction to a specific node, create a profile in the RRSFDATA class to protect a resource called `DIRECT.node`, where *node* is the node name.

The request issuer needs to have READ access to the resource. The request will fail if:

1. The RRSFDATA class is inactive.
2. There is no RRSFDATA profile protecting `DIRECT.node` for the specified node.
3. The command issuer is not properly authorized to the resource.

For information on implementing command direction using the AT operand, see [“Directing commands using the AT option” on page 412](#).

## Controlling the use of the ONLYAT operand

Commands that specify the ONLYAT operand are subject to a security check to determine if the command issuer is authorized as follows:

- The command issuer and the target user ID must be SPECIAL.
- No user ID association is required if the target user ID is the same as the command issuer. The user IDs can be on different nodes.
- If the target user ID is different from the command issuer, a user ID association between the command issuer and the target user ID is required. This prevents a SPECIAL user from unauthorized use of another remote SPECIAL user ID.

For information on implementing command direction using the ONLYAT operand, see [“Directing commands using the ONLYAT option” on page 415](#).

## Controlling automatic direction

You can control automatic direction of commands, passwords, and application updates through profiles in the RRSFDATA class. These are also controlled at a system level through the AUTODIRECT, AUTOPWD, and AUTOAPPL operands of the SET command. See [z/OS Security Server RACF Command Language Reference](#) for details.

### Controlling automatic direction of commands

Profiles in the RRSFDATA class control which commands are automatically directed to which nodes. The resource name format is:

```
AUTODIRECT.target-node.classname.command-name
```

where:

**target-node**

Is the remote node where the command is to be directed.

**classname**

Is the class name associated with the command issued. The class name can be USER, GROUP, DATASET, or any general resource class defined in the class descriptor table (CDT).

**command-name**

Is the name of the command issued.

The use of these profiles provides security for automatic command direction. An authorization check is made against these resource names to determine if the user is allowed to automatically direct the specified command. The command is directed to the remote node if:

- The RRSFDATA class has been activated.
- SET AUTODIRECT is in effect.
- There is a profile for the resource name associated with the command.
- The command issuer has at least READ access to that resource.

[Table 31 on page 437](#) lists the resource name for each RACF command that can be used with automatic command direction.

*Table 31. Automatic command direction: Resource names*

Command	Class	Resource name
ADDUSER or AU	USER	AUTODIRECT.target-node.USER.ADDUSER
ALTUSER or ALU	USER	AUTODIRECT.target-node.USER.ALTUSER
CONNECT or CO	USER	AUTODIRECT.target-node.USER.CONNECT

Table 31. Automatic command direction: Resource names (continued)

Command	Class	Resource name
DELUSER or DU	USER	AUTODIRECT. <i>target-node</i> .USER.DELUSER
PASSWORD or PW or PHRASE	USER	AUTODIRECT. <i>target-node</i> .USER.PASSWORD
REMOVE or RE	USER	AUTODIRECT. <i>target-node</i> .USER.REMOVE
ADDGROUP or AG	GROUP	AUTODIRECT. <i>target-node</i> .GROUP.ADDGROUP
ALTGROUP or ALG	GROUP	AUTODIRECT. <i>target-node</i> .GROUP.ALTGROUP
DELGROUP or DG	GROUP	AUTODIRECT. <i>target-node</i> .GROUP.DELGROUP
ADDSD or AD	DATASET	AUTODIRECT. <i>target-node</i> .DATASET.ADDSD
ALTDSD or ALD	DATASET	AUTODIRECT. <i>target-node</i> .DATASET.ALTDSD
DELDSD or DD	DATASET	AUTODIRECT. <i>target-node</i> .DATASET.DELDSD
PERMIT or PE	any general resource class or DATASET	AUTODIRECT. <i>target-node.classname</i> .PERMIT
RALTER or RALT	any general resource class	AUTODIRECT. <i>target-node.classname</i> .RALTER
RDEFINE or RDEF	any general resource class	AUTODIRECT. <i>target-node.classname</i> .RDEFINE
RDELETE or RDEL	any general resource class	AUTODIRECT. <i>target-node.classname</i> .RDELETE
SETROPTS or SETR	none (use RACF)	AUTODIRECT. <i>target-node</i> .RACF.SETROPTS

**Note:**

1. To activate automatic command direction, issue the SET AUTODIRECT command. See “Automatic direction” on page 417 and [z/OS Security Server RACF Command Language Reference](#) for more information.
2. Automatic command direction occurs only at the command level. You cannot direct a command operand or segment information for a command. For example, if you direct the ADDUSER command, you direct all ADDUSER commands, including the TSO, DFP, and OPERPARM segment information. You cannot specify automatic command direction for only the TSO segment information in the ADDUSER command.
3. You can use generic profiles to define these profiles. No commands will be directed if the RRSFDATA class is inactive or if no RRSFDATA profiles that protect AUTODIRECT exist.
4. These profiles are only checked on the node where the command was issued. Once the command is directed to another node, no authorization check is made against these profiles on the receiving node.
5. Profiles for turning on automatic direction of passwords and application updates are similar. Therefore, using \* for the command names will turn on these functions, too.
6. If your installation renames any RACF TSO commands, they are still protected under the resource names shown in Table 31 on page 437. For example, if you renamed ADDGROUP as ADDBUNCH, RACF would still use AUTODIRECT.*target-node*.GROUP.ADDGROUP as the resource name.
7. The maximum length of a directed command is approximately 4,700 bytes.

## Sample automatic command direction profiles

You can activate automatic direction of commands without activating automatic direction of application updates by using SET AUTODIRECT NOAUTOAPPL. You can also turn off password propagation by issuing the SET AUTODIRECT NOAUTOPWD command. See [z/OS Security Server RACF Command Language Reference](#) for details.

Some examples of using profiles to control automatic command direction follow. For each example, assume that no other profiles beginning with AUTODIRECT are present in the RRSFDATA class.

- To disable automatic command direction for TAPEVOL profiles and direct all other commands to all remote nodes:

```
AUTODIRECT.*.TAPEVOL.*    UACC(NONE), no users on access list
AUTODIRECT.**              UACC(READ), no users on access list
```

- To direct ADDUSER commands issued by BOB to all remote nodes:

```
AUTODIRECT.*.USER.ADDUSER UACC(NONE), BOB on access list with READ access
```

- To disable automatic command direction for TAPEVOL and RRSFDATA profiles and direct all other commands to all remote nodes:

```
AUTODIRECT.*.TAPEVOL.*    UACC(NONE), no users on access list
AUTODIRECT.*.RRSFDATA.*   UACC(NONE), no users on access list
AUTODIRECT.**              UACC(READ), no users on access list
```

- To enable automatic command direction only to NODE1 for the USER and GROUP classes:

```
AUTODIRECT.NODE1.USER.*    UACC(READ), no users on access list
AUTODIRECT.NODE1.GROUP.*   UACC(READ), no users on access list
AUTODIRECT.**              UACC(NONE), no users on access list
```

## Controlling automatic direction of passwords

Profiles in the RRSFDATA class control which passwords and password phrases get automatically directed to which nodes. The format for the resource names are any of the following:

```
AUTODIRECT.target-node.USER.PWSYNC
AUTODIRECT.target-node.USER.PHRSSYNC
```

where:

### **target-node**

Is the remote node where the command is to be directed

These profiles provide security for automatic password direction. An authorization check is made against these resource names to determine if the user's password and password phrase can be synchronized automatically. The password and password phrase change is directed to the remote node if:

- SET AUTOPWD is in effect.
- The RRSFDATA class has been activated.
- There is a profile to cover any of the resources that control automatic password direction.
- The user changing the password or password phrase has at least READ access to that resource.

You can use generic profiles to define these profiles. If the RRSFDATA class is inactive or if there is no RRSFDATA profile for automatic password direction, password and password phrase changes are not directed automatically.

The RRSFDATA profiles for automatic password direction are checked only on the node where the password is originally changed. Once the password or password phrase change is directed to another node, no authorization check is made on the receiving node.



## Sample automatic password direction profiles

Some examples of using profiles to control automatic password direction follow. For each example, assume that no other profiles beginning with AUTODIRECT are present in the RRSFDATA class.

- To enable password synchronization for users with RACLINK PEER PWSYNC associations:

PWSYNC	UACC (READ)
AUTODIRECT.*.USER.PWSYNC	UACC (NONE)

AUTODIRECT.\*.USER.PWSYNC is not required, but if you have other profiles that protect AUTODIRECT, this prevents automatic password direction.

- To enable automatic password direction for users without RACLINK associations:

PWSYNC	UACC (NONE)
AUTODIRECT.*.USER.PWSYNC	UACC (READ)

- To enable automatic password direction for users without RACLINK associations to node MVS1:

PWSYNC	UACC (NONE)
AUTODIRECT.MVS1.USER.PWSYNC	UACC (READ)

## Controlling automatic direction of application updates

Profiles, including generic profiles, in the RRSFDATA class control which application updates get automatically directed to which nodes. The format for the resource names for USER, GROUP, class descriptor table (CDT) classes, and some DATASET updates is:

```
AUTODIRECT.target-node.classname.APPL
```

where:

### **target-node**

Is the remote node the update is to be propagated to

### **classname**

Is the class name associated with the update. This is USER, GROUP, any general resource class, or DATASET for updates not covered by the AUTODASD and AUTOTAPE profiles.

The formats when you are using this syntax for automatic direction of application updates in the DATASET class are:

```
AUTODIRECT.target-node.DATASET.APPL
```

```
AUTODASD.target-node.DATASET.APPL
```

```
AUTOTAPE.target-node.DATASET.APPL
```

where:

### **target-node**

Is the remote node the update is to be propagated to

Use AUTODIRECT.target-node.DATASET.APPL to control automatic direction of application updates for DATASET when the request is RACROUTE REQUEST=EXTRACT, RACXTRT, or ICHEINTY.

Use AUTODASD when:

- The request is a RACROUTE REQUEST=DEFINE or a RACDEF.
- The CLASS value is set to, or defaults to, DATASET.
- The DSTYPE value is not T.

Use AUTOTAPE when:

- The request is a RACROUTE REQUEST=DEFINE or a RACDEF.



- The CLASS value is set to, or defaults to, DATASET.
- The DSTYPE value is T.

These profiles provide security for automatic direction of application updates. An authorization check is made against these resource names to determine if the user is allowed to make these updates. The application updates are directed to the remote node if:

- Automatic direction has been activated using SET AUTOAPPL.
- The RRSFDATA class is active.
- There is a profile to cover the resource name AUTODIRECT.*target-node.classname*.APPL, AUTODASD.*target-node*.DATASET.APPL, or AUTOTAPE.*target-node*.DATASET.APPL.
- The user directing the application update has at least READ access to that resource.

The RRSFDATA profile that protects AUTODIRECT.*target-node.classname*.APPL, AUTODASD.*target-node*.DATASET.APPL, or AUTOTAPE.*target-node*.DATASET.APPL is only checked on the node where the update originates. Once the update is propagated to another node, no AUTODIRECT authorization check is made on the receiving node.

When automatic direction of application updates is enabled, private key information is not propagated. For more information, see [“Suppression of private key information propagation”](#) on page 431.

**Note:** The maximum length of the parameter list of an application update is approximately 4,700 bytes.

### ***Sample automatic direction of application updates***

Some examples of using profiles to control automatic application update direction follow. For each example, assume that no other profiles beginning with AUTODIRECT are present in the RRSFDATA class.

- To disallow both automatic direction of commands and automatic direction of application updates for TAPEVOL and RRSFDATA profiles, disallow automatic updates for TAPEVOL and RRSFDATA profiles, disallow automatic direction of application updates for all DATASET profiles, and allow all other updates to propagate to all remote nodes:

```
AUTODIRECT.*.TAPEVOL.*      with UACC(NONE) and no users on access list
AUTOTAPE.*.DATASET.*        with UACC(NONE) and no users on access list

AUTODIRECT.**                with UACC(READ) and no users on access list
AUTODASD*.**                with UACC(READ) and no users on access list
```

- The following RRSFDATA profiles allow both automatic direction of commands and automatic direction of application updates only to NODEA for the USER and GROUP classes. In this example, no AUTOTAPE or AUTODASD profiles are defined. This gives the same results as defining the profiles with a UACC of NONE (updates are not propagated):

```
AUTODIRECT.NODEA.USER.*     with UACC(READ) and no users on access list
AUTODIRECT.NODEA.GROUP.*    with UACC(READ) and no users on access list
```

## **Establishing RACF security for RRSF TCP/IP connections**

Several implementation steps are required to enable your RRSF network to use TCP/IP node connections. They are detailed in [z/OS Security Server RACF System Programmer's Guide](#). This topic describes your role.

When your programmer implements TCP/IP for RRSF node connections, you must issue RACF commands to enable RRSF to use TCP/IP node connections.

**Note:** If you are running in a multi-task environment, your programmer does not have the ability to use TCP/IP for an RRSF node connection.

## Task roadmap for establishing RACF security for RRSF TCP/IP connections

### About this task

The following table shows the subtasks and associated instructions for using RACF commands to establish security for TCP/IP node connections.

Subtask	Associated instructions (see ...)
<a href="#">“Administer profiles in the SERVAUTH class to enable RRSF to use TCP/IP node connections” on page 442</a>	<a href="#">“Steps for administering SERVAUTH class profiles to enable RRSF to use TCP/IP node connections” on page 443.</a>
<a href="#">“Implementing an RRSF trust policy” on page 444</a>	<a href="#">“Steps for using the same, self-signed certificate for all RRSF nodes” on page 445</a>
	<a href="#">“Steps for using an internal CA to sign a server certificate for each RRSF node” on page 448</a>

### Before you begin

Before you begin, contact your programmer for the following information:

- The name of each RRSF node to be enabled to use TCP/IP node connections. For a multisystem node (an RRSF node comprised of systems that share a RACF database), you will need the name of only one member system.
- The SAF name assigned to the RRSF listener port for each node. This is the port on which an RRSF node establishes a TCP/IP socket to listen for RRSF requests from target nodes. The SAF name is assigned in the TCP/IP profile using the PORT definition statement. For details about the PORT statement, see [z/OS Communications Server: IP Configuration Reference](#).

**Guideline:** For increased security, ensure that the listener port for each node is assigned a SAF name in the TCP/IP profile.

- The following details about the Application Transparent Transport Layer Security (AT-TLS) policy defined for your RRSF network in z/OS Communication Server. You need to know the following:

- The name of the RACF key ring

The key ring name shown in the examples of the steps in [“Implementing an RRSF trust policy” on page 444](#) is IRR.RRSF.KEYRING. (This matches the default name in the sample AT-TLS policy provided in the IRRSRRSF member of SYS1.SAMPLIB.)

- The client authentication level

The acceptable level is either **Required** or **SAFCheck**. The **Required** level is sufficient for the approaches described in this topic. The higher **SAFCheck** option is briefly described in [“Considerations when using an external CA” on page 451](#).

### Administer profiles in the SERVAUTH class to enable RRSF to use TCP/IP node connections

When your programmer implements TCP/IP for RRSF node connections, you must define and activate certain SERVAUTH class profiles to enable the RRSF functions on each node to access TCP/IP network resources. Be sure to add the user ID of the RACF subsystem to the appropriate access lists as shown in the steps that follow, even when the user ID has the TRUSTED or PRIVILEGED attribute on your system.

You might need to authorize the RACF subsystem to access additional SERVAUTH resources depending on the security definitions in effect for your network. For detailed information about network security options using SERVAUTH resources, see [z/OS Communications Server: IP Configuration Guide](#).

In general, if insufficient access to a SERVAUTH resource causes an ICH408I message to be issued during an RRSF connection attempt, you will likely need to explicitly permit the user ID of the RACF subsystem.

**Exception:** During a system IPL or TCP/IP restart, if an ICH408I message occurs that is related to insufficient access to the EZB.INITSTACK*sysname.tcpname* resource, do *not* explicitly permit the RACF subsystem. This message should be ignored.

## Steps for administering SERVAUTH class profiles to enable RRSF to use TCP/IP node connections

Perform the following steps to define the required SERVAUTH profiles on each node to be enabled to use TCP/IP connections:

1. Determine whether access to the TCP/IP stack is protected. If it is, the following resource is protected in the SERVAUTH class:

```
EZB.STACKACCESS.sysname.tcpname
```

2. If the TCP/IP stack is not protected, skip this step.

If it is protected, add the user ID of the RACF subsystem to the access list for the TCP/IP stack:

### Example:

```
PERMIT EZB.STACKACCESS.NODE1.TCPIP CLASS(SERVAUTH) ID(RACFSUB) ACCESS(READ)
```

**Note:** If the TCP/IP stack is protected, do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

3. If a SAF name is assigned to the RRSF listener port in the TCP/IP profile, protect the listener port with a profile that protects the following resource:

```
EZB.PORTACCESS.sysname.stack-name.SAF-name
```

### Examples:

```
RDEFINE SERVAUTH EZB.PORTACCESS.NODE1.TCPIP.RRSF
-or-
RDEFINE SERVAUTH EZB.PORTACCESS.*.*.RRSF
```

Specify the SAF name provided by the programmer in [“Before you begin” on page 442](#).

4. Add the user ID of the RACF subsystem to the access list for the RRSF listener port:

### Example:

```
PERMIT EZB.PORTACCESS.NODE1.TCPIP.RRSF CLASS(SERVAUTH) ID(RACFSUB) ACCESS(READ)
```

**Note:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

5. Activate your SERVAUTH profile changes, as follows.

- If the SERVAUTH class is not already active, activate and RACLIST it.

### Example:

```
SETROPTS CLASSACT(SERVAUTH) RACLIST(SERVAUTH)
```

- If the SERVAUTH class is already active and RACLISTed, refresh it.

**Example:**

```
SETROPTS RACLIST(SERVAUTH) REFRESH
```

When you are finished, you have administered the SERVAUTH class profiles required to enable each RRSF node to use TCP/IP node connections.

## Implementing an RRSF trust policy

When your programmer implements TCP/IP for RRSF node connections, you must implement a trust policy based on digital certificates to allow TCP/IP communication to take place between RRSF nodes. RRSF node connections that use TCP/IP are protected using Application Transparent Transport Layer Security (AT-TLS). This trust policy is based on the requirements of AT-TLS and requires that you create a RACF key ring for each node and one or more signed server certificates. Whenever a connection between two RRSF nodes is attempted, the key ring of each node is accessed (under the authority of the RACF address space identity) and the server certificates are exchanged and validated. The key ring used for each node is the one associated with the RACF user ID of the RACF subsystem address space on the node.

This topic details the necessary steps for implementing a trust policy using either one of two approaches. At the end of this topic, some considerations for using an external CA are summarized.

- [“Using the same, self-signed certificate for all RRSF nodes” on page 445](#)
- [“Using an internal CA to sign a server certificate for each RRSF node” on page 448](#)
- [“Considerations when using an external CA” on page 451](#)

Select one approach and customize the steps as needed. The examples in these steps use the same user ID (RACFSUB) for the RACF subsystem on each RRSF node. In general, security administration is simplified when the subsystem user IDs are kept consistent across your network.

**Important:** With each approach, exclusive use of the signing CA certificate is the basis for securely authenticating the nodes in your RRSF network. Whenever a local RRSF node receives a connection attempt from a remote server that presents a digital certificate signed by a CA certificate within the local server's key ring, and the name of that key ring is specified in the AT-TLS policy, the local node accepts it as a valid RRSF node connection. Therefore, you must ensure that your signing CA certificate is used to sign *only* RRSF server certificates. Additional security controls can be optionally applied. Some of these are briefly described in [“Considerations when using an external CA” on page 451](#) and are presented in the context of a situation in which you lack exclusive use of the signing CA certificate.

## Before you begin

If your installation propagates digital certificate information using an existing RRSF APPC connection, be sure to review [“RRSF considerations for digital certificates” on page 431](#) before performing the steps below. Propagated certificate information does not contain private key information and is insufficient to help you complete the steps that follow. These instructions assume that you are not propagating digital certificate information, and the instructions in [“Using an internal CA to sign a server certificate for each RRSF node” on page 448](#) do not work if you are currently propagating digital certificate updates. This is because the same distinguished name and label are used for all instances of an RRSF server certificate, which results in collisions caused by propagation. Alternatively, you could devise a naming convention for the certificate labels and distinguished names of each RRSF server certificate and modify the instructions accordingly. If you plan to have such a convention, then propagating all of the server certificates to all of the nodes automatically accomplishes the mapping function recommended for certain trust policies (described in [“Considerations when using an external CA” on page 451](#)). If you are propagating digital certificate updates, carefully consider its effect on each of these steps, or the steps that you follow for your environment, before performing them.

## Using the same, self-signed certificate for all RRSF nodes

In this approach, implement an RRSF trust policy for TCP/IP node connections by creating one self-signed certificate that you will export and send to each TCP/IP node in your RRSF network.

For each node, you will also create a key ring to hold only the node's server certificate. For a multisystem node, a single server certificate and key ring are shared among the member systems.

### *Steps for using the same, self-signed certificate for all RRSF nodes*

Perform the following steps to create one self-signed certificate and send an exported copy of it to each RRSF node.

1. Choose a node in your RRSF network and create a self-signed certificate.

#### **Example:**

```
RACDCERT GENCERT
  WITHLABEL('RRSF Server')
  SUBJECTSDN(CN('RACF Address Space') O('YOURORG') C('US'))
  KEYUSAGE(HANDSHAKE)
  NOTAFTER(DATE(2016-09-01))
```

Do not specify the PKDS option. The private key in this step must be stored in the RACF database so that it can be exported with the certificate in Step “2” on page 445.

- 
2. Export the certificate as a PKCS #12 package.

#### **Example:**

```
RACDCERT EXPORT(LABEL('RRSF Server'))
  DSN(RACFSUB.PK12DER)
  FORMAT(PKCS12DER)
  PASSWORD('The circus is coming 2 town.')
```

- 
3. Using FTP in binary mode, transfer the export package from the local node to a data set on each remote TCP/IP node in your RRSF network. For a multisystem node, transfer the package to only one of the member systems.

- 
4. (Optional) On the local node, move the private key from the RACF database to the ICSF PKA key data set (PKDS), if available, where it will have hardware protection.

If your installation controls resources in the CSFSERV and CSFKEYS classes, ensure that the user ID of the RACF subsystem has sufficient authority even if the user ID has the TRUSTED attribute.

- a. Delete the self-signed certificate you created in Step “1” on page 445.

#### **Example:**

```
RACDCERT DELETE(LABEL('RRSF Server'))
```

- b. Re-add the self-signed certificate using the export package you created in Step “2” on page 445 and store the private key in the ICSF PKDS.

#### **Example:**

```
RACDCERT ADD(RACFSUB.PK12DER) ID(RACFSUB)
  TRUST
  WITHLABEL('RRSF Server')
  PASSWORD('The circus is coming 2 town.')
  PKDS(RRSFserverkey)
```

- 
5. (Optional) Delete the data set containing the export package because it is no longer needed.

If you opted to leave the private key in the RACF database, you can delete the export package. If you want to add a new TCP/IP node in the future, you can reuse the same RRSF server certificate by exporting it, as you did in Step “2” on page 445, and transferring it to the new node.

If you opted in Step “4” on page 445 to move the private key to the ICSF PKDS, do not delete the export package when you want to use the same RRSF server certificate with any new TCP/IP node that you might add in the future. If you delete the export package, you will need to create and distribute a new self-signed server certificate (with a different subject's distinguished name) for a new TCP/IP node.

---

6. On the local node, create a RACF key ring for RRSF and add the server certificate to the ring.

a. Create the RRSF key ring.

**Example:**

```
RACDCERT ID(RACFSUB) ADDRING(IRR.RRSF.KEYRING)
```

Specify the key ring name provided by the programmer in “Before you begin” on page 442.

b. Connect the server certificate to the key ring.

**Example:**

```
RACDCERT ID(RACFSUB) CONNECT(LABEL('RRSF Server')
RING(IRR.RRSF.KEYRING)
DEFAULT
USAGE(PERSONAL))
```

c. Permit the user ID of RACF subsystem to access the key ring by administering a profile in either the FACILITY or the RDATA LIB class.

**Note:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACFSUB) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB IRR.RRSF.KEYRING.LST UACC(NONE)
PERMIT IRR.RRSF.KEYRING.LST CLASS(RDATA LIB) ID(RACFSUB) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

---

7. On each remote RRSF node, add the self-signed certificate using the export package you transferred in Step “3” on page 445.

**Note:** These are the same steps you performed for the local node in Steps “4.b” on page 445 and “5” on page 445.

- a. Add the certificate and store the private key in the ICSF PKDS, if available.

If your installation controls resources in the CSFSERV and CSFKEYS classes on the remote node, ensure that the user ID of the RACF subsystem has sufficient authority even if the user ID has the TRUSTED attribute.

**Example:**

```
RACDCERT ADD(RACFSUB.PK12DER) ID(RACFSUB)
  TRUST
  WITHLABEL('RRSF Server')
  PASSWORD('The circus is coming 2 town.')
  PKDS(RRSFserverkey)
```

- b. (Optional) Delete the data set containing the export package because it is no longer needed.

8. On each remote RRSF node, create a RACF key ring for RRSF and add the server certificate to the ring.

**Note:** These are the same steps you performed for the local node in Step “6” on page 446.

- a. Create the RRSF key ring.

**Example:**

```
RACDCERT ID(RACFSUB) ADDRING(IRR.RRSF.KEYRING)
```

Specify the key ring name provided by the programmer in “Before you begin” on page 442.

- b. Connect the server certificate to the key ring.

**Example:**

```
RACDCERT ID(RACFSUB) CONNECT(LABEL('RRSF Server')
  RING(IRR.RRSF.KEYRING)
  DEFAULT
  USAGE(PERSONAL))
```

- c. Permit the user ID of RACF subsystem to access the key ring by administering a profile in either the FACILITY or the RDATA LIB class.

**Note:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACFSUB) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB IRR.RRSF.KEYRING.LST UACC(NONE)
PERMIT IRR.RRSF.KEYRING.LST CLASS(RDATA LIB) ID(RACFSUB) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETRPTS RACLST(RDATA LIB) REFRESH
```

When you are finished, you have created a key ring for each TCP/IP node and added its signed server certificate to the ring. You have now implemented an RRSF trust policy for TCP/IP node connections.

Using an internal CA to sign a server certificate for each RRSF node

In this approach, implement an RRSF trust policy for TCP/IP node connections by creating an internal certificate-authority (CA) certificate that you will use to sign an individual server certificate for each TCP/IP node connection in your RRSF network.

**Remember:** Be sure to use this CA certificate to sign *only* server certificates for RRSF nodes.

For each node, you will also create a key ring to hold only the node's server certificate and the signing CA certificate. For a multisystem node, a single server certificate and key ring are shared among the member systems.

Steps for using an internal CA to sign a server certificate for each RRSF node

Perform the following steps to create a CA certificate that you will use to sign an individual server certificate for each TCP/IP node connection in your RRSF network.

1. Choose a system in your RRSF network as the CA host system. This system will host your new signing CA certificate.
  - a. On the CA host system, create the RRSF CA certificate.

**Example:**

```
RACDCERT CERTAUTH GENCERT
  WITHLABEL('RRSFCA')
  SUBJECTSDN(CN('RRSFCA') O('YOURORG') C('US'))
  NOTAFTER(DATE(2031-09-01))
```

- b. List the new RRSF CA certificate and record the issuer's distinguished name and serial number.

**Example:**

```
RACDCERT CERTAUTH LIST(LABEL('RRSFCA'))
```

Issuer's distinguished name	Serial number

2. On the CA host system, create the server certificate for the local RRSF node and associate it with the user ID of the RACF subsystem.

**Example:**

```
RACDCERT ID(RACFSUB) GENCERT
  SIGNWITH(CERTAUTH LABEL('RRSFCA'))
  WITHLABEL('RRSF Server')
  SUBJECTSDN(CN('RACF Address Space') O('YOURORG') C('US'))
  KEYUSAGE(HANDSHAKE)
  NOTAFTER(DATE(2016-09-01))
```

3. On a remote RRSF node, create a certificate request for a server certificate for this remote node. For a multisystem node, create the request on only one of the member systems.
  - a. Create a self-signed *placeholder* certificate.



**Example:**

```
RACDCERT ID(RACFSUB) GENCERT
  WITHLABEL('RRSF Server')
  SUBJECTSDN(CN('RACF Address Space') O('YOURORG') C('US'))
  KEYUSAGE(HANDSHAKE)
  NOTAFTER(DATE(2016-09-01))
```

- b. Create a certificate request based on the self-signed certificate you just created.

**Example:**

```
RACDCERT GENREQ(LABEL('RRSF Server')) ID(RACFSUB) DSN(RRSF.REQ)
```

**Result:** A base64-encoded certificate request is stored in the specified data set on the remote node.

- 
4. Transfer the certificate request from the data set on the remote node to a data set on the CA host system.

Because this is a base64-encoded request, transfer the request as text to ensure that the ASCII translation from EBCDIC takes place. To do this, use ASCII (not binary) FTP or copy the text of the request from the data set on the remote node and paste it into an empty data set with identical attributes on the CA host system. Be sure to include the BEGIN and END lines.

**Note:** The example in the next step specifies a data set on the CA host system with the same name as the data set on the remote node.

- 
5. On the CA host system, create the server certificate for the remote node and sign it with the RRSF CA certificate you created in Step “1.a” on page 448.

Because the certificate created in this step is for the remote node and will not be used on the host system, you can associate the certificate with any user ID. If you choose to associate it with the same user ID as the certificate used by the local RRSF node (created in Step “2” on page 448), specify a different certificate label using the WITHLABEL operand to avoid an error in this step.

**Example:**

```
RACDCERT GENCERT(RRSF.REQ) ID(RACFSUB)
  SIGNWITH(CERTAUTH LABEL('RRSFCA'))
  WITHLABEL('RRSF Server2')
```

- 
6. On the CA host system, export the newly signed certificate to a data set. You can use the same data set that you used to store the certificate request in Step “3.b” on page 449.

**Example:**

```
RACDCERT EXPORT(LABEL('RRSF Server2')) ID(RACFSUB)
  DSN(RRSF.REQ)
  FORMAT(PKCS7B64)
```

**Result:** RACF stores the resulting export package in the specified data set in the PKCS #7 format. The package contains both the new server certificate and the RRSF CA certificate that signed it.

Optionally, now delete the server certificate on the CA host system that you created in Step “5” on page 449 because it is no longer needed.

**Example:**

```
RACDCERT DELETE(LABEL('RRSF Server2')) ID(RACFSUB)
```

- 
7. Transfer the export package from the data set on the CA host system to a data set on the remote node.

Use ASCII (not binary) FTP or copy the text of the certificates from the data set on the CA host system and paste it into an empty data set with identical attributes on the remote node. Be sure to include the BEGIN and END lines. (To view sample certificate text, see [“Base64-encoded certificates”](#) on page 542.)

For a multisystem node, transfer the export package to only one of the member systems.

Optionally, now delete the data set on the CA host system because it is no longer needed.

- 
8. On the remote node, add the newly signed certificate to replace the self-signed *placeholder* certificate you created in Step [“3.a”](#) on page 448.

- a. Add the RRSF CA certificate:

**Example:**

```
RACDCERT ADD(RRSF.REQ) ID(RACFSUB) TRUST WITHLABEL('RRSF Server')
```

**Results:** Both the server certificate and the RRSF CA certificate are added to the RACF data base on the remote node. The RRSF CA certificate is assigned a generated label. The following message is issued:

```
IRRD152I Root Certificate Authority not currently defined to RACF. Top
CERTAUTH certificate added with the TRUST status.
```

- b. Find out the generated label of the RRSF CA certificate. To do this, list the RRSF CA certificate by specifying the issuer's distinguished name and the serial number you recorded in Step [“1.b”](#) on page 448. Make note of the generated certificate label.

**Example:**

```
RACDCERT LIST(ISSUERSDN('CN=RRSFCA.O=YOURORG.C=US')) SERIALNUMBER(00))
CERTAUTH
```

If you did not record the issuer's name and serial number in Step [“1.b”](#) on page 448, issue the RACDCERT LIST CERTAUTH command and review the list of CA certificates. Locate the RRSF CA certificate by its subject's distinguished name, for example CN('RRSFCA') O('YOURORG') C('US')), and make note of the label.

- c. (Optional) Modify the label of the RRSF CA certificate.

**Example:**

```
RACDCERT ALTER(LABEL('generated-label')) CERTAUTH NEWLABEL('RRSFCA')
```

- 
9. On each RRSF node, create a RACF key ring for use with RRSF and add both the RRSF CA certificate and the server certificate to the ring.

- a. Create the RRSF key ring.

**Example:**

```
RACDCERT ID(RACFSUB) ADDRING(IRR.RRSF.KEYRING)
```

Specify the key ring name provided by the programmer in [“Before you begin”](#) on page 442.

- b. Connect the server certificate to the key ring.

**Example:**

```
RACDCERT ID(RACFSUB) CONNECT(LABEL('RRSF Server')
RING(IRR.RRSF.KEYRING)
DEFAULT
USAGE(PERSONAL))
```

- c. Connect the RRSF CA certificate to the key ring.

**Example:**

```
RACDCERT ID(RACFSUB) CONNECT(CERTAUTH LABEL('RRSFCA'))
RING(IRR.RRSF.KEYRING)
USAGE(CERTAUTH)
```

- d. Permit the user ID of RACF subsystem to access the key ring by administering a profile in either the FACILITY or the RDATA LIB class.

**Note:** Do not skip this step even when the user ID of RACF subsystem has the TRUSTED or PRIVILEGED attribute on your system.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACFSUB) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB IRR.RRSF.KEYRING.LST UACC(NONE)
PERMIT IRR.RRSF.KEYRING.LST CLASS(RDATA LIB) ID(RACFSUB) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

---

When you are finished, you have created two key rings, one for the CA host system and one remote TCP/IP node in your RRSF network. You have also added to each ring the signed server certificate for each node and its signing CA certificate.

**Important:** For each additional RRSF node that you want to allow to communicate using TCP/IP, repeat Steps “3” on page 448 through “9” on page 450. When you are finished, you have implemented an RRSF trust policy for TCP/IP node connections.

## Considerations when using an external CA

When you must use an external CA, there are two possible approaches. The first approach is to request only an intermediate CA certificate from the external CA and then use it to sign *only* an individual server certificate for each RRSF node. This approach is similar to the approach described in [“Using an internal CA to sign a server certificate for each RRSF node” on page 448](#).

The second approach when using an external CA is to generate a request for a server certificate for each RRSF node and send those requests to the external CA. When returned, add each server certificate and CA certificate to the key ring for each RRSF node. If the CA is well known, the CA certificate might already reside in the RACF database. If so, connect it to the key ring of each node and mark it as trusted. Remember that *any* server presenting a certificate signed by this CA certificate will be accepted as a valid RACF node. Therefore, the following guidelines for increased security apply to the second approach when you lack exclusive use of the signing CA certificate:

**Guidelines:**

- Ensure that the **SAFCheck** level of client authentication is specified in the Communication Server AT-TLS policy for RRSF. This level requires that you map each RRSF server certificate to a RACF user ID. To do this, you can use the `hostIdMapping` certificate extension if it is available from your external CA or if you use PKI Services. Otherwise, you can add a certificate name filter on each node for every other node or add each server certificate to the RACF database of every other node. If your installation propagates digital certificate information using an existing RRSF APPC connection, the task of adding each server certificate to the RACF database of every other node can be automatically performed for you. (For important RRSF considerations, see [“Before you begin” on page 444.](#)) If propagation is not available, using certificate name filters requires the least administrative effort. (For details, see [“Certificate name filtering” on page 571.](#))
- If your external CA might issue other certificates using the same CA certificate used to issue your server certificates, implement a *one-to-one* certificate filter for each server certificate. This ensures that no other issued certificate will match your filter's distinguished name (DN). If you are confident that the external CA will not issue other certificates with the same DN, you can reduce administrative effort by implementing a single *many-to-one* filter on each node based on the naming convention used for the distinguished names of your RRSF servers.
- Protect a resource named `IRR.RRSF.CONNECT` in the `RRSFDATA` class. Be sure to give the mapped user ID at least READ access to it even if the user ID has the TRUSTED or PRIVILEGED attribute. This `RRSFDATA` profile can also be used to create a RACF audit trail that logs the establishment of inbound RRSF connections to the local system.

## Chapter 18. Providing security for JES

You can use JES initialization statements, JES installation exits, RACF, or any other functionally equivalent program to provide security for JES.

This topic describes how to use RACF to provide security for JES. In this topic, the term JES refers to both JES2 and JES3, except when there is a difference between the two.

### Planning for security

You and your JES system programmer should develop a security plan for JES. Together, you should determine which resources you want to protect and decide who should have access to those resources. Your security plan should address questions such as:

- What resources must I protect?
- Should I restrict jobs and users from certain information depending on criteria such as security labels?
- Should I limit the job names that users can submit, cancel, or modify?
- Is it important to protect SYSIN and SYSOUT?
- Which remote workstations should access my system?
- Can other nodes submit jobs to my system?
- To which nodes should I allow my system to send data?
- Should I limit the commands that an operator can use?
- Do I want to restrict the consoles that an operator can use to enter certain commands?
- What commands should I allow jobs, workstations, and nodes to submit to my system?
- Do I want only selected output devices to process particular output?
- Should the security label of the output appear on the header pages?

You must gather a great deal of information from your JES system programmer about specific resources and access requirements. See [z/OS JES2 Initialization and Tuning Guide](#).

### How JES and RACF work together

JES requests RACF services by issuing the RACROUTE macro. The MVS system authorization facility (SAF) handles the RACROUTE macro invocation. If RACF is installed, SAF passes the security information specified by JES on the RACROUTE macro invocation to RACF. RACF evaluates the security information and returns the results of that evaluation to JES. JES then enforces the results of the security check, such as permitting or denying access to a data set, or allowing a job to execute.

Your JES system programmer can use JES2 macros to place additional calls to SAF in JES installation exits. See [z/OS JES2 Macros](#) and [z/OS JES2 Installation Exits](#).

**JES code that can bypass RACF protection:** Your installation can use JES initialization statements and JES installation exits to provide protection and, in some cases, to bypass RACF protection. For more information, see [z/OS JES2 Initialization and Tuning Guide](#).

### Defining JES as a RACF started procedure

For performance and ease of migration, JES should be defined as a started procedure *with the trusted attribute* when RACF is installed.

To do this, perform the following steps:

1. Ask your JES system programmer for the name of your JES started procedure.
2. Verify that the name of your JES started procedure exists as one of the following:

- An entry in the STARTED class (the preferred method). If it is not defined, you need to create one. For example, if the name of your started procedure is JES2, issue:

```
SETRPTS GENERIC(STARTED)
RDEFINE STARTED JES2.* UACC(NONE)
STDATA(USER(jes-userid) TRUSTED(YES))
```

Be sure to refresh the class after you create the entry. Issue:

```
SETRPTS RACLIST(STARTED) REFRESH
```

- An entry in the RACF started procedures table (module ICHRIN03). If none is defined, create an entry for JES.

In either case, be sure that JES has the *trusted* attribute.

### 3. Create a user profile for the JES started procedure:

```
ADDUSER jes-userid
DATA('JES started procedure')
NOPASSWORD
DFLTGRP(appropriate-group)
```

For more information on adding a started procedure, see [“Using started procedures” on page 141](#) and [z/OS Security Server RACF System Programmer's Guide](#).

## Forcing batch users to identify themselves to RACF

To prevent unauthorized users from running batch jobs, you can require all batch jobs to have RACF identification. To do this, enter the following:

```
SETRPTS JES(BATCHALLRACF)
```

When you specify BATCHALLRACF, any batch job that does not have a RACF-defined user specified on the USER parameter of the JOB statement, or propagated security information associated with it, fails.

Specifying NOBATCHALLRACF allows such jobs to run.

## Support for execution batch monitor (XBM)

Execution batch monitor (XBM) jobs are processed in the same way as normal batch jobs. To require all XBM jobs to have RACF identification, enter:

```
SETRPTS JES(XBMALLRACF)
```

With this operand in effect, any XBM job that does not have a RACF-defined user ID and password on the JOB statement, or propagated RACF identification associated with it, fails.

Specifying NOXBMALLRACF allows XBM jobs to run without RACF user IDs.

**Note:** On systems with the XBMALLRACF operand in effect, the BATCHALLRACF operand controls batch jobs *other than* jobs that run under an execution batch monitor (XBM).

## Defining and grouping operators

Your security plan can require that system programmers and operators at your installation be defined to RACF. To improve accountability, you should create user profiles for any persons who:

- Issue JES commands
- Enter commands from an MCS-managed console
- Update system data sets that JES uses

You can organize your installation's support personnel, particularly operators, into groups that are responsible for a particular area. For example, you might want to group your installation's support personnel by shift, functional area, or both.

To identify and group your installation's support personnel, you should:

- List the user IDs of all of the system programmers and operators at your installation
- Group any of the user IDs together if they:
  - Perform similar tasks
  - Work on the same shift
  - Are responsible for the same area

If you decide to combine users into groups, you must:

- Record all of the user IDs in the group
- Describe why the users were grouped together
- Assign a unique name to the group

You should keep a record of user IDs and group name handy for use in securing other system resources such as spool data, console access, and commands as well as for updating groups in the future.

## JES user ID early verification

---

Early verification is always done, even if the SETROPTS command has been issued with JES(NOEARLYVERIFY) specified.

## User ID propagation when jobs are submitted

---

For each previously validated RACF user who submits a batch job to JES through a JES internal reader, SAF propagates the following security information to the batch job:

- If USER is not specified on the JOB statement, the current RACF user ID is used.
- If PASSWORD is not specified on the JOB statement, the current user password is not required if the submitter propagates.
- If SECLABEL is not specified on the JOB statement, the submitter's current security label is used.

**Note:** If GROUP is not specified on the JOB statement, the default connect group is used from the user profile of the user used for the job.

This has the following advantages:

- It reduces the possible exposure of security information (especially passwords) stored in clear text in JCL.
- It reduces administrative overhead of maintaining RACF user IDs, passwords, and security labels in the JOB statements for all batch jobs.

As a result, a TSO user, for example, is not required to specify this security information for each job submitted.

**Note:** You can prevent user ID propagation for specific users. See [“Controlling user ID propagation in a local environment”](#) on page 457.

## Allowing surrogate job submission

You can allow the use of surrogate users. A *surrogate* user is a RACF-defined user who has been authorized to submit jobs on behalf of another user (the *execution* user) without specifying the execution user's password. Jobs submitted by a surrogate user run with the identity of the execution user. For

example, if user JOE submits a job with the following JOB statement, JOE is the surrogate user and TOM is the execution user:

```
//jobname JOB 'accounting-information',USER=TOM
```

All access checks are done with TOM's user ID. Any auditing records produced by RACF because of the submitted job's actions include the information that the job is a surrogate job (unless the PASSWORD parameter is specified on the JOB statement).

**Important:** A user should not allow another user to act as surrogate user unless the surrogate user can be trusted as highly as the execution user is trusted. This is because the surrogate user can do anything the execution user can do (unless the surrogate user lacks access to a security label that protects a resource). For example, the surrogate user can submit a job to copy, alter, or delete the execution user's data.

The surrogate user must specify the execution user's user ID on the USER parameter on the JOB statement and must *not* specify a password. If the PASSWORD parameter is specified with a password, surrogate processing is not performed, and audit records generated by the job's activities do not indicate that the job is a surrogate job. This applies not only to jobs submitted through the TSO SUBMIT command, but any time the surrogate user is a RACF-defined user.

#### When the SECLABEL class is active and the SETROPTS MLS option is in effect:

- If a security label is specified for the submitted job, the specified label must be equal to or greater than the current security label of the surrogate user, *and* both the surrogate and execution users must have READ authority to the specified label. After job verification is complete, the job submitted by the surrogate user runs as if the execution user had submitted the job.
- If a security label is *not* specified for the submitted job but the surrogate user has a current security label, the submitted job runs with the surrogate user's current security label.

To allow surrogate users, do the following:

1. Ensure that the installation exit for the TSO SUBMIT command (IKJEFF10) does not prevent users from submitting jobs with job names that do not match their user IDs. The installation exit supplied by IBM meets this requirement, because it does not check the JCL of submitted jobs. For more information, see *z/OS TSO/E Customization*.
2. If your installation implemented the sample ICHRTX00 exit from SYS1.SAMPLIB member RACINSTL to enable surrogate user processing, you should migrate to profiles in the SURROGAT class. After RACF is installed, the code in the ICHRTX00 exit that checks \$SUBMIT.userid profiles is not used. You should copy the \$SUBMIT.userid profiles to SURROGAT profiles as follows:

```
RDEFINE SURROGAT execution-userid.SUBMIT
FROM($SUBMIT.execution-userid) FCLASS(FACILITY)
```

3. Define resource profiles in the SURROGAT class for each execution user who needs to allow others to be surrogate users:

```
RDEFINE SURROGAT execution-userid.SUBMIT UACC(NONE) OWNER(execution-userid)
```

**Note:** Specifying the OWNER operand allows the execution user to issue the PERMIT command for this profile.

4. To specify that another user can act as the surrogate for an execution user, give the surrogate user READ access authority:

```
PERMIT execution-userid.SUBMIT CLASS(SURROGAT) ID(surrogate-userid) ACCESS(READ)
```

Only users and groups that have READ access authority are allowed to submit jobs on behalf of another user.

To check whether a user can submit jobs for another user, make sure the user (or a group the user is a member of) is in the access list with READ access authority:

```
RLIST SURROGAT execution-userid.SUBMIT AUTHUSER
```



5. When you are ready to control access using the SURROGAT profiles, activate the SURROGAT class:

```
SETROPTS CLASSACT(SURROGAT)
```

If the SURROGAT profile is not already raclisted, then run:

```
SETROPTS RACLIST(SURROGAT)
```

If the SURROGAT profile is already raclisted, you should run:

```
SETROPTS RACLIST(SURROGAT) REFRESH
```

To disable surrogate support for a particular user, delete the profile for that user. To disable surrogate support for all users, deactivate the SURROGAT class.

#### NJE notes:

The node in which SURROGAT checking occurs depends on the job statements (see [“Where NJE jobs are verified”](#) on page 458). For verification done on the receiving node, the SURROGAT checking is done *after* any translation due to NODES profiles. (See [“Understanding NODES profiles”](#) on page 469.)

If the submitter of a job is a started procedure, the execution node is not checked during SURROGAT processing.

## Controlling user ID propagation in a local environment

In some environments, such as CICS, jobs submitted without the USER operand specified on the JOB statement run under a user ID other than the user submitting the job. For example, if a user running under CICS submits a batch job without specifying a user ID on the JOB statement, the job runs under the CICS user ID and has the access authorities of the CICS user ID.

You can prevent the CICS user ID from being propagated to these batch jobs by defining a profile whose name is the CICS user ID. Follow these steps:

1. Define a profile in the PROPCNTL class where the profile name is the user ID that is not to be propagated (in this case, user ID CICS1 is not to be propagated):

```
RDEFINE PROPCNTL CICS1
```

2. Activate the PROPCNTL class:

```
SETROPTS CLASSACT(PROPCNTL)
```

3. Issue the SETROPTS command with the RACLIST operand to activate SETROPTS RACLIST processing for the PROPCNTL class:

```
SETROPTS RACLIST(PROPCNTL)
```

If you do not activate SETROPTS RACLIST processing for the PROPCNTL class, RACF ignores the profiles you create in this class.

The above sequence of commands eliminates user ID propagation of the user ID CICS1.

#### Note:

1. For a profile in the PROPCNTL class, RACF checks only for the presence or absence of a profile in this class. If a profile exists for a particular user ID, user ID propagation does not occur for that user ID.
2. RACF performs no logging and issues no messages for profiles in the PROPCNTL class.
3. PROPCNTL profiles only control propagation for local jobs. If the installation uses &RACLNDE for its remote nodes, only the PROPCNTL profiles are necessary. For more information on the use of &RACLNDE, see [“Defining nodes as local input sources”](#) on page 482. If the installation uses NODES profiles, it must also use them to control propagation (see [“Controlling user ID propagation in an NJE environment”](#) on page 475).

4. If you have controlled a user ID using the PROPCNTL class, and that user wants to submit a batch job to run from that user ID, the JOB statement must contain both the user ID and proper password. For example, if user A submits a job with USER=A, PASSWORD=*password* must also be specified.

However, if a different user wants to submit a job using the controlled user ID, that user can either specify the user ID and password as above, or use the facilities provided by the SURROGAT class and just specify the user ID. For example, if you controlled user A using the PROPCNTL class, user B could submit a job, specifying only USER=A with the appropriate SURROGAT authorization.

## Using protected user IDs for batch jobs

You can define the user IDs associated with batch jobs as *protected* user IDs. This will prevent them from being revoked through inadvertent or malicious incorrect password and password phrase attempts, or from being used for another purpose when a password or password phrase is normally supplied, such as logging on to the system. See [“Defining protected user IDs” on page 74](#) for information on implementing protected user IDs.

In order to execute a batch job using a protected user ID, you must submit the job through a means that does not require a password, such as through user ID propagation or surrogate job submission. Jobs that are submitted with USER= and PASSWORD= specified on the JOB statement cannot be associated with protected user IDs.

## Propagating protected user IDs

If a started procedure or job executes with a protected user ID (for example, USERA) and user ID propagation is enabled for USERA, any job submitted by USERA that does not have USER= specified on the job statement will execute with a protected user ID, USERA.

See [“Assigning RACF user IDs to started procedures” on page 141](#) for information on using protected user IDs for started procedures.

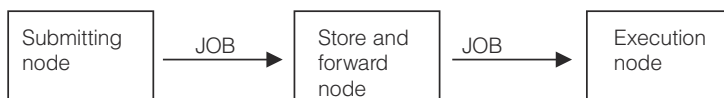
## Using protected user IDs for surrogate job submission

Surrogate user IDs and execution user IDs can be defined as protected user IDs. A started procedure or job that executes with a protected user ID can be authorized to submit jobs that have USER= specified on the job statements. The execution user IDs associated with the submitted jobs can also be defined as protected at your option.

A started procedure or job that does not execute with a protected user ID can be authorized to submit jobs that execute with protected user IDs.

## Where NJE jobs are verified

The following is a simple network showing the path of a job.



User verification for NJE jobs normally is done at the execution node. However, RACF authorization checking might occur additionally at the submitting node, depending on the following:

- Those jobs sent using the JES2 /\*ROUTE XEQ statement or /\*XEQ statement are verified at the execution node only.
- Those jobs sent using the JES2 /\*XMIT statement or the JES3 /\*ROUTE XEQ or //XMIT statement have their first JOB statement verified at the submitting node and their second JOB statement verified at the execution node.

Submitter information is propagated from trusted nodes. The submitter information is:

- The token for a verified first job card

- The original submitter's token if the job was submitted from an internal reader
- The *unknown user* token if the job was submitted from a physical reader
- NJE header information (no token available) if the job was submitted from a downlevel node

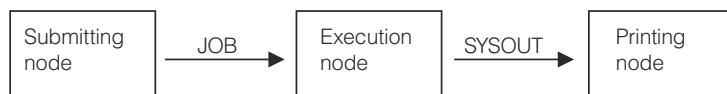
Whether a job is accepted is based on a combination of the submitter's ID, group, or security label.

Whether security information is propagated and translated is based on the submitter's ID (as taken from above). Job acceptance and translation is done using profiles in the NODES class. RACF finds the best fit among the profiles in the NODES class and uses the information specified in the UACC and ADDMEM information.

For more information, refer to [“Authorizing network jobs and SYSOUT \(NJE\)”](#) on page 468.

## How SYSOUT requests are verified

The following is a simple network showing the path of a job:



For inbound SYSOUT, user verification occurs at the printing node instead of the submitting node (as it can for inbound jobs). On the printing node, RACF authorization checking occurs in the NODES class, as it does for inbound jobs. RACF finds the best fit among the profiles in the NODES class and uses the information specified in the UACC and ADDMEM information.

Whether the SYSOUT is accepted is based on a combination of the owner's ID, group, or security label. Whether the security information is accepted and translated is based on the owner's ID taken from:

- The job token from the NJE header as verified at the executing node
- If no token is available (SYSOUT is from a downlevel node), the owner is considered to be the NJE undefined user as defined by:

```
SETROPTS(JES(NJEUSERID(userid)))
```

In addition, if &SUSER (submitting user) is specified on the ADDMEM operand, the submitter can be used as the owner if one of the following is also true:

- The submitting node is defined as a local node in the &RACLNDE profile in the RACFVARS class. In this case, the submitting user and group are used as the SYSOUT owner values and are unchanged (no translation).
- The NODES profile that matches is the profile named *submit-node*.USERS.*submitter* and UACC(CONTROL) is specified.

If there is a translate value, but it is not &SUSER, the SYSOUT owner user ID is the translate value. If it is &SUSER, the owner is the unchanged submitter user ID. In addition, a lookup is done for the NODES profile that matches the form *submit-node*.GROUPS.*submit-group*. If this profile has an ADDMEM translate value, that value is used as the SYSOUT owner group. Otherwise, the unchanged submit group is used. The UACC for this profile does not matter.

## Security labels for JES resources

When your installation uses security labels, each protected JES resource can have a security label associated with it. For spooled data sets, JES maintains the security label with the data set itself (not in a RACF profile). For other resources like consoles and DASD data sets, RACF maintains the security label in the resource profile that protects the resource.

**Note:** Security labels should be consistent throughout a JES complex to prevent information from being declassified. For more information about security labels, see Chapter 5, [“Classifying users and data,”](#) on page 95. For details, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

## Controlling access to data sets JES uses

---

The JES spool and checkpoint data sets are critical for proper operation of your JES system. It is critical that JES be the only user that can update the information in these data sets. However, a limited group of users must be able to re-create the spool and checkpoint data sets (should the data set become unusable because of hardware problems). Also, restrict access to the data set that contains the modules that JES uses. Make sure profiles exist for any data sets you might use for JES2 checkpoint reconfiguration.

You can define data set profiles to protect the system data sets that JES uses to control its own processing. Protecting your installation's system data sets prevents unauthorized users or jobs from accessing, modifying, or destroying critical system data.

The JES system programmer should supply you with the following information for each data set to be protected:

- The name of the data set
- The universal access authority to be associated with the data set
- The security label to be associated with the data set (if labels are being used)
- Whether audit records should be generated:
  - Each time the data is accessed
  - When an unauthorized attempt is made to access the data set
  - When an authorized attempt is made to access the data set

Make sure to define JES as a started procedure with the trusted attribute. See [“Defining JES as a RACF started procedure”](#) on page 453.

## Controlling input to your system

---

You can use RACF to ensure that jobs entering your JES system are authorized for processing. JES carries security information about the submitter of a job internally and invokes the services of RACF at key points during JES processing, such as when a job enters the system through a TSO SUBMIT command or through a device reader.

You can protect several job entry resources including the use of job names and sources of job entry such as internal readers, device readers, RJP, RJE, and network nodes. This topic describes how RACF validates users, how to control the use of job names, and how to control the use of input sources. [“Authorizing network jobs and SYSOUT \(NJE\)”](#) on page 468 provides a separate discussion of how to control security for inbound and outbound network jobs and SYSOUT.

## How RACF validates users

When RACF is active, RACF ensures that the job's password, user ID, group name, and security label are valid before allowing the job to be processed. If security labels are being used, JES obtains the label from the job card. If the job card does not specify a label, RACF obtains the security label from the RACF profile associated with that job's user ID. If no security label exists in the RACF profile, the job is automatically assigned a security label of SYSLOW.

The extent to which RACF performs user validation for jobs entering the system through NJE nodes depends on the universal access authority assigned to that node. [“Authorizing network jobs and SYSOUT \(NJE\)”](#) on page 468 lists those values and their effects on user validation.

You can allow the setting up of surrogate users. A surrogate user is a user who submits jobs on behalf of another user.

Surrogate job submission allows a user to submit jobs on behalf of another user without having to specify the original user's password. Jobs submitted by a surrogate user execute with the identity of the original user. Although the surrogate user does not have to provide the password of the original user, RACF ensures that the job's security label overrides the surrogate's security label and that the original user

is authorized to use the security label associated with the job. For additional information about defining surrogate job submission, see [“Surrogate job submission” on page 467](#).

## Propagating security information

If RACF is active and a job enters the system with some or all security information missing, SAF propagates the submitter's security information to the job. For example, if JOB1 submits JOB2, and JOB2 does not have SECLABEL= specified on the JOB JCL statement, SAF passes the value of SECLABEL= from JOB1's JOB statement to JOB2.

Any user in a currently active session (for example, TSO or an executing batch job) can have SAF propagate the security information associated with the session by omitting the information on the JOB statement for the job. If SAF cannot determine the group or security label information from the current session, SAF uses the default information in the RACF profile. However, SAF does not propagate security information to a job that enters the system from an RJE work station or a physical card reader.

## Propagating security information across a network

To allow users to submit jobs without specifying security information such as a user ID, JES propagates the submitter's security information when the information is omitted. For example, you can have users omit their password from their job statements if you do not want to send passwords through the network. Propagation also occurs when you use the /\*XMIT card to transmit a job to another node in the network. In this case, SAF passes the information that appears in the first JOB statement to the JOB statement that follows the /\*XMIT card.

If SAF is unable to verify the security information on the first JOB statement:

- If a TSO user submitted the job, SAF passes the TSO user's security information to the second JOB statement.
- If the job entered the system through a local card reader, SAF uses a default user ID for propagation purposes. (For information about the default user ID, see [“Understanding default user IDs” on page 480](#).)

[“How JES sends security information” on page 481](#) describes what security information RACF propagates for NJE jobs.

## Controlling the use of job names

You can use profiles in the JESJOBS class to control which job names users can submit, cancel, or modify.

Because most installations have many jobs and users, it might not be practical to define all of the profiles needed to authorize every job name. A more reasonable approach would be to restrict the use of only certain job names.

You can specify which job names can be entered from a specific device. For more information, see [“Conditional access lists for general resource profiles” on page 193](#) and the note in Step “4” on page 462 of [“Controlling who can submit jobs by job name” on page 462](#).

### Notes:

1. TSO installation exit IKJEFF53 must be modified to become a dummy exit when the JESJOBS class is active. See [z/OS TSO/E Customization](#) for specific information.
2. At least one generic profile with a universal access of READ is required for the TSO SUBMIT command when the JESJOBS class is active. A generic profile is not required for the TSO CANCEL command. However, without a generic profile, only the owner of a job or those granted access through RACF profiles can use the CANCEL command on data sets associated with that job.
3. The JESJOBS class does not apply to jobs submitted via RJE/RJP consoles. Refer to [“Remote workstations \(RJP/RJE consoles\)” on page 491](#).

## Controlling who can submit jobs by job name

To control who can submit jobs by job name, perform the following steps:

1. Ask your TSO system programmer to ensure that TSO installation exit IKJEFF53 checks if the JESSPOOL or JESJOBS class is active, and if either is active, returns to the caller with no action. For specific information, see [z/OS TSO/E Customization](#).

**Note:** SYS1.SAMPLIB contains a sample of such an exit.

2. Define at least one profile with a universal access of READ to allow users to submit jobs when the JESJOBS class is activated:

```
RDEFINE JESJOBS ** UACC(READ)
```

**Note:** This example assumes that a SETROPTS GENERIC(JESJOBS) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

3. Define profiles with UACC(NONE) for the job names you want to protect.

```
RDEFINE JESJOBS SUBMIT.nodename.jobname.userid UACC(NONE)
```

where:

***nodename***

is your local node name.

**Note:** It is recommended that you define a profile in the RACFVARS class named &RACLNDE, and use &RACLNDE for all profiles in the JESJOBS class.

***jobname***

is the name of the job specified on the JOB statement.

***userid***

is the user ID under which the job is to execute (either the USER operand on the JOB statement or the propagated user ID).

For example, the following command would prevent any user from submitting jobs whose job names begin with PAYROLL.

```
RDEFINE JESJOBS SUBMIT.*.PAYROLL*. * UACC(NONE)
```

4. To allow users to submit jobs protected by the profile, give them READ access authority:

```
PERMIT SUBMIT.*.PAYROLL*. * CLASS(JESJOBS) ID(PAYGROUP) ACCESS(READ)
```

**Note:** By denying a user sufficient access to a SUBMIT profile, you can prevent that user from submitting jobs protected by the profile *even if that user knows the password or is an authorized surrogate user*.

For example, the following profile would prevent jobs from being submitted with USER01 as the user ID:

```
RDEFINE JESJOBS SUBMIT.*.*.USER01 UACC(NONE)
```

You can also provide conditional access to the job name, depending on the class and ID of the port of entry (POE) associated with the submitter of the job. The class name you would use is determined by what the submitter is. For a regular submission from a TSO logon session, the submitter's POE is a terminal ID and the class name is TERMINAL. The submitter's POE can also be a JESINPUT device when the submitter of the job is another job.

Making use of the job name conditional on the JESINPUT device is not recommended because this is very much dependent on what type of job was submitted. If the submitting job is a local job, its JESINPUT POE would be an internal reader, a local card reader, or an RJE reader.

However, if the submitting job is an NJE job (for example, from another JES node), its JESINPUT POE would be the node name. This uncertainty can make the use of WHEN(JESINPUT) for the JESJOBS class difficult. Therefore, you should be careful if you decide to use it.

For example, you can allow a user to submit a job only from a certain terminal ID by specifying the WHEN(TERMINAL) operand on the PERMIT command as follows:

```
PERMIT SUBMIT.*.PAYROLL*.* CLASS(JESJOBS) ID(USER01)
ACCESS(READ) WHEN(TERMINAL(terminal-ID))
```

where *terminal-ID* is the terminal to which the submitter is logged on.

5. When you are ready to use the JESJOBS class to control who can submit jobs, activate the JESJOBS class:

```
SETOPTS CLASSACT(JESJOBS)
```

**Note:** If you activate this class and create no profiles for it, users cannot submit batch jobs.

## Controlling who can register a job to a job group

JES2 supports JES3 DJC (//\*NET) semantics by creating a //\*NET version of a JES2 JOBGROUP. The job group is created the first time a unique NETID is encountered on a //\*NET card. The name of the job group will be the value specified on the NETID parameter. Subsequent jobs specifying the same NETID will be associated with this job group. Job group security credentials such as USER/PASSWORD come from the job causing the job group creation. Once the job group is created, subsequent jobs with the same NETID parameter value will be associated with that job group. The rest of this section pertains to JEC and DJC job group use.

In JES2, job groups allow installations the ability to submit a series of related batch jobs and defining the sequence and conditions under which each job will execute. The job group owns certain resources and is authenticated using the same process as normal batch jobs. This includes checking profiles such as the JESJOBS SUBMIT profile.

When a batch job is submitted that will be registered to a job group, a check is made while the job is converting to validate the jobs access to the job group. If the userid that owns the job group is the same as the userid that owns the batch job, then no additional validation is done (no profiles are checked). If the userids are not the same, then an auth check is made. To control who can register a job to a job group that they do not own, perform the following steps:

1. Define at least one profile with a universal access of READ to allow users to register jobs to job groups when the JESJOBS class is activated:

```
RDEFINE JESJOBS GROUPREG.** UACC(READ)
```

**Note:** This example assumes that a SETROPTS GENERIC(JESJOBS) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

2. Define profiles with UACC(NONE) for the job groups you want to protect.

```
RDEFINE JESJOBS GROUPREG.nodename.groupname.userid
UACC(NONE)
```

where:

### nodename

The name of the node on which the group registration will occur.

**Note:** It is recommended that you define a profile in the RACFVARS class named &RACLNDE, and use &RACLNDE for all profiles in the JESJOBS class.

### groupname

The name of the job group to which the batch job is desiring to register userid - The user ID associated with group being registered to.



For example, the following command would prevent any user, other than the owner of the job group, from registering a job to a job group whose names begin with PAYROLL.

```
RDEFINE JESJOBS GROUPREG.*.PAYROLL*. UACC(NONE)
```

3. To allow users to submit jobs protected by the profile, give them READ access authority:

```
PERMIT GROUPREG.*.PAYROLL*. CLASS(JESJOBS) ID(PAYGROUP)
ACCESS(READ)
```

## Controlling who can cancel jobs by job name

Users are always authorized to cancel jobs that they have submitted. Using RACF, you can control who can use the TSO CANCEL command to cancel jobs, depending on the job names. To do this, perform the following steps:

1. Ask your TSO system programmer to change TSO installation exit IKJEFF53 to become a dummy exit. For specific information, see [z/OS TSO/E Customization](#).
2. Define profiles for the job names you want to protect:

```
RDEFINE JESJOBS CANCEL.nodename.userid.jobname UACC(NONE)
```

**Note:** The qualifiers for CANCEL profiles have the same meaning as for SUBMIT profiles. However, the *jobname* and *userid* qualifiers are reversed in CANCEL and SUBMIT profiles. This is because of the expected use of the profiles:

- It is likely that many users would submit jobs having common job names, with certain exceptions. For example, the following profiles would allow many users to submit jobs whose names begin with PAYROLL, except when those jobs run with BEN's authority:

```
RDEFINE JESJOBS SUBMIT.*.PAYROLL*. UACC(READ)
RDEFINE JESJOBS SUBMIT.*.PAYROLL*.BEN UACC(NONE)
```

- It is likely that one user would give another the authority to cancel all of the first user's jobs, with certain exceptions. For example, the following profiles would allow JOE the authority to cancel BEN's jobs, except for his PAYROLL jobs:

```
RDEFINE JESJOBS CANCEL.*.BEN.* UACC(NONE)
PERMIT CANCEL.*.BEN.* CLASS(JESJOBS) ID(JOE) ACCESS(ALTER)
RDEFINE JESJOBS CANCEL.*.BEN.PAYROLL* UACC(NONE)
```

- These examples assume that a SETROPTS GENERIC(JESJOBS) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.
3. Give users the appropriate access authority:

```
PERMIT CANCEL.*.PAYROLL* CLASS(JESJOBS) ID(PAYGROUP) ACCESS(ALTER)
```

Users must have ALTER access authority to issue the CANCEL command for the job.

4. If the JESJOBS class is not already active, activate the JESJOBS class:

```
SETROPTS CLASSACT(JESJOBS)
```

## Controlling job notifications

When submitting job through the JES2 internal reader, application may request JES2 to send a notification when job terminates. This function is described in section [Requesting job notification in z/OS JES Application Programming](#). Using RACF, you can control which users can request such job notification by defining a profile in JESJOBS class. The format of the RACF profile name for job notification is:

```
JOBNFY.nodename.jobclass.jobname
```

where:



***nodename***

The name of the node on which the job is submitted.

***jobclass***

The class of the submitted job.

***jobname***

The job name of the submitted job.

The user submitting the job must have at least READ access to the profile to be allowed to request job notification.

If no matching profile is found, the job notification request is allowed.

Job notification requests can be explicitly disallowed by creating a profile with access NONE for public (UACC) or for specific user(s).

## Controlling job class usage

An installation can control job class usage by granting access based on the submitter's profile, or based on the owner's profile, or both.

The control is based on the presence or absence of two FACILITY class profiles:

- If the profile JES.JOBCLASS.OWNER is defined in the FACILITY class, job class profiles for owners are enforced.
- If the profile JES.JOBCLASS.SUBMITTER is defined in the FACILITY class, job class profiles for submitters are enforced.
- If both profiles are defined in the FACILITY class, a job class must pass both sets of profiles before it is considered valid.
- If neither profile is defined in the FACILITY class, any job class can be used.

The job class profiles are of the form:

```
JOBCLASS.localnodeid.jobclass.jobname
```

where:

***localnodeid***

Is the name of the node on which the job is located

***jobclass***

Is the 1 - 8 character job class that is being controlled

***jobname***

Is the name that is in the name field of the JOB statement

The JOBCLASS-prefixed resources must be defined in the JESJOBS class.

## Controlling who can modify job attributes using the Job Modify SSI 85

The Job Modify SSI 85 can be used to modify a variety of job attributes. Resources in the JESJOBS class control who can use the functions of the Job Modify SSI 85. [Table 32 on page 465](#) shows the format of these resources.

<i>Table 32. Resource names in the JESJOBS class for the Job Modify SSI</i>		
<b>SSI action</b>	<b>JESJOBS resource</b>	<b>Access required</b>
Cancel	CANCEL.nodename.userid.jobname	ALTER
Hold	HOLD.nodename.userid.jobname	UPDATE
Modify	MODIFY.nodename.userid.jobname	UPDATE
Purge	PURGE.nodename.userid.jobname	ALTER

Table 32. Resource names in the JESJOBS class for the Job Modify SSI (continued)

SSI action	JESJOBS resource	Access required
Release	RELEASE. <i>nodename.userid.jobname</i>	UPDATE
Reroute execution	REROUTE. <i>nodename.userid.jobname</i>	UPDATE
Restart	RESTART. <i>nodename.userid.jobname</i>	CONTROL
Spin	SPIN. <i>nodename.userid.jobname</i>	CONTROL
Start	START. <i>nodename.userid.jobname</i>	CONTROL

To control who can modify job attributes using the Job Modify SSI, perform the following steps:

1. Ask your TSO system programmer to change TSO installation exit IKJEFF53 to a dummy exit. For information, see [z/OS TSO/E Customization](#).
2. Define profiles for the job names that you want to protect. For example:

```
RDEFINE JESJOBS MODIFY.nodename.userid.jobname UACC(NONE)
```

3. Give users the appropriate access authority. For example:

```
PERMIT MODIFY.*.*.PAYROLL* CLASS(JESJOBS) ID(PAYGROUP) ACCESS(UPDATE)
```

4. If the JESJOBS class is not already active, activate it:

```
SETROPTS CLASSACT(JESJOBS)
```

**Note:** Be sure that you do not have JESJOBS profiles that specify generic job names or user IDs, and that provide access higher than Read. If you created such profiles on a system earlier than z/OS V2R1, the profiles only affected the ability of users to cancel or submit jobs. But on a z/OS V2R1 system, those profiles might have the unintended effect of allowing users to use the Job Modify SSI functions to modify the attributes of a job.

To avoid this unplanned access, look for JESJOBS profiles that match the resource names listed in [Table 32 on page 465](#). You can use the SEARCH command to list all of the profiles in the JESJOBS class:

```
SEARCH CLASS(JESJOBS)
```

If you find any profiles that match the resource names listed in [Table 32 on page 465](#), use the RLIST command to list information about each of the profiles found, and ensure that the UACC, access list, and audit options are appropriate.

```
RLIST JESJOBS profile_name
```

## Allowing a TSO user to cancel all jobs originating from local nodes

To allow a TSO user to cancel all jobs that originate on nodes you treat as local nodes, do the following:

1. Define a profile named &RACLNDE in the RACFVARS class, specifying on the ADDMEM operand which nodes are to be treated as local:

```
RDEFINE RACFVARS &RACLNDE UACC(NONE) ADDMEM(POKMVS1 POKMVS2)
```

UACC(NONE) is recommended to protect the &RACLNDE profile itself.

2. Define a profile in the JESJOBS class as follows:

```
RDEFINE JESJOBS CANCEL.&RACLNDE.*.* UACC(NONE)
```

This example assumes that a SETROPTS GENERIC(*classname*) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

3. Give the appropriate access to the TSO user:

```
PERMIT CANCEL.&RACLNDE.*.* CLASS(JESJOBS) ID(USER1) ACCESS(ALTER)
```

If there are any other JESJOBS resources that begin with CANCEL, you might also need to permit users appropriate access to those.

4. If you have not already done so, activate the JESJOBS and RACFVARS classes:

```
SETROPTS CLASSACT(JESJOBS RACFVARS)
```

5. Refresh SETROPTS RACLIST processing for the RACFVARS class for the change to take effect:

```
SETROPTS RACLIST(RACFVARS) REFRESH
```

If, later, you decide that node POKMVS2 should no longer be treated as a local node, do the following:

```
RALTER RACFVARS &RACLNDE DELMEM(POKMVS2)
SETROPTS RACLIST(RACFVARS) REFRESH
SETROPTS GENERIC(JESJOBS) REFRESH
```

Also, be sure to issue the SETROPTS RACLIST REFRESH or GENERIC REFRESH commands for any classes that contain profiles that use the RACFVARS value affected by your change.

If, later, you decide that USER2 should also be allowed to cancel local jobs, do the following:

```
PERMIT CANCEL.&RACLNDE.*.* CLASS(JESJOBS) ID(USER2) ACCESS(ALTER)
SETROPTS GENERIC(JESJOBS) REFRESH
```

## Surrogate job submission

You can allow the use of surrogate users. A surrogate user is a RACF-defined user who has been authorized to submit jobs on behalf of another user (the original user) without having to specify the original user's password. Jobs submitted by the surrogate user execute with the identity of the original user. This can be useful when a person is assuming a production workload for someone going on vacation or a leave of absence.

For information on setting up surrogate users, see [“Allowing surrogate job submission” on page 455](#).

## Authorizing the use of input sources

You can use RACF to limit which sources of input are valid for job submission, including RJP workstations, device readers, nodes, and internal readers. For example, you might want to prevent certain users from entering jobs from a particular RJP workstation.

To authorize the submission of work from specific input sources, perform the following steps:

1. Ask your JES system programmer for the following information:
  - The name of the device. This is described in the topic on authorizing the use of input sources in [z/OS JES2 Initialization and Tuning Guide](#).
  - The user ID or group name of the users you want to authorize or restrict.
  - The universal access authority to associate with each device. Valid access authorities for input devices are:

### NONE

Specifies that the input device can be used only by those users explicitly permitted through the access list.

### READ

Specifies the minimum authority required to use the input source.

2. Define a profile for each input source, as follows:

```
RDEFINE JESINPUT source-name UACC(NONE)
```

3. It is *strongly recommended* that you create a profile with a UACC of READ for all JES input sources that are otherwise not defined:

```
RDEFINE JESINPUT ** UACC(READ)
```

This example assumes that a SETROPTS GENERIC(JESINPUT) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

If you do not, users can access only JES input sources to which they (or their groups) are explicitly authorized.

4. For each protected input source, grant access to the users or groups who need to use it:

```
PERMIT source-name CLASS(JESINPUT) ID(user-or-group) ACCESS(READ)
```

5. When you are ready to start using the protection provided by the profiles you have created, activate the JESINPUT class:

```
SETROPTS CLASSACT(JESINPUT) REFRESH
```

If you activate this class and create no profiles for it, users cannot submit batch jobs.

## Authorizing network jobs and SYSOUT (NJE)

This topic contains information about how to use RACF to ensure that all work entering or leaving your node complies with your installation's security policy. You can control:

- Jobs and data received from other nodes in a network, as long as the inbound job or data includes a standard NJE header. This includes jobs or data from an RSCS node.
- The extent of security validation performed at your node.
- Jobs and data destined for other nodes in a network.

JES does not validate work passing through your node on its way to another node in the network, but it does protect the work from unauthorized access while the work is temporarily stored on spool at your node.

To provide security for network job entry (NJE), activate the NODES class (for inbound work) or the WRITER class (for outbound work), and define the profiles needed to enforce your installation's NJE security policy. Define the appropriate security labels and ensure that compatible SECLABEL system options (using the SETROPTS command) are active on all member nodes in the NJE network.

Consult with your JES system programmer to determine what type of protection is needed for your NJE network and to obtain the information you need to correctly define these profiles. Together, you should decide whether you want to protect inbound work, outbound work, or both. For inbound work, decide whether you want to protect jobs, SYSOUT, or both. You must also determine which users and groups of users can submit work and which security labels are valid for processing on your node.

## Authorizing inbound work

The following list outlines the topics discussed here:

- [“Understanding NODES profiles” on page 469](#) explains how the values you select for the NODES profile determine what type of work and which users or security labels you want RACF to validate.
- [“Understanding mixed security environments” on page 473](#) explains how different levels of JES and RACF affect security processing.
- [“Authorizing jobs” on page 473](#) explains how you can use RACF to validate inbound jobs from other nodes in a network.
- [“Controlling user ID propagation in a local environment” on page 457](#) explains how you can use RACF to validate inbound jobs from other nodes in a network when the installation does not use the NODES profile.

- “[Using submitter information during job verification](#)” on page 475 explains how you can use RACF to validate inbound jobs using the submitter's security information from the NJE environment.
- “[Authorizing SYSOUT](#)” on page 476 explains how you can use RACF to validate inbound job output (SYSOUT) from other nodes in a network.
- “[Validating SYSOUT based on the submitter](#)” on page 477 explains how RACF can be used to validate inbound work using the submitter's security information instead of the owner's security information.
- “[Translating security information](#)” on page 478 explains how RACF can be used to replace inbound user IDs, group names, or security labels with locally defined values.
- “[Understanding default user IDs](#)” on page 480 describes how JES handles security information for work from incompatible nodes and explains how JES handles security information that belongs to *store-and-forward* work. This topic also describes how you can use RACF to manipulate default security information.
- “[How JES sends security information](#)” on page 481 explains where JES obtains missing security information for NJE work.
- “[Defining profiles in the NODES class](#)” on page 481 shows you how to set up the protection defined in the profiles and how to activate not only the NODES class but also the SETROPTS RACLIST processing for the class.
- “[Defining nodes as local input sources](#)” on page 482 explains how you can use RACF to treat SYSOUT from another node as if the SYSOUT originated at the home node.

## Understanding NODES profiles

You can use profiles in the NODES class to control how RACF validates inbound work on an NJE network. As with other RACF profiles, a NODES profile consists of a profile name, a profile class, a universal access authority, and an ADDMEM value. The profile name is a three-part identifier that indicates the origin of the work and the type of security information that you want to validate. The universal access authority determines the actions that RACF performs on the inbound work. This information is described in [Table 33 on page 473](#) and [Table 34 on page 477](#).

### Notes:

1. Access lists do not apply to NODES class profiles. The ADDMEM value is used to translate to locally defined values.
2. During a RACROUTE REQUEST=VERIFY call, if the user ID has the required access to the corresponding NODES class profile for user ID translation, the user ID session type (TOKSTYP) can be changed to TOKTSO. This behavior can impact the use of conditional access lists that use TOKSTYP and TOKPOEX to determine which class to use to check the list.

A NODES profile name has the following format:

```
nodename.worktype.name
```

Where:

### ***nodename***

Is the name of the node from which you expect inbound work. For jobs, this is the submitting node. For SYSOUT, this is the execution node.

### **Note:**

1. If &SUSER is specified as an ADDMEM value in a profile that controls SYSOUT, a second check is done in which *nodename* is the submitting node.
2. If &DFLTGRP is specified as an ADDMEM value in a profile that deals with groups (either jobs or SYSOUT), the user's default group is used.
3. It is recommended that you define a profile in the RACFVARS class named &RACLNDE, and use &RACLNDE for all nodes that are considered to be local to your system. For more information, see “[Setting up NODES profiles](#)” on page 471.

**worktype**

Is the type of work to be controlled by the profile.

Notice that the last character, J or S, indicates the type of work to be validated. J indicates jobs; S indicates SYSOUT.

**RUSER**

Controls commands originating from NJE nodes. The *nodename* is used as the name on the third qualifier.

**USERJ**

Controls jobs by the user ID specified on the third qualifier. The job is controlled by who the submitter is. This type of profile is also used to determine the amount of trust the job has. For details, see [“Understanding mixed security environments” on page 473](#).

**USERS**

Controls SYSOUT by the user ID specified on the third qualifier. The SYSOUT is controlled by who the owner is. This type of profile is also used to determine the amount of trust the SYSOUT has. For details, see [“Understanding mixed security environments” on page 473](#).

**GROUPJ**

Controls jobs by the group name that is specified on the third qualifier.

**GROUPS**

Controls SYSOUT by the group name specified on the third qualifier.

**SECLJ**

Controls jobs by the security label that is specified on the third qualifier.

**SECLS**

Controls SYSOUT by the security label specified on the third qualifier.

For example, a value of USERJ specifies that you want RACF to use the profile to validate inbound jobs; a value of USERS specifies that you want RACF to use the profile to validate inbound SYSOUT.

**name**

Is the actual user ID, group name, or security label you want validated. If you are using NODES profiles to allow the use of these input values, you must either define these values in your RACF database or use the ADDMEM operand to translate them into acceptable values for your system. For jobs, the submitter information is substituted. For SYSOUT, the owner information is used. (See [“Understanding mixed security environments” on page 473](#).)

For example, the following profile controls whether jobs coming from user ID WAYNE at node BERMUDA can be executed here:

```
BERMUDA.USERJ.WAYNE
```

You can optionally associate a local user ID with user ID WAYNE by specifying the user ID on the ADDMEM operand.

You can specify generic characters in the profile name to control a wider range of work. For example, if you place an asterisk in place of the *nodename* value, RACF performs the requested type of validation for work from all nodes in the network (unless a more specific profile exists). Examples of generic profiles in the NODES class are shown in this topic. For more information, see [“Choosing between discrete and generic profiles in general resource classes” on page 187](#).

If you installed RACF and did not activate the NODES class, JES validates jobs and SYSOUT in the following manner:

- JES runs only those jobs that are destined for your node and that have a valid user ID and password on the job card if BATCHALLRACF is active. If BATCHALLRACF is not active, the job can run without a RACF user ID.
- A security label of SYSHIGH is assigned to all SYSOUT destined for your node (if security labels are being used) and can be printed only on those devices that are permitted to SYSHIGH data. JES assigns the default user ID to this SYSOUT. For information about default user IDs, see [“Understanding default user IDs” on page 480](#).

- All work that is destined for another node remains unchanged.

If you choose to activate the NODES class, you must gather information from your JES system programmer so that you can set up profiles to control the work entering your system. The following sections identify the appropriate values for each type of work.

## Setting up NODES profiles

To set up NODES profiles, you must activate the RACFVARS class first, issue SETROPTS RACLIST, and, if you are going to define generics, make sure that SETROPTS GENERIC is active for the RACFVARS class. You should consider the following approach to setting up NODES profiles:

1. Define a profile for each node for which you want to control inbound work. (If you have several nodes that you are treating identically, consider creating RACFVARS profiles and using the RACF variables in NODES profile names. This can reduce the number of NODES profiles that you must maintain.)
2. Define a *top* generic profile to control all work not controlled by more specific NODES profiles.
3. For each node, define profiles with USER $\bar{x}$ , SECL $\bar{x}$  or GROUP $\bar{x}$  qualifiers only if you want to:
  - Prevent work with the specified user ID, security label, or group name from entering your node (determined by the UACC of the profile).
  - Translate the specified user ID, security label, or group name to a local value (specify the ADDMEM operand to do this).
4. Define the local node or nodes in the &RACLNDE profile in the RACFVARS class. Enter:

```
RDEFINE RACFVARS &RACLNDE ADDMEM(nodea nodeb...)
```

In effect, this allows security information to be accepted for verification without the use of NODES profiles. That is, the information is used as passed because it is considered local.

For SYSOUT, this allows the owner information to be used without a NODES lookup, or automatically allows the submitter to become the SYSOUT owner when &SUSER is used. (See [“How SYSOUT requests are verified”](#) on page 459.)

For jobs, this allows the special JES2 pre-execution reroute case to use the information as passed without translation, and allows the spool unload and reload of jobs to propagate the information automatically without requiring NODES profiles. See [“Defining nodes as local input sources”](#) on page 482.

**Note:** Group names are not propagated when the node is defined to &RACLNDE. The default group of the execution user is used.

5. If an inbound job has been submitted as a surrogate job on its originating system (see [“Allowing surrogate job submission”](#) on page 455), the PASSWORD parameter is not specified on its JOB statement. Therefore, you must specify UACC(CONTROL) or higher in the NODES profile controlling such jobs, or UACC(UPDATE) or higher if the job is from an uplevel node to prevent requiring password verification. (See [“Understanding mixed security environments”](#) on page 473.)

## Unknown, blank, and undefined security labels

When you receive a SYSOUT data set with an unknown security label (consisting of hexadecimal zeros) or a blank security label while the SECLABEL class is active on the receiving node, RACF assigns a security label called RACSLUNK to the SYSOUT data set. When you receive a SYSOUT data set with a security label that is not defined on your system, the data set keeps its security label and RACSLUNK label is *not* assigned.

While the SECLABEL class is active, no users are authorized to access SYSOUT data sets with unknown, blank, or undefined labels, until you take *one* of the following actions:

- Define the RACSLUNK or undefined label to RACF as a security label in the SECLABEL class and authorize users to access it.
- Translate the RACSLUNK or undefined label to a defined security label on your node using the ADDMEM value of a NODES class profile. (See [“Understanding NODES profiles”](#) on page 469.) **Tip:** Translate an



undefined label to a defined security label that uses the same level and category authorizations, if one already exists.

With either action, a user must be logged on with the appropriate security label to access the SYSOUT data set.

### ***Learning which NODES profiles are used***

For an exercise to learn which NODES profiles are used, see [Figure 40 on page 472](#).

Assume the following profiles:

(1)	POKMVS.SECLJ.A	ADDMEM(ALPHA)	UACC(READ)
(2)	POKMVS.SECLS.A	ADDMEM(ALPHA)	UACC(READ)
(3)	POKMVS.SECL%.A		UACC(NONE) /*never used*/
(4)	POKMVS.USERJ.JOHN	ADDMEM(JOHNYY)	UACC(UPDATE)
(5)	POKMVS.USERS.JOHN	ADDMEM(JOHNYY)	UACC(UPDATE)
(6)	POKMVS.USER%.JOHN		UACC(NONE) /*never used*/
(7)	POKMVS.USER%.TOM		UACC(NONE)
(8)	POKMVS.USER%.*	ADDMEM(NONAME)	UACC(UPDATE)
(9)	POKMVS.*.*	ADDMEM(X)	UACC(READ)
(10a)	*		UACC(NONE)
(10b)	*.USERJ.*		UACC(NONE)

1. If a job is submitted from user JOHN at node POKMVS with SECLABEL A, profiles (1), (4), and (9) are used.
  - Profile (4) translates the user ID to JOHNNY.
  - Profile (9) translates the group name to X. (There is no profile with the GROUP operand.)
  - Profile (1) translates the SECLABEL to ALPHA.
2. Profile (3) would never be used because profiles (1) and (2) are discrete profiles that cover all work from node POKMVS that has security label A.  
 Profile (6) would never be used because profiles (4) and (5) are discrete profiles that cover all work from user JOHN at node POKMVS.
3. If jobs or SYSOUT come in from user TOM at POKMVS, profile (7) fails the job or purges the output.
4. If a job comes in from anyone other than JOHN or TOM at POKMVS, with SECLABEL A, profiles (1), (8), and (9) are used.
  - Profile (8) translates the user ID to NONAME.
  - Profile (9) translates the group name to X (there is no profile with the GROUP operand.)
  - Profile (1) translates the SECLABEL to ALPHA.

**Note:** Profile (8) translates many user IDs to one. You might do this to create a guest user ID that can be used by any otherwise unknown user coming in from POKMVS. With such a user ID, you can allow people from POKMVS to access certain resources without having to give each of them a user ID on your system.
5. Because there is no POKMVS profile with the GROUP operand, profile (9) is the generic that is used to translate group names. Therefore all jobs and SYSOUT that come from POKMVS get group X. (If profile (9) did not have ADDMEM specified, there would be no translation of group names.)  
 Also, all security labels from POKMVS, except security label A, are translated to X.
6. Profile (10a) fails all NJE jobs and SYSOUT for any other user, group, or security label that is not covered by a more specific NODES profile. If you want to have just default control for any NJE jobs, and not control SYSOUT, use profile (10b) instead.

*Figure 40. Which NODES profiles are used?*



## Understanding mixed security environments

Your network might be a mixed environment, that is, it can contain nodes in which different levels of JES and RACF, or non-JES systems, are installed. Networking in a mixed environment causes JES and RACF to validate work differently in some cases. For example, certain security information, such as security labels, might not be sent with work from some systems. The following list categorizes the various environments into three groups:

- Uplevel security systems
  - Systems running z/OS where RACF is installed and active.
- Downlevel security systems
  - Systems running z/OS where RACF is not used but another security product is being used.
  - Systems not running z/OS.
- Default security systems
  - Systems running z/OS where no security product is being used.

The terms (*uplevel*, *downlevel*, and *default*) describe the security systems of the source nodes. This tells JES and RACF how much of the security information has been verified at the source. They are used to determine how much the receiving node trusts the source nodes.

For jobs, the amount of trust determines under what circumstances RACF propagates the submitter. For SYSOUT, it determines under what circumstances RACF accepts the owner information. NJE uses the NODES profile UACC to determine level of trust. Use these definitions when this topic refers to uplevel, downlevel, and default security systems. See [Table 33 on page 473](#) and [Table 34 on page 477](#).

## Authorizing jobs

You can control which network jobs are authorized for processing at your installation on the basis of submitter's user ID, group name, or security label associated with the inbound job.

To authorize or restrict jobs entering your system from another node, define a NODES profile that specifies the criteria on which jobs are accepted. Ask your JES system programmer for the following:

- The node names from which you expect jobs
- The user IDs or group names from which you expect jobs
- The security labels that you should accept
- The universal access authority, which determines how JES3 processes the job. [Table 33 on page 473](#) lists the universal access authorities you can assign and defines the validation that RACF performs.

Table 33. NODES class operands and the UACC meaning for inbound jobs				
Type of check (operand)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
User ID (USERJ)	Fails the job.	Verifies all security information available including password validation.	<p>If the job is from an uplevel node, that is, if a non-default valid security token is passed, propagates the submitter's or translated security information as the owning security information without password validation. See <a href="#">“Understanding mixed security environments” on page 473</a>.</p> <p>If the job is from a default or downlevel node, processing is the same as for a UACC of READ.</p>	Same as UPDATE, but default security or downlevel information is allowed. CONTROL allows a downlevel system to send jobs to your node without passwords. RACF does not validate passwords. See <a href="#">“Understanding mixed security environments” on page 473</a> .

Table 33. NODES class operands and the UACC meaning for inbound jobs (continued)

Type of check (operand)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
Group name (GROUPJ)	Fails the job.	Translates group name to that specified in ADDMEM. If ADDMEM is not specified, uses the group name received.		
Security label (SECLJ)	Fails the job.	Translates the submitter's SECLABEL to that specified in ADDMEM. If ADDMEM is not specified, uses the security label received.		
<b>Note:</b> For more details on how NJE jobs are processed, see <a href="#">“Authorizing jobs” on page 473</a> and for information on surrogate job submission, see <a href="#">“Allowing surrogate job submission” on page 455</a> .				

**Note:** If no profile exists for a job when the NODES class is active or if the NODES class is inactive, RACF performs only user ID, group name, and password validation without performing any translation.

If no profile exists for a job when the NODES class is active, RACF verifies all security information available and a valid password and user ID must be specified on the job card.

You can further reduce the risk of security exposures by allowing jobs to be submitted from other nodes without requiring a password if the sending node properly validates and transmits a user's identity. You can either allow the submitter's identity (that is, the user ID and security label) to be propagated to the job or you can specify that the submitter is a surrogate submitter who can submit jobs on behalf of other users without needing a password.

For either case, you indicate in NODES class profiles which nodes are trusted to provide valid submitter identity information. You can restrict the trusted information to specified user IDs, group names, or security labels, if desired.

This submitter identity information in combination with user data on the job card is used to determine the user identity to be used for the job.

- If no user ID or password is specified on the job card, the submitter's identity is propagated to the job.<sup>13</sup>
- If a user ID but no password is specified, the user ID is allowed if the submitter is authorized as a surrogate for that user ID.<sup>13</sup>
- If both user ID and password are specified on the job card, the submitter's identity is not propagated to the job, but will still be used for JESJOBS checking. Normal password validation is performed.

Your network might be a mixed environment, that is, it can contain nodes in which different levels of JES and RACF, or non-JES systems, are installed. Networking in a mixed environment causes JES and RACF to validate work differently in some cases. For example, certain security information, such as security labels, might not be sent with work from some systems. The following list categorizes the various environments into three groups:

- Uplevel security systems
  - Systems running z/OS where RACF is installed and active.
- Downlevel security systems
  - Systems running z/OS where RACF is not used but another security product is being used.
  - Systems not running z/OS.
- Default security systems
  - Systems running z/OS where no security product is being used.

The terms (*uplevel*, *downlevel*, and *default*) describe the security systems of the source nodes. This tells JES and RACF how much of the security information has been verified at the source. They are used to determine how much the receiving node trusts the source nodes.

<sup>13</sup> In either case, if SECLABEL is specified on the job card, it is used. If not, the SECLABEL of the submitter is propagated to the job.

For jobs, the amount of trust determines under what circumstances RACF propagates the submitter. For SYSOUT, it determines under what circumstances RACF accepts the owner information. NJE uses the NODES profile UACC to determine level of trust. Use these definitions when this topic refers to uplevel, downlevel, and default security systems. See [Table 33 on page 473](#) and [Table 34 on page 477](#).

## Controlling user ID propagation in an NJE environment

If an installation does not use NODES profiles for its networking protection (in other words, it uses the RACFVARS profile &ACLNDE to treat all its remote nodes as local), propagation control is the same as for actual local jobs (see [“Controlling user ID propagation in a local environment” on page 457](#)). Otherwise, an installation wanting propagation control across the network needs to define one or more NODES profiles, possibly with a RACFVARS profile, similar to the following:

```
RDEFINE RACFVARS &PROPCON ADDMEM(USER25 USER42 USER19 USER22 USER11)
RDEFINE NODES *.USERJ.&PROPCON UACC(READ)
```

Be sure that both the RACFVARS and the NODES classes are active and generics are active for the NODES class, that you bring the classes in storage using SETROPTS RACLIST, and that you issue a SETROPTS RACLIST REFRESH after you define the two profiles. You need these profiles on every receiving system where you want propagation to be controlled. Every user ID that has a PROPCNTL entry on that system should be included in the ADDMEM list for &PROPCON.

With this setup, if a user ID is *not* coded on the job card, the job is routed to another node, and the submitting user ID is a member of &PROPCON on the receiving side, the job runs with the undefined user ID (default of ++++++++), assuming SETROPTS (JES (BATCHALLRACF)) and SETROPTS (JES (XBALLRACF)) are not in effect.

Note that a better-fitting NODES profile with a higher UACC negates this protection. For example, if in addition to the two preceding profiles you have a NODES profile NODEX.USERJ.CICS1 with UACC(CONTROL), even if CICS1 is a member of &PROPCON, an incoming job submitted by CICS1 runs as CICS1.

For more information about why you would want user propagation controlled, see [“Controlling user ID propagation in a local environment” on page 457](#).

## Using submitter information during job verification

With NJE jobs, as with local jobs, RACF makes several checks based on the submitter of a job. The submitter information is used during SURROGATE checking (see [“Allowing surrogate job submission” on page 455](#)). It helps to ensure that the security label of the job takes precedence over the security level of the submitter. RACF does this check when security labels are being used, taking into consideration the setting of the SETROPTS MLS option. Submitter information is also used for JESJOBS checking during submission of a job (see [“Controlling who can submit jobs by job name” on page 462](#)).

With local jobs, the submitter information is used as it is passed to RACF. Normally, it is assumed to be valid. During any of the submitter checks, however, it is subject to reverification. Any incorrect information causes the specific check to fail.

With NJE jobs, the submitter information used depends on whether the submitting node is trusted. If the submitting node is trusted, the submitter information is either used as passed or translated through NODES profiles. This information is subject to reverification during any submit check that might be performed. This is consistent with local jobs.

If the submitting node is not trusted, the submitter information cannot be used as passed to RACF. When the submitter is identified by token information, the submitter is then represented by the NJE unknown user (that is, no user ID). The original submitter information is discarded. This allows UACC access to the checks made on behalf of the submitter, such as SURROGATE and JESJOBS.

RACF validates an NJE batch job based on the submitter node and submitter user ID in a USERJ profile and on the submitter node and submitter group name in a GROUPJ profile. If there is an ADDMEM value, the NJE batch job submitter user ID is translated to the ADDMEM value before the validation checks are made.

When RACF determines that a job is not from a trusted node, the submitter user ID of the NJE batch job is set to the NJE unknown user ID and the submitter group name is changed to blanks. For a job that is submitted from a trusted node, the translated submitter user ID is propagated and becomes the user ID with which the NJE batch job runs.

USERJ NODES profiles are checked before the GROUPJ NODES profiles. After successful verification based on the submitter node and user ID, GROUPJ NODES profiles are used to validate NJE batch jobs, based on the submitter node and group name. If there is an ADDMEM value, the NJE batch job submitter group name is translated to the ADDMEM value before the validation checks are made.

**Note:** If no USERJ NODES profile exists, the GROUPJ NODES profile is not checked.

A GROUPJ NODES profile can be used to fail incoming jobs based on the submitter's group by specifying UACC of NONE in the profile. A GROUPJ NODES profile can also be used to translate the submitting group to an appropriate group for the receiving system. This is done by specifying a UACC of at least READ and an appropriate ADDMEM member.

If the installation does not want incoming jobs to fail based on the groups, a special ADDMEM of &DFLTGRP can be used. This is not a RACFVARS variable. It just specifies that for jobs matching this GROUPJ profile, the resulting user's default group should be used in the verification.

**Example:**

```
RDEFINE NODES Z.GROUPJ.* UACC(READ) ADDMEM(&DFLTGRP)
```

Assuming appropriate use of USERJ profiles, all NJE batch jobs from node Z will have SURROGAT and JESJOBS checking done based on the default group of the submit user. Checking done on the execution user (assuming the submit group is propagated, that is, GROUP is not on the job card), will be done with the default group of the execution user.

## Authorizing SYSOUT

You can control the processing of SYSOUT at your installation based on the user ID, group name, or security label associated with the inbound SYSOUT. If no profile exists for an NJE SYSOUT when the NODES class is active, SYSOUT ownership cannot be assigned. See [“Understanding default user IDs” on page 480](#).

To authorize or restrict SYSOUT entering your system from another node, define NODES class profiles that identify the criteria on which SYSOUT is accepted. Ask your JES system programmer for the following:

- The node names from which you expect SYSOUT
- The user IDs, group names, and security labels from which you expect SYSOUT
- The universal access authority, which determines how JES processes the SYSOUT. RACF can assign ownership based on either the user ID and node that created the SYSOUT or the user ID and node that submitted the job that created the SYSOUT.

**Note:**

1. If the NODES profile allows the user ID to be associated with the SYSOUT, but the user security information is incorrect, an IRR808I message is issued and processing continues with the NJE unknown user as set by SETROPTS JES(NJEUSERID(*userid*)).
2. &DFLTGRP can be used for SYSOUT in the same way as in batch jobs. Specifying ADDMEM of &DFLTGRP in a GROUPS profile will cause verification to be done for the default group of the owning user ID.

[Table 34 on page 477](#) lists the universal access authorities you can assign and defines the validation that RACF performs.

Table 34. NODES class operands, UACC, and SYSOUT ownership when node is not defined to &amp;RACLNDE

Type of check (operand)	UACC			
	NONE	READ	UPDATE	CONTROL or greater
User ID (USERS)	Check of user ID and node that created SYSOUT			
	Purges the output.	If the translation value from ADDMEM is &SUSER, check submitting user ID and node.  Otherwise, assigns ownership of the output to the default NJE user ID (default is ????????).	If default or no security information is available, that is, from a downlevel or default node, processing is the same as a UACC of READ.  If security information is from an uplevel node, that is, a non-default valid security token is passed, assigns the translation value from ADDMEM to the output. When ADDMEM is not specified, ownership is assigned to the user ID that created the output. See “Understanding mixed security environments” on page 473.  If the translation value from ADDMEM is &SUSER, check submitting user ID and node.	Processing is similar to UACC(UPDATE) except RACF translates any available information from any type of security system. This allows RACF to assign local user IDs to output from downlevel systems. See “Understanding mixed security environments” on page 473.  If the translation value from ADDMEM is &SUSER, check submitting user ID and node.
	Check of submitting user ID and node (only when &SUSER is specified for ADDMEM)			
	Assigns ownership of the output to the default NJE user ID (default is ????????).	Assigns ownership of the output to the default NJE user ID (default is ????????).	Assigns ownership of the output to the default NJE user ID (default is ????????).	Assigns the translation value from ADDMEM to the output, if available.  If the translation value from ADDMEM is &SUSER, assigns the submitting user ID to the output.  Otherwise, assigns ownership of the output to the default NJE user ID (default is ????????).
	<b>Note:</b> When you specify &SUSER for ADDMEM and the submitting node is defined to &RACLNDE, ownership is assigned to the submitter. See “How SYSOUT requests are verified” on page 459.			
Group name (GROUPS)	Purges the output	Translates group name to that specified in ADDMEM. If ADDMEM is not specified, uses the group name received.		
Security label (SECLS)	Purges the output	Translates SECLABEL to that specified in ADDMEM. If ADDMEM is not specified, uses the security label received.		
<b>Note:</b>  1. If the node name is specified in the RACFVARS profile named &RACLNDE, the node is treated as a locally attached node and RACF verifies the supplied security information.  2. For more details on how NJE SYSOUT is processed, see “Authorizing SYSOUT” on page 476 and “Validating SYSOUT based on the submitter” on page 477.				

## Validating SYSOUT based on the submitter

JES normally validates SYSOUT based on the owner's security information. The owner's security information accompanies each piece of SYSOUT as it travels through the network.

You can define profiles that cause RACF to assign ownership of the SYSOUT to the submitter. For example, you can allow a user to submit a job to another node, have the job execute under another user ID, and allow the submitting user to view the output on its return.

To translate inbound SYSOUT ownership to the submitter, specify &SUSER as the value on the ADDMEM operand of the NODES profile.

This works with potentially multiple NODES profiles as follows:

First, the NODES profile is used that matches the form *execution-node*.USERS.*userid*. If the UACC is not NONE and the ADDMEM is &SUSER, a check is made to see if the submitter is set up to be the owner. If the submit node is found to be a member of the RACFVARS &RACLNDE profile, the submitter user ID and group are associated with the SYSOUT without change. This is because the submit node is considered local.

If the submit node is not local in this way, a second NODES profile that matches the form *submit-node*.USERS.*submitter-id* is used; and, if the UACC is CONTROL and there is an ADDMEM value, the submitter values are associated with the SYSOUT. If the ADDMEM value is not &SUSER, the ADDMEM value is used as the SYSOUT owner user ID.

If the ADDMEM is &SUSER, the original submitter is used as the SYSOUT owner user ID. The second NODES profile cannot be used to purge SYSOUT. The first NODES profile has already established the level of trust and the second NODES profile is used only for determining the owning user ID of the SYSOUT. A UACC of NONE on the second NODES profile assigns the ???????? user ID. For more details, see [Table 34 on page 477](#).

When associating the submitter with the SYSOUT in the non-local case, a third NODES profile can be used that matches the form *submit-node*.GROUPS.*submit-group*. If this profile exists and has an ADDMEM value, the ADDMEM value is used as the SYSOUT owner group, regardless of the UACC. Otherwise, the original submit group is associated with the SYSOUT. Verification of the SYSOUT continues with the owner values altered as described above.

## Translating security information

You can avoid having to maintain identical user IDs, group names, and security labels in RACF databases throughout a network by translating inbound user IDs, group names, and security labels into predefined values defined at your node.

Use the ADDMEM operand on the RDEFINE or RALTER command to specify the translation values for inbound security information. For example, if you want all inbound work with a security label of VERYCONF to be translated to a security label of NOLOOKAT at your system, enter:

```
RDEFINE NODES *.SECL*.VERYCONF ADDMEM(NOLOOKAT) UACC(READ)
```

### Note:

1. Specify only one value with the ADDMEM operand. If you specify multiple values, RACF stores them in the NODES profile but translates using only the last one specified.

**Restriction:** When more than one value is defined in a NODES profile, you cannot use the RLIST command to determine which value was the last one specified.

**Guideline:** If one or more values are already defined in a NODES profile, use the DELMEM operand to remove them before specifying the new value.

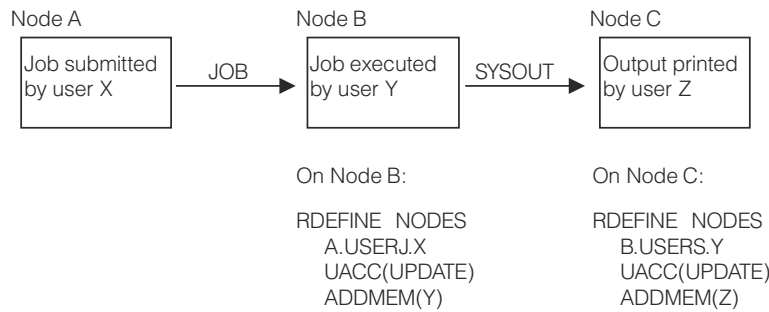
2. For jobs, an ADDMEM of &SUSER is ignored, as the NODES profile lookup for jobs automatically deals with submitter information. It would be treated as though no ADDMEM were specified for the profile. For more information on &SUSER, see [“Validating SYSOUT based on the submitter” on page 477](#).

If you do not define profiles that translate inbound user IDs, group names, and security labels, those inbound values must be defined in your RACF database or the work does not pass RACF validation.

**Note:** If the SECLABEL class is not active on your system, inbound security labels are ignored.

### Example: Simple NJE user translation

[Figure 41 on page 479](#) shows how user IDs are translated.



*Figure 41. Example: Simple NJE user translation*

User X is known as user Y on node B, and user Z on node C. In this example, user X on node A submits a job that runs on node B. The output is printed on node C under user ID Z.

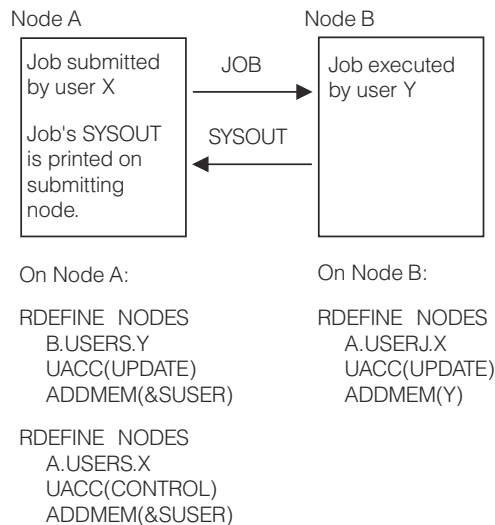
On node B, the existence of a profile named A.USERJ.X, with ADDMEM(Y) specified, causes RACF to translate the user ID of jobs from user X at node A. On node B, such jobs run as if submitted by user Y.

On node C, the existence of a profile named B.USERS.Y, with ADDMEM(Z) specified, causes RACF to translate the user ID of SYSOUT from user Y at node B. On node C, such SYSOUT can print as if submitted by user Z.

### **Example: Simple NJE user translation using &SUSER**

Figure 42 on page 479 shows how user IDs are translated when &SUSER is specified on the ADDMEM operand. This can be useful when jobs are run on a remote system, but the output is printed on the submitter's system.

**Note:** If you want, you could specify &SUSER on a third system (as in node C in [Figure 41 on page 479](#)).



*Figure 42. Example: Simple NJE user translation using &SUSER*

User X is known as user Y on node B. In this example, user X on node A submits a job that runs on node B. The output is returned to user X's home node (node A) to be printed.

On node B, the existence of a NODES profile named A.USERJ.X with UACC(UPDATE) and ADDMEM(Y) means that jobs from user X at node A are to be executed under user Y.

On node A, the existence of a NODES profile named B.USERS.Y with UACC(UPDATE) and ADDMEM(&SUSER) means that SYSOUT from user Y at node B is to be owned by the user who originally submitted the job, provided the second lookup is successful. The second lookup involves the submitter of the job (X) and the node that he submitted it from (A). Therefore, on node A, the existence of a NODES profile named A.USERS.X with UACC(CONTROL) and ADDMEM(&SUSER) means that the SYSOUT is to be owned by user X.



### Example: Trusted, semitrusted, and untrusted Nodes

Figure 43 on page 480 shows a sample NJE network in which some nodes are trusted (see “Understanding mixed security environments” on page 473), some nodes are semitrusted (verification is done on inbound work), and some nodes are not trusted (no inbound work is allowed to run).

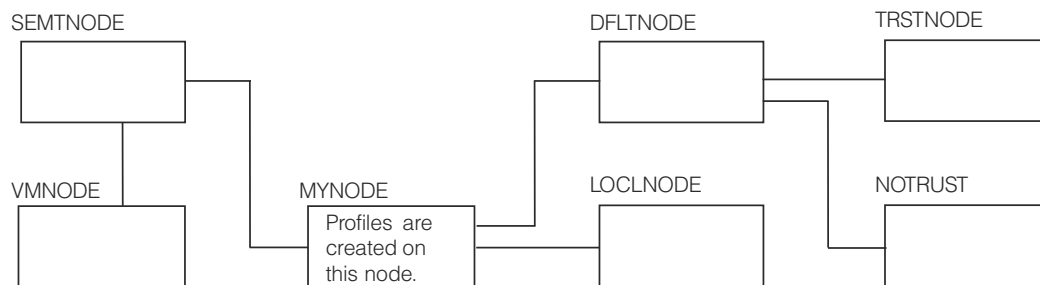


Figure 43. Example: Trusted, semitrusted, and untrusted nodes

In this example, profiles on node MYNODE control inbound work as follows:

#### Trusted nodes:

```
RDEFINE NODES TRSTNODE.USER%.* UACC(UPDATE)
RDEFINE NODES LOCLNODE.USER%.* UACC(UPDATE)
RDEFINE NODES VMNODE.USER%.* UACC(CONTROL)
```

#### Semitrusted nodes:

```
RDEFINE NODES SEMTNODE.USER%.* UACC(READ)
RDEFINE NODES DFLTNODE.USER%.* UACC(READ)
```

#### Untrusted node:

```
RDEFINE NODES NOTRUST.*.* UACC(NONE)
```

**Note:** To prevent any unknown nodes from submitting work to be done on your node, create the following profile:

```
RDEFINE NODES *.*.* UACC(NONE)
```

## Understanding default user IDs

RACF assigns a default user ID to all work that enters your node when:

- SYSOUT enters from one of the following:
  - A downlevel node
  - A default node

For more information, see “Understanding mixed security environments” on page 473.

- SYSOUT or a job enters your node, but your node is an intermediate (*store-and-forward*) node on the path to the work’s final destination. The default user ID protects work while it resides on spool awaiting transmission.
- SYSOUT enters your node when the NODES class is active and no applicable USERS profile exists.

RACF uses eight question marks (????????) as the user ID for all inbound work meeting the preceding criteria. RACF also assigns the default user ID to all *store-and-forward* work that resides temporarily at your node. The default user ID protects work while it resides on spool.

You cannot directly permit the default user ID (???????? or installation-defined) to any resources. However, you can translate the default user ID to a valid user ID if you want to process any of this type of work at your system.



You can change the ???????? user ID by using the NJEUSERID operand on the SETROPTS command:

```
SETROPTS JES(NJEUSERID(?NETWORK))
```

The user ID you specify on the NJEUSERID operand cannot be a user ID defined in the RACF database. Also, if you specify a user ID on the NJEUSERID operand, you cannot later define a user profile for that user ID. This prevents network jobs from having access to RACF-protected resources on your system.

The following example shows how to do this for jobs:

```
RDEFINE NODES nodename.USERJ.???????? UACC(READ or higher) ADDMEM(NJEJOBS)
```

The following example shows how to do this for SYSOUT:

```
RDEFINE NODES nodename.USERS.???????? UACC(UPDATE or higher) ADDMEM(NJESOUT)
```

The following example shows how to do this for both SYSOUT and jobs:

```
RDEFINE NODES nodename.USER%.???????? UACC(READ or higher) ADDMEM(NJEWORk)
```

**Note:** This example assumes that a SETROPTS GENERIC(NODES) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

You would also need to create user profiles for the translated user IDs (NJEJOBS, NJESOUT, or NJEWORk), and permit the user IDs to appropriate resource profiles (or connect them to appropriate groups).

Local jobs that enter the system without a user ID are assigned a user ID of ++++++++ (8 plus signs). You can specify which user ID to assign to such jobs by entering the following command:

```
SETROPTS JES(UNDEFINEDUSER(userid))
```

**Note:** The user ID you specify on the UNDEFINEDUSER operand cannot be a user ID defined in the RACF database. Also, if you specify a user ID on the UNDEFINEDUSER operand, you cannot later define a user profile for that user ID. This prevents undefined users from having access to RACF-protected resources on your system.

However, these user IDs can be used in JESSPOOL profile names. JES uses these names to associate an owner with the spool data, and to keep logical undefined users from accessing the data of network undefined users.

## How JES sends security information

Security information is sent from node to node in an NJE network. When a node receives a job through a network, RACF determines who submitted the job. After determining the submitting user ID, RACF can translate the submitting user ID to a valid user ID on this system if a profile on the receiving node specifies that the user ID must be translated. RACF uses the submitting user ID or its translation to supply any missing security information. Security information that RACF propagates is:

- User ID
- Password
- Security label

## Defining profiles in the NODES class

To create profiles in the NODES class, perform the following steps:

1. Ask your JES system programmer for the information needed to create the profiles. This includes the following:
  - Information for specifying profile names
  - For each profile to be created, the UACC to be specified

- For each profile to be created, the values to be translated (to be specified on the ADDMEM operand)

**Note:** You should work with your JES system programmer to determine which user or group should be specified in the OWNER field of the profiles. This user or group is responsible for maintaining the profiles.

2. Use the RDEFINE command to create the profiles. For examples, see [Figure 40 on page 472](#), [Figure 41 on page 479](#), [Figure 42 on page 479](#), and [Figure 43 on page 480](#).
3. When you are ready to start using the protection defined in the profiles, activate the NODES class and activate SETROPTS RACLIST processing for the class. You can do these two actions in one command:

```
SETROPTS CLASSACT(NODES) RACLIST(NODES)
```

**Note:**

- a. Any time you make a change to a NODES profile, you must also refresh SETROPTS RACLIST processing for the NODES class for the change to take effect.
- b. RACF does not do any logging nor issue any messages for the NODES class.

## Defining nodes as local input sources

You can use RACF to treat nodes the same as locally attached devices. To do this, use the &RACLNDE profile in the RACFVARS class to identify the nodes that you want RACF to consider as local. See [“Setting up NODES profiles” on page 471](#). A node name defined to &RACLNDE either shares your RACF database or is the value you are using to rename your node when you are changing node names.

To do this perform the following steps:

1. Ask your JES system programmer for the names of the nodes to be treated as local.
2. Define profile &RACLNDE in class RACFVARS:

```
RDEFINE RACFVARS &RACLNDE UACC(NONE)
```

3. Using the ADDMEM operand on the RALTER command, identify which nodes are to be treated as local nodes:

```
RALTER RACFVARS &RACLNDE ADDMEM(node1 node2 node3...)
```

**Note:**

- a. If you define a node as a local node, you must ensure that its RACF database is identical to the one on your node.
  - b. Because there are no defaults for &RACLNDE profiles in the RACFVARS class, you must identify your own local node using the ADDMEM operand.
4. When you are ready to start using the protection defined in the profiles, activate the RACFVARS class and activate SETROPTS RACLIST processing for the class. You can do these two actions in one command:

```
SETROPTS CLASSACT(RACFVARS) RACLIST(RACFVARS)
```

**Note:**

- a. Any time you make a change to a RACFVARS profile, you must also refresh SETROPTS RACLIST processing for the RACFVARS class for the change to take effect.
- b. This also activates other functions that are administered through the RACFVARS class.

## Authorizing outbound work

You can use the WRITER class to control whether work is authorized for transmission to a specific NJE node. To do this, create profiles in the WRITER class that have profile names with the following format:

```
jesname.NJE.nodename
```

For more information, see [“Controlling where output can be processed” on page 493](#).

## Using security labels to control writers

If both the WRITER and the SECLABEL class are active when RACF checks a writer's authority to process outbound work, RACF uses "reverse MAC" (mandatory access checking). That is, the outbound work must have a security label, and the security label of the writer must be equal to or greater than the outbound work's security label.

You can use this to limit the sensitivity of the data that writers can process. For example, if you have some writers that process low-sensitivity information, you can assign those writers a low-sensitivity security label, such as SYSLOW. This prevents sensitive work from leaving your node through those writers.

## Controlling access to spool data

You can use RACF to provide security for your spool data, including:

- Access to spool data
- Data sets dumped from spool
- Data sets restored to spool
- JESNEWS
- SYSIN data sets
- SYSOUT data sets
- SYSLOG
- Trace data sets (for JES2).

The following sections identify which spool resources you can protect, why you might want to protect each resource, and what information you must gather from your JES system programmer so that you can implement RACF protection.

## Protecting data sets on spools

You can use RACF to protect data sets that reside on spool, including spool files that JES appends to job output, such as JESNEWS. Using RACF prevent users other than the owner of a data set to read, copy, print, or delete sensitive job data.

To enable RACF protection of spool data sets, activate the JESSPOOL class:

```
SETROPTS CLASSACT(JESSPOOL)
SETROPTS GENERIC(JESSPOOL)
```

Profiles are not required in the JESSPOOL class for protection to be in effect because the default for the class is failure when no profiles exist. IBM recommends that you activate the generics for the JESSPOOL class because the profile names are system generated.

### Note:

1. When the JESSPOOL class is not active, data sets that reside on spool are not protected and could be accessed using APIs that do not require the program to be APF authorized. Products like SDSF and TSO/E will provide a level of protection for the spool data sets when the JESSPOOL class is not active, but that does not imply that the data that resides on spool cannot be accessed by any user on the system.

2. When the JESSPOOL class is active, RACF ensures that only authorized users obtain access to job data sets on spool. Authorization to job data sets is provided through RACF user profiles. If there is no profile for a data set, only the user that created the data set can access, modify, or delete it.
3. While a job is executing, RACF optionally audits actions against SYSIN and SYSOUT data sets. For SYSIN data sets, JES invokes RACF each time a SYSIN data set is allocated, opened, or deleted. For SYSOUT data sets, JES invokes RACF each time a SYSOUT data set is created, opened, deleted, or selected for output.
4. For output selection, a data set can be selected by a TSO user through the TSO OUTPUT command. A profile must exist to enable users other than the creator to access data sets using the TSO OUTPUT command.
5. External writers, which are usually started tasks that process output to special devices (such as microfiche), require at least ALTER access to the spool data sets they process. If your installation has external writers, and you activate the JESSPOOL class, you must either ensure that the external writers have ALTER access to appropriate JESSPOOL profiles, or define the external writers as a started procedure with the trusted attribute. You can define them either in the STARTED class or in the RACF started procedures table (ICHRIN03). Otherwise, the external writers cannot process output. Because external writers are installation-written programs, you are strongly recommended to avoid giving them the trusted attribute.
6. If SDSF is installed on your system, JESSPOOL profiles control which action characters and overtypeable fields users can enter on SDSF panels. For complete information on creating JESSPOOL profiles for use with SDSF, see [z/OS SDSF Operation and Customization](#).
7. SYSOUT application program interface (SAPI) applications, which are usually started tasks that process output to special devices (like microfiche), require at least UPDATE access to the spool data sets they process. If your installation has SAPI applications, and you activate the JESSPOOL class, you must either ensure that the SAPI applications have UPDATE access to appropriate JESSPOOL profiles, or define the applications as a started procedure with the trusted attribute. You can define them either in the STARTED class or in the RACF started procedures table. Otherwise, the SAPI applications cannot process output.

## Defining profiles for SYSIN and SYSOUT data sets

Activating the JESSPOOL class provides protection for SYSIN and SYSOUT data sets. However, you might want to allow specific users to see or work with the SYSIN and SYSOUT data sets created by other users. To do this, perform the following steps:

1. Create JESSPOOL profiles for the spool data sets:

```
RDEFINE JESSPOOL profile-name UACC(NONE)
```

where *profile-name* is a 6-part name with the following format:

```
local-nodename.userid.jobname.jobid.dsidentifier.name
```

where:

### ***local-nodename***

is the name of the node on which the SYSIN or SYSOUT data set currently resides. The local node name appears in the JES job log of every job.

**Note:** It is recommended that you define a profile in the RACFVARS class named &RACLNDE, and use &RACLNDE for all profiles in the JESSPOOL class.

### ***userid***

is the user ID associated with the job. This is the user ID RACF uses for validation purposes when the job runs.

### ***jobname***

is the name that appears in the name field of the JOB statement.

**jobid**

is the job ID assigned to the job by JES. The job ID appears in notification messages and the JES job log of every job.

**dsidentifier**

is the unique data set identifier that JES assigned to the spool data set. This identifier is 8 bytes of alphanumeric characters. It is an encoded printable representation of the internal data set number (data set key) of the SPOOL data set. The internal data set number and data set name (which includes the *dsidentifier*) are available from the Extended Status SSI (SSI 80).

**Note:** The first 10 million data sets created by a job can be sorted chronologically on data set name. The same is true for data sets created after the first 10 million data sets. However, when the two subsets are sorted together, the resulting sequence is not in the order of data set creation.

**name**

is the name of the data set specified in the DSN= parameter of the DD statement. This name cannot be JESYSMSG, JESJCLIN, JESJCL, or JESMSG LG and follows the naming conventions for a temporary data set. For the temporary data set naming conventions, see [z/OS MVS JCL Reference](#). If the JCL did not specify DSN= on the DD statement that creates the spool data set, JES uses a single question mark (?).

**Note:** You can specify generic characters for any of the qualifiers in the profile name. For example, you can substitute an asterisk (\*) for one of the qualifiers, such as *jobid*, if it is not known.

A sample JESSPOOL profile name could be as follows. If user MYUSER submits a job named MYJOB to run on NODEA, and JES assigns a job ID of JOB08237, and the value of DSN= for a SYSOUT data set is OUTPUT, the profile name for a SYSOUT data set created by this job could be:

```
NODEA.MYUSER.MYJOB.JOB08237.D0000112.OUTPUT
```

If job MYJOB is run several times, and the same protection is desired for the OUTPUT data set each time, the profile name could be:

```
NODEA.MYUSER.MYJOB.*.*.OUTPUT
```

## 2. Give users the appropriate access authority, as follows:

```
PERMIT profile-name CLASS(JESSPOOL)
      ID(userid/groupname)
      ACCESS(access-authority)
```

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

**NONE**

Gives the user *no* access.

**READ**

Lets the user view the spool data set, but does *not* let the user change the data set's contents or attributes. For example, READ does *not* allow the following operands on the TSO OUTPUT command: DELETE, DEST, NEWCLASS, NOHOLD, and NOKEEP.

**UPDATE**

Lets the user read or update the contents of a spool data set. UPDATE does not allow the user to change the data set's attributes. UPDATE also allows users to update spool data sets opened by an application in the same address space.

**CONTROL**

Is equivalent to UPDATE.

**ALTER**

Lets the user read or update a spool data set or change the attribute of a spool data set. For example, ALTER allows any operand to be specified on the TSO OUTPUT command, including operands for deleting and printing. Also, when specified for a discrete profile, ALTER lets the user change the profile itself.<sup>14</sup>

**Note:** If SDSF is installed on your system, JESSPOOL profiles control which action characters and overtypable fields users can enter on SDSF panels. For complete information on creating JESSPOOL profiles for use with SDSF, see [z/OS SDSF Operation and Customization](#).

## Letting users create their own JESSPOOL profiles

Users can create their own JESSPOOL profiles if they have CLAUTH authority to the JESSPOOL class. If your installation decides to put the SETROPTS GENERICOWNER option into effect, you can restrict each user to creating JESSPOOL profiles only for his or her own spool data.

To do this, perform the following steps:

1. Issue this command:

```
SETROPTS GENERICOWNER
```

2. To prevent all users except the system administrator from being able to create JESSPOOL profiles, issue either of the following commands:

```
RDEFINE JESSPOOL ** OWNER(sys_admin_id) UACC(NONE)
RDEFINE JESSPOOL *  OWNER(sys_admin_id) UACC(NONE)
```

3. For each user who should be able to create JESSPOOL profiles for his or her own spool data, create a JESSPOOL profile with the user's user ID specified. Make the user the owner of the profile. For example, for users SMITH and BEN:

```
RDEFINE JESSPOOL nodename.SMITH.** OWNER(SMITH) UACC(NONE)
RDEFINE JESSPOOL nodename.BEN.**  OWNER(BEN)   UACC(NONE)
```

**Note:** These examples assume that a SETROPTS GENERIC(JESSPOOL) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

4. Give users CLAUTH authority to the JESSPOOL class:

```
ALTUSER SMITH CLAUTH(JESSPOOL)
ALTUSER BEN   CLAUTH(JESSPOOL)
```

5. Users with CLAUTH authority can define their own JESSPOOL profiles:

```
RDEFINE JESSPOOL profile-name OWNER(SMITH) UACC(NONE)
RDEFINE JESSPOOL profile-name OWNER(BEN)   UACC(NONE)
```

where *profile-name* is more specific than the JESSPOOL profile name you defined for this user in Step “3” on page 486.

6. After defining their own JESSPOOL profiles, the users with CLAUTH can use the following PERMIT command to grant other users access to the spool data sets protected by that profile:

```
PERMIT profile-name CLASS(JESSPOOL)
      ID(userid/groupname)
      ACCESS(access-authority)
```

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

### NONE

Gives the user *no* access.

### READ

Lets the user view the spool data set, but does *not* let the user change the data set's contents or attributes. For example, READ does *not* allow the following operands on the TSO OUTPUT command: DELETE, DEST, NEWCLASS, NOHOLD, and NOKEEP.

<sup>14</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

**UPDATE**

Lets the user read or update the contents of a spool data set. UPDATE does not allow the user to change the data set's attributes. UPDATE also allows users to update spool data sets opened by an application in the same address space.

**CONTROL**

Is equivalent to UPDATE.

**ALTER**

Lets the user read or update a spool data set or change the attribute of a spool data set. For example, ALTER allows any operand to be specified on the TSO OUTPUT command, including operands for deleting and printing. Also, when specified for a discrete profile, ALTER lets the user change the profile itself.<sup>15</sup>

**Note:** If SDSF is installed on your system, JESSPOOL profiles control which action characters and overtypable fields users can enter on SDSF panels. For complete information on creating JESSPOOL profiles for use with SDSF, see [z/OS SDSF Operation and Customization](#).

## Protecting JESNEWS

JESNEWS is a spool file that contains data to be printed following each job's output. Protecting JESNEWS prevents unauthorized users from adding, modifying, or deleting these files, or (if security labels are used) writing data with a higher security label into these files.

The procedure for protecting JESNEWS depends on whether JES2 or JES3 is installed.

### Protecting JESNEWS for JES2

To protect JESNEWS for JES2, perform the following steps:

1. Ask the JES2 system programmer for the following information:
  - The fully qualified name of each JESNEWS file to be protected
  - The universal access authority to be associated with each JESNEWS file. For JESNEWS, this value should always be READ to allow all JES users to receive JESNEWS.
  - The user IDs or group names of operators and users that are to be authorized to update JESNEWS. Assign each of these users or groups an access authority of UPDATE to the appropriate profile in the OPERCMDS class. Ensure that all users and operators are defined to RACF.
  - The security label to be associated with each JESNEWS file (if security labels are being used). For JESNEWS, this value should always be the lowest security label (SYSLOW) to allow JESNEWS to be printed for all users.
2. Create the following profiles:

```
RDEFINE JESSPOOL nodename.userid.$JESNEWS.STCtaskid.Dnewslvl.JESNEWS
UACC(READ)
```

where:

***nodename***

is the name of the node that created the JESNEWS data set.

***userid***

is the user ID associated with your JES2 system.

***STCtaskid***

is the name of the task that created the JESNEWS data set.

***Dnewslvl***

is the level of this copy of JESNEWS.

For example, for JESNEWS on NODEB:

<sup>15</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

```
RDEFINE JESSPOOL NODEB.*.$JESNEWS.*.*.JESNEWS UACC(READ)
```

**Note:**

- a. This example assumes that a SETROPTS GENERIC(JESSPOOL) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.
- b. To improve system performance, you should consider including an entry for JESNEWS in the global access checking table. For example:

```
NODEB.*.$JESNEWS.*.*.JESNEWS/READ
```

3. To prevent unauthorized updating of JESNEWS, define a profile in the OPERCMDS class. Any users authorized to update JESNEWS must have ALTER access to this resource:

```
RDEFINE OPERCMDS jesname.UPDATE.JESNEWS UACC(NONE)
PERMIT jesname.UPDATE.JESNEWS CLASS(OPERCMDS) ID(user or group) ACCESS(ALTER)
```

If RACF is not active, JES2 requests authorization to update JESNEWS from the operator.

**Note:** If RACF and the SECLABEL class are active, RACF assigns the SECLABEL of the last job that updated JESNEWS to the JESNEWS profile. This could cause jobs with lower security labels than the updating job to miss important information and RACF records security violations for jobs accessing JESNEWS that did not previously occur. To make JESNEWS accessible to all users, the job that creates it should have a SECLABEL of SYSLOW and the data set profile should have a UACC of READ. If the SECLABEL is greater than SYSLOW, JESNEWS does not print in the output of any jobs submitted with a lower SECLABEL.

## Protecting trace data sets (JES2 only)

For JES2, trace data sets contain information that could compromise your installation's security (for example, user IDs and passwords). You can protect these data sets by defining profiles in the JESSPOOL class and permitting only those users that need access to the data sets.

For example:

```
RDEFINE JESSPOOL NODE1.JES.*.*.*.JESTRACE UACC(NONE)
PERMIT NODE1.JES.*.*.*.JESTRACE CLASS(JESSPOOL) ID(SMITH) ACCESS(ALTER)
```

**Note:** This example assumes that a SETROPTS GENERIC(JESSPOOL) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

## Protecting SYSLOG

Your security policy might require that you protect SYSLOG because it is the record of your system's daily activities.

To control SYSLOG, define a JESSPOOL profile for the data set, specifying an appropriate universal access, and then grant access to the user IDs or group names that need a different access.

For example:

```
RDEFINE JESSPOOL system-name.+MASTER+.SYSLOG.*.* UACC(NONE)
PERMIT system-name.+MASTER+.SYSLOG.*.* CLASS(JESSPOOL) ID(SMITH) ACCESS(ALTER)
```

**Note:** This example assumes that a SETROPTS GENERIC(*classname*) was previously issued to turn generics on for this class and that a SETROPTS REFRESH was then done.

## Spool offload considerations (JES2 only)

You should protect offloaded information by defining the offload data set to RACF a universal access authority of NONE. If you have security labels active, you should assign the offload data a SECLABEL of SYSHIGH to prevent unauthorized access.



## Offloading data

When you offload data from the spool to another device, JES2 copies the security information for the data to the offload job and data set headers. No validation is made of the security information written to the offload data set. JES2 calls RACF to ensure the operator starting the offload operation has sufficient authority to issue the command to start the offload.

During the offload process, JES2 calls RACF (using the WRITER class) to ensure the owner of the SYSOUT data set has at least READ access to the offload device by checking the security information associated with the data against the device's profile in the WRITER class. The offload device profile for offload SYSOUT transmitter 1 would be:

```
jesxLOCAL.OFF1.ST
```

During spool offload, jobs are not checked for access to the device.

## Reloading data

When JES2 reloads the information from an offload data set, it performs any security validation necessary (similar to reading a job into the system or receiving a network SYSOUT data set) before writing the data to spool by checking the JESJOBS class for reloaded jobs and the NODES class. When RACF performs the NODES class check, if the node associated with the data is in the &RACLNDE profile, RACF accepts the data.

The following profiles for the JESINPUT class apply to spool reload:

```
OFFn.JR      for jobs
OFFn.SR      for SYSOUT
```

As with offload, JES2 calls RACF to ensure the operator starting the reload operation has sufficient authority to issue the commands.

When reloading a data set that was offloaded on this node, the name of the node must be defined in the RACFVARS profile &RACLNDE, or NODES profiles are required for NJE processing to associate user IDs with jobs or data.

## Dumping jobs

JES3 dumps all security information associated with each job when you use the dump job facility. However, JES3 does not perform security validation while dumping jobs.

## Restoring jobs

JES3 calls RACF to revalidate the job. RACF validates the job using the security information saved when the job was dumped and writes an SMF audit record for each restored job.

**Important:** Jobs and data transported to a complex that uses different security labels might be inadvertently declassified.

## Controlling access to key labels for spool data sets

You can use RACF to control key label processing for SPOOL data sets, which then controls what instream or SYSOUT data sets created by jobs are encrypted and compressed. The following topics will describe the RACF profiles that are used and what capabilities they provide.

### JESJOBS ENCRYPT profiles

Earlier topics in this chapter identify the JESJOBS class as a mechanism that restricts access to or the ability to affect/modify certain objects using criteria such as job names. SPOOL data set encryption has expanded on that function of the JESJOBS class. The creation of an ENCRYPT profile in the JESJOBS class

now provides the ability to specify a default key label that can be used by the system to encrypt and compress instream and SYSOUT data sets.

The format of the ENCRYPT profile name is:

```
ENCRYPT.nodename.userid.jobname.dsname
```

where:

**nodename**

The NJE node name where this SPOOL data set is being created.

**userid**

Owning user ID of the job, if the owner of the job is known when the SPOOL data set is created. Otherwise, the submitting user ID is used.

**jobname**

The name associated with the job.

**dsname**

The one to eight character data set name specified on a JCL DD statement via the DSNAME or DSN keyword.

The creation of the profile name allows the user to define a default key label for whatever level of specification that fits the user's security scheme. If the user wishes to define a default for all new instream or SYSOUT SPOOL data sets they could create a default key label on the profile:

```
ENCRYPT.*
```

Refer to [“Creating the JES segment in JESJOBS ENCRYPT profile” on page 490](#) for more information on how to create a default key label within a JES segment on a JESJOBS ENCRYPT profile.

## Creating the JES segment in JESJOBS ENCRYPT profile

You can create a JES segment within an ENCRYPT profile. The JES segment contains an installation-defined default key label that is used as a default key label for encryption operations. You can specify the following information in a JES segment:

**KEYLABEL**

The public name of a protected encryption key in the ICSF key repository. The key label is used by the system to encrypt a SPOOL data set.

To define or change information in the JES segment of an ENCRYPT profile, you must have either the SPECIAL attribute, or authority to the appropriate FIELD class profile. See [“Field-level access checking” on page 200](#) for more information.

**Example:**

```
RDEFINE JESJOBS ENCRYPT.NODE1.USER1.PRJOB*.* JES(KEYLABEL(NODE1.SPOOL.ENCRYPT))
```

## Specifying key labels in JCL

The user can override any defined default key label by supplying a key label in the JCL of a DD JCL statement. The DSKEYLBL keyword on the DD statement is used to supply a key label that overrides the default. See [DSKEYLBL parameter](#) in z/OS MVS JCL Reference for more information on DSKEYLBL.

To override the default key label value, the user must have access to a FACILITY class RACF profile. The profiles associated with overriding default key labels are:

**JES.ENCRYPT.SUBMITTER**

This profile is used when the owner of the job is not yet known. Generally, this profile is checked for SYSIN data sets during input processing. READ access to this profile of the submitting user ID allows overriding the default key label with a key label supplied in the DSKEYLBL keyword of the DD JCL statement.

**JES.ENCRYPT.OWNER**

This profile is used for jobs submitted through other means such as the card reader. The owning user's READ access to this profile will allow overriding the default key label with a key label supplied in the DSKEYLBL keyword of the DD JCL statement.

This profile is used when the owner of the job is known. Generally, this profile is checked for SYSOUT data sets when a job enters execution. READ access to this profile of the owning user ID allows overriding the default key label with a key label that is supplied in the DSKEYLBL keyword of the DD JCL statement.

## Authorizing console access

---

This topic discusses protecting MCS consoles, remote workstations, and JES3 consoles.

### MCS consoles

Your MVS system programmer can require operators to log on to and log off from MCS-managed consoles by specifying options in the CONSOLxx member of the SYS1.PARMLIB data set. When the CONSOLE class is active and a console being used is protected by a profile in the CONSOLE class, RACF ensures that the person attempting to LOGON has the proper authority to do so.

For information about controlling access to MCS consoles, see [“Protecting consoles” on page 228](#).

### Remote workstations (RJP/RJE consoles)

Your JES system programmer can require that remote workstation operators enter a password during workstation logon. This can be done through RACF or by using JES initialization statement parameters.

**Note:** In JES2, remote workstations are called RJE consoles. In JES3, they are called RJP consoles. If the workstation is connected using BSC, the operator must issue a /\*SIGNON statement. If the workstation is connected using SNA, the operator must issue a LOGON statement.

If you want RACF to check LOGON or /\*SIGNON passwords, you must activate the FACILITY class and define a profile for each workstation in both the FACILITY and USER classes. You should also ask your JES system programmer for the workstation name. If JES2 is installed, the workstation name has the form RMTnnnn, where nnnn is the remote workstation number. If JES3 is installed, the workstation name is derived from the RJPWS initialization statement for an SNA workstation or the RJPTERM initialization statement for BSC. This workstation name serves as the user ID for the workstation console. Users of the RJP console have to log on using this terminal ID and supply the same password.

You might also need similar support for NJE nodes for command and user ID authorization from the network. NJE nodes do not sign on as RJE workstations do, but rather perform the FACILITY/USERID verification as each command is issued. Also see [“Authorizing the use of operator commands” on page 495](#).

Command validation in JES is composed of two parts:

1. Validating that the originator of the command can issue the command.
2. Validating that the originator is authorized to the object of the command.

RACF control is only applied to the issuance of the command. JES continues to validate what object a particular workstation or node can affect.

**Note:**

1. JES password protection or command authorization is used instead of RACF protection if any of the following conditions exist:
  - RACF is not installed.
  - No NJE node or remote workstation profile exists in the FACILITY class.
  - RACF is active, but the FACILITY class is not active.

2. If RACF is installed but not active, control returns to JES, and JES does its own password checking or command authorization.
3. Workstation operators can change their user passwords only at logon time.
4. RACF password protection replaces JES password protection for remote workstations. That is, either RACF or JES, but not both, verifies logons and passwords. Similarly, RACF command authorization across the network replaces JES NJE command authorization. That is, RACF or JES, but not both, verifies these commands.
5. The password for an RJE workstation must be changed the first time the workstation issues a LOGON or SIGNON.
6. Because the remote workstation or node name is also used as a port of entry, it needs to be defined to the JESINPUT class (if active). If it is not defined and the class is later activated, RJE signons or NJE command authorizations fail because of incorrect port of entry. For more information, see *MVS/ESA and RACF 1.9 Security Implementation Guide*.

To use RACF to check LOGON or /\*SIGNON passwords, perform the following steps:

1. For each remote workstation or node to be protected, ask your JES system programmer for the following:
  - The ID of the remote workstation. The ID serves as the user ID of the remote workstation. All users using a particular remote workstation must log on using this ID and supply the same password. (The password will never expire.) The ID is one of the following:
    - If JES2 is installed, the remote ID of the RJE console to be protected, which takes the form RMTnnnn.
    - If JES3 is installed, the ID of the console you want to protect.
    - For NJE nodes, the name of the node to be used as the user ID of that node.
2. For each remote workstation or NJE node, create a user profile:

```
ADDUSER userid
      DATA('data')
      PASSWORD(initial-password)
      DFLTGRP(groupname)
```

where:

**userid**

is the RJE remote ID or NJE node name.

**data**

is installation-defined, for example:

```
DATA('RJE console at xxx, phone yyy')
```

**initial-password**

is the initial password (to be changed immediately to another password that will never expire).

**groupname**

is a group that you allow to use certain RACF-protected resources, such as commands.

Specify that the passwords for these profiles will never expire:

```
PASSWORD USER(userid) NOINTERVAL
```

3. For each workstation for which you want RACF to check the user's password, create a profile in the FACILITY class, as follows:

```
RDEFINE FACILITY RJE.workstation
```

where *workstation* has been supplied by the JES system programmer.

**Note:** The existence of a profile in the FACILITY class for a remote workstation forces the user to enter a password to be checked by RACF, rather than by JES. The specification of UACC for these profiles has no effect.

4. For each NJE node for which you want RACF to check the user's command authorization, create a profile in the FACILITY class, as follows:

```
RDEFINE FACILITY NJE.nodename
```

where *nodename* has been supplied by the JES system programmer. The specification of UACC for these profiles has no effect.

5. Run a batch job with old and new passwords specified to set a new password (which will never expire).
6. When you are ready to start using the protection provided by the profiles you have created, activate the FACILITY class:

```
SETROPTS CLASSACT(FACILITY)
```

7. If the class is active, define the workstation or node name to the JESINPUT class, as follows:

```
RDEFINE JESINPUT workstation UACC(appropriate-access)
RDEFINE JESINPUT nodename UACC(appropriate-access)
```

If the workstation or node name is not defined and the class is later activated, sign on or command authorization fails because of incorrect port of entry. For more information, see *MVS/ESA and RACF 1.9 Security Implementation Guide*.

## Controlling where output can be processed

You can use the WRITER class to control where output can be printed. For example, you can authorize or restrict the use of writers for local printers and punches, remote workstations (RJE and RJP devices), and network nodes. You can also limit which classification of data can be sent to a particular device or node. For information about how to use the WRITER class to control outbound jobs and SYSOUT for NJE, see [“Authorizing outbound work” on page 483](#).

When the WRITER class is active, RACF ensures that the user is authorized to use a writer. For network devices, RACF also verifies the security of outbound data sets to ensure that the originator is authorized to send the data set to another node in a network.

To control where output can be sent, do the following:

1. Ask your JES system programmer for the following information:
  - The name of your JES system
  - If you are protecting local printers, local punches, or RJE devices, their device names
  - If you are protecting network devices, the name of the node that will ultimately receive the output

**Note:** The node name as specified in the JES initialization stream.

  - The security label if you want to limit which classifications of output can be sent to a particular output destination
  - The list of users to be authorized or restricted from using a specific output destination
2. Create a profile in the WRITER class to protect each writer:

```
RDEFINE WRITER profile-name UACC(appropriate-access)
```

where *profile-name* has one of the following formats:

- For local printers and punches:

```
jesname.LOCAL.devicename
```

- For JES2 RJE devices:

```
jesname.RJE.devicename
```

- For JES3 RJP devices:

```
jesname.RJP.devicename
```

- For data whose destination is a node:

```
jesname.NJE.nodename
```

where *nodename* is the name of the node to ultimately receive the output.

Also, UACC can be one of the following:

**NONE**

Allows no access

**READ**

Allows all users to send output to the protected device or node.

3. Give the appropriate access to users and groups:

```
PERMIT profile-name CLASS(WRITER) ID(user or group)
ACCESS(appropriate-access)
```

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

**NONE**

Allows no access

**READ**

Allows the user or group to send output to the protected device or node.

4. When you are ready to start controlling access to writers based on the profiles you have defined, activate the WRITER class:

```
SETOPTS CLASSACT(WRITER)
```

**Note:** If SDSF is installed on your system, WRITER profiles control which operations related to printers (such as displaying information about a printer or purging output) users can enter on SDSF panels. For complete information on creating WRITER profiles for use with SDSF, see [z/OS SDSF Operation and Customization](#).

## Authorizing the use of your installation's printers

You can use RACF to control who can use your installation's printers. Printers at your installation are defined to JES3 by DEVICE statements in the JES3 initialization stream. Printers are also defined in the Hardware Configuration Definition (HCD) program.

To authorize or restrict the use of your installation's printers, perform the following steps:

1. Ask your JES system programmer for the following information:

- A 4-part profile name that represents the printer. The format of the 4-part profile name is:

```
sysname.dev-class.modelno.ddd
```

where:

**sysname**

identifies the name of the system.

**dev-class**

specifies the type of device. For printers, you must always specify unit record (UR).

**modelno**

specifies the model number of the printer.

**ddd**

specifies the device number associated with the printer.

- The universal access authority associated with the printer. A UACC of READ indicates the printer can be allocated to all users in your installation. A UACC of NONE indicates the printer can only be allocated to the users you specify.
- A list of users and groups that have access other than the UACC. READ access allows the device to be allocated to the job submitted by the specified user.
- The security label associated with the printer (if security labels are being used).

2. Create a profile in the DEVICES class to protect each writer:

```
RDEFINE DEVICES profile-name UACC(NONE)
```

3. When you are ready to start using the protection provided by the profiles you have created, activate the DEVICES classes:

```
SETROPTS CLASSACT(DEVICES)
```

## Authorizing the use of operator commands

You can control which commands operators can enter at consoles. For more information, see [“Administering the use of operator commands” on page 239](#).

### Commands from RJE work stations

To control the commands entering from RJE workstations, do the following:

1. Add a user profile for the workstation. The user ID should be the name of the remote, with parentheses removed. For example, for RMT(1), the user ID is RMT1. If you are using RACF to sign on RJE workstations, see [“Remote workstations \(RJP/RJE consoles\)” on page 491](#). Here is a sample command:

```
ADDUSER RMT1
  DATA('RJE workstation at xxx, phone yyy')
  PASSWORD(initial-password)
  DFLTGRP(groupname)
```

2. Permit the RJE user ID to the appropriate command profiles:

```
PERMIT command-profile-name CLASS(OPERCMDS) ID(RMT1)
  ACCESS(appropriate-access)
```

3. If the OPERCMDs class is not already active, activate it:

```
SETROPTS CLASSACT(OPERCMDS)
```

### Commands from NJE nodes

To control the commands entering from NJE nodes, do the following:

1. Take the steps to define a user profile and FACILITY class profile for each node. FACILITY/USERID for NJE commands are verified as each command comes through the network. No advance sign on exists as with RJE workstations. See [“Remote workstations \(RJP/RJE consoles\)” on page 491](#). For example, for a node named HYDEPARK:

```
ADDUSER HYDEPARK
  DATA('NJE node at xxx, phone yyy')
  PASSWORD(initial-password)
  DFLTGRP(groupname)

RDEFINE FACILITY NJE.HYDEPARK
```

2. If the NODES class is active, create a NODES profile with RUSER as the second qualifier:

```
RDEFINE NODES nodename.RUSER.userid UACC(appropriate-access)
```

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

**NONE**

Reject the command

**READ**

Reverify

**UPDATE or higher**

Pass

3. Permit the node's user ID to the command profiles the node can issue:

```
PERMIT command-profile-name CLASS(OPERCMD) ID(HYDEPARK)  
ACCESS(appropriate-access)
```

4. If the OPERCMDS and FACILITY classes are not already active, activate them:

```
SETROPTS CLASSACT(OPERCMDS FACILITY)
```

## Who authorizes commands when RACF is active

If you are using MCS-managed consoles and enable RACF command authority checking, RACF performs command authorization.



---

## Chapter 19. RACF and Storage Management Subsystem (SMS)

This topic describes using RACF with the DFSMSdfp facility Storage Management Subsystem (SMS). It describes factors that administrators should consider when using RACF with SMS. A number of scenarios are used to explain these factors. Many of the names used in these scenarios are arbitrary. The names you use for user IDs, group names, and resources will differ, and the procedures you decide to follow might vary from those given as examples in this topic.

---

### Overview of RACF and SMS

You can use RACF to protect and control the use of SMS classes, data sets, functions, options, and commands. RACF provides the following facilities to support DFP:

- Supplied general resource classes that you can use to protect SMS classes (general resource classes are *not* the same as SMS classes)
- A DFP segment in both user and group profiles in which you can specify default information that DFP uses to determine data management and storage characteristics for data sets
- A DFP segment in data set profiles in which you can specify the owner of SMS-managed data sets protected by the profile
- Field-level access checking to provide security for fields in the DFP segment of user, group, and data set profiles

The following sections describe the details of these RACF facilities.

---

### RACF general resource classes for protecting SMS classes

RACF provides the following general resource classes for protecting SMS management classes and SMS storage classes. (SMS data classes do not require RACF protection.)

- **MGMTCLAS.** Use this RACF resource class to protect specific SMS management classes. (Management class is the DFP construct name for a collection of attributes related to the migration and backup of data sets.)
- **STORCLAS.** Use this RACF resource class to protect specific SMS storage classes. (Storage class is the DFP construct name for attributes related to space for a data set and the device and volume on which a data set resides.)

**Note:** The RACF general resource classes MGMTCLAS and STORCLAS are different from, and should not be confused with, the DFP construct names management class and storage class.

---

### Controlling the use of SMS classes

To control the use of SMS classes, issue the following RACF commands as described.

First, issue the SETROPTS command with the CLASSACT operand to activate the RACF general resource classes MGMTCLAS and STORCLAS. The format of the command is as follows:

```
SETROPTS CLASSACT(MGMTCLAS STORCLAS)
```

Then, to define a specific SMS class, issue the RDEFINE command and specify the appropriate operands. After you define a profile to protect a specific SMS class, issue the PERMIT command to create entries in the access list of the profile. You might want to look at [“Determining the owner of an SMS-managed data set” on page 501](#) for more information.

For example, suppose you want to define a profile in the RACF general resource class STORCLAS to protect an SMS storage class named DFP2STOR. You can control which users and groups can use DFP2STOR by issuing one of the following sequences of commands:

- To limit the number of users who can use DFP2STOR:
  1. Issue the RDEFINE command to define the profile for DFP2STOR and assign a UACC of NONE to the profile. The format of the command is as follows:

```
RDEFINE STORCLAS DFP2STOR UACC(NONE)
```

This command specifies that no users can access DFP2STOR, except for the creator of the profile. For more information, see [z/OS Security Server RACF Command Language Reference](#).

2. Selectively allow certain users and groups access to DFP2STOR by issuing the PERMIT command and specifying an ACCESS of READ. The format of the command is as follows:

```
PERMIT DFP2STOR CLASS(STORCLAS) ID(SMITH JONES) ACCESS(READ)
```

This command allows SMITH and JONES the use of storage class DFP2STOR.

- To allow many users the use of DFP2STOR:
  1. Issue the RDEFINE command to define the profile for DFP2STOR and assign a UACC of READ to the profile. The format of the command is as follows:

```
RDEFINE STORCLAS DFP2STOR UACC(READ)
```

This command specifies that all users can access DFP2STOR.

2. You can selectively exclude certain users and groups from using DFP2STOR by issuing the PERMIT command and specifying an ACCESS of NONE. The format of the command is as follows:

```
PERMIT DFP2STOR CLASS(STORCLAS) ID(SMITH JONES) ACCESS(NONE)
```

This command prevents SMITH and JONES from using storage class DFP2STOR.

- For SMS resource classes that you want to be available to all users, consider creating an entry in the global access checking table. For example, to allow all users access to DFP2STOR, enter:

```
RDEFINE GLOBAL STORCLAS ADDMEM(DFP2STOR/READ)
SETROPTS GLOBAL(STORCLAS) REFRESH
```

Global access checking helps reduce processing overhead associated with RACF authorization checking. For SMS resources that you want to have available to a limited number of users, consider using SETROPTS RACLIST processing for STORCLAS and MGMTCLAS to provide the best performance.

After you define profiles in the MGMTCLAS and STORCLAS resource classes, you should activate SETROPTS RACLIST processing for these classes. This can improve performance by reducing I/O to the RACF database.

To activate SETROPTS RACLIST processing for the MGMTCLAS and STORCLAS resource classes, issue the SETROPTS command with the RACLIST operand and specify the appropriate RACF resource class names. The format of the command is as follows:

```
SETROPTS RACLIST(STORCLAS MGMTCLAS)
```

For more information, see [“SETROPTS RACLIST processing” on page 127](#).

## Refreshing profiles for SETROPTS RACLIST processing for MGMTCLAS and STORCLAS

If SETROPTS RACLIST processing has been activated for the MGMTCLAS and STORCLAS resource classes, you must refresh profiles for RACLIST processing for either class when you do one of the following:

- Define a new profile in the class
- Make changes to existing profiles in the class

Refreshing profiles for SETROPTS RACLIST processing for a RACF resource class ensures that the most current copy of a profile resides in storage and is available for RACF authorization checking.

To refresh profiles for SETROPTS RACLIST processing for the MGMTCLAS or STORCLAS resource classes, issue the SETROPTS command with the RACLIST and REFRESH operands and specify the appropriate RACF resource class names. The following command refreshes profiles for SETROPTS RACLIST processing for both MGMTCLAS and STORCLAS:

```
SETROPTS RACLIST(STORCLAS MGMTCLAS) REFRESH
```

For more information, see [“Refreshing profiles for SETROPTS RACLIST processing”](#) on page 129.

## DFP segment in RACF profiles

To support DFP, RACF provides a DFP segment in user, group, and data set profiles. The following sections describe the information you can specify in this segment, how RACF and DFP use this information, and how you can use field-level access checking to control access to the DFP segment.

### DFP segment in user and group profiles

When SMS is installed and active on your system, every SMS-managed data set is assigned the following DFP constructs:

- Data class, which contains attributes related to the allocation of the data set
- Management class, which contains attributes related to the migration and backup of the data set
- Storage class, which contains attributes related to space for the data set and the device and volume on which the data set resides.

RACF provides the DFP segment in user and group profiles in which you can specify default values for these constructs as well as a data application identifier. During allocation of a new SMS-managed data set, RACF retrieves these default values for DFP. DFP, in turn, uses these values as input to the automatic class selection (ACS) routines that are used by SMS to assign constructs to the new data set.

The fields contained in the DFP segment of user and group profiles are as follows:

- DATAAPPL, which specifies the identifier for the data set application
- DATACLAS, which specifies the default data class
- MGMTCLAS, which specifies the default management class
- STORCLAS, which specifies the default storage class

For user and group profiles, you can specify information in the DFP segment using one of the following commands:

- ADDUSER, when defining a new user profile
- ALTUSER, when changing a user profile
- ADDGROUP, when defining a new group profile
- ALTGROUP, when changing a group profile

When defining or changing values in the DFP segment of user or group profiles, you should consider the following:

- The values that you specify for MGMTCLAS and STORCLAS must be defined as profiles in their respective RACF general resource classes and the user or group must be granted at least READ access. Otherwise, RACF does not allow the user or group to use the specified SMS class. For more information, see [“Controlling the use of SMS classes”](#) on page 497.

- RACF does not control access for DATAAPPL or DATACLAS. However, the values you specify in these fields should be defined for use on your system.
- Your storage administrator defines the names for the DFP constructs data class, management class, and storage class. To determine what construct names have been defined on your system, you can display a list of these names by using the Interactive Storage Management Facility (ISMF). For information on how to use ISMF, see *z/OS DFSMS Using the Interactive Storage Management Facility*.

You can display the information in the DFP segment of a user profile by issuing the LISTUSER command with the DFP operand and, for a group profile, by issuing the LISTGRP command with the DFP operand. For more information on the RACF commands, see *z/OS Security Server RACF Command Language Reference*.

**Note:** If you want to display the information in the DFP segment of any RACF profile, you must have the SPECIAL, AUDITOR, or ROAUDIT attribute, or at least READ access to the segment through field-level access checking. For information on field-level access checking for the DFP segment, see *“Controlling access to the DFP segment”* on page 501.

## Choosing different default values for DFP constructs

In most cases, the default values for constructs specified in the DFP segment of a user or group profile are sufficient for managing new data sets. When defining a new SMS-managed data set, however, a user can choose different default values for any of the following fields by using JCL or dynamic allocation:

- DATACLAS
- MGMTCLAS
- STORCLAS

For more information, see *z/OS MVS JCL User's Guide*.

## DFP segment in data set profiles

In data set profiles, the DFP segment contains the RESOWNER field in which you can specify the owner (RACF-defined user or group) of an SMS-managed data set protected by the profile. When a user allocates a new SMS-managed data set protected by this profile, the user ID or group ID that you specify in the RESOWNER field must have at least READ access authority to the MGMTCLAS or STORCLAS profile used in the allocation. If RESOWNER is not specified, the user or group name matching the high-level qualifier is used. In most cases, the owner of an SMS-managed data set is the user ID or group name that matches the high-level qualifier of the data set name. RACF provides the RESOWNER field to give your installation the flexibility to select any RACF-defined user or group to be the data set owner.

You should specify a value for RESOWNER when the owner of a data set must be different from the high-level qualifier of the data set name. For example, assume that you have defined the groups PAYROLL and LEGAL on your system. Assume also that PAYROLL needs to create some data sets for LEGAL, but LEGAL requires ownership of the data sets. If you issue the following command, you create the data set profile PAYROLL.LGL88.\*\* with LEGAL as owner of any SMS-managed data sets protected by the profile:

```
ADDSD 'PAYROLL.LGL88.**' DFP(RESOWNER(LEGAL)) UACC(NONE)
```

The PAYROLL group can then create data sets such as PAYROLL.LGL88.WEEK1, PAYROLL.LGL88.WEEK2, and PAYROLL.LGL88.MARCH.SUM, but the LEGAL group actually owns the data sets.

(The profile name PAYROLL.LGL88.\*\* is a generic profile name that uses enhanced generic naming. Before you issue the ADDSD command, both generic profile checking for the DATASET class and enhanced generic naming must be active. If these options are not active, issue the SETROPTS GENERIC(DATASET) and SETROPTS EGN commands before you define the generic profile.)

You can specify a value for RESOWNER when you define a new data set profile using the ADDSD command or when you change an existing data set profile using the ALTDSD command. You can display

the information in this field using the LISTDSD command. See *z/OS Security Server RACF Command Language Reference* for more information on these commands.

Note that the RESOWNER field, which represents the data set owner for data set allocation purposes, is different from the OWNER field, which represents the user or group that owns the data set profile and can therefore work with the profile itself.

## How RACF uses the information in the DFP segments

When a user creates a new SMS-managed data set, RACF uses the information in the DFP segment of a data set profile together with the information in the DFP segment of a user or group profile as described in the following sections.

### Determining the owner of an SMS-managed data set

DFP invokes RACF during data set allocation to determine the owner of the data set.

If the data set is not protected by a profile, RACF returns the high-level qualifier of the data set name as the default value for the owner of the data set.

If there is a data set profile, RACF checks it to determine whether the DFP segment contains a value for the RESOWNER field. If the RESOWNER field contains a value (user ID or group name), RACF returns this value to DFP as the owner of the data set. If the RESOWNER field does not contain a value, RACF uses the high-level qualifier of the data set name as the default value for the owner of the data set. In either situation, the value that RACF returns can be used as an input variable to ACS routines.

### Retrieving default DFP information from user and group profiles

To have SMS use RACF for retrieving default values from the DFP segment of user and group profiles, your installation must specify ACSDEFAULTS=YES in the IGDSMSmm member of SYS1.PARMLIB. For more information, see *z/OS DFSMSdfp Storage Administration*.

After invoking RACF to determine the owner (user or group) of an SMS-managed data set, DFP again invokes RACF to retrieve the default values for DATAAPPL, DATACLAS, MGMTCLAS, and STORCLAS from the DFP segment of the owner's RACF profile. DFP uses these values as input to ACS routines during allocation of the data set.

### Authorization checking for protected SMS classes

During allocation of an SMS-managed data set, DFP performs a RACF authorization check to verify that the data set owner is allowed to use the specified MGMTCLAS and STORCLAS. If the data set owner does not have at least READ authority to both of these classes, RACF denies the request. As a result, DFP does not allocate the data set.

This authorization checking occurs regardless of the source of the DATACLAS, MGMTCLAS, or STORCLAS values. It does not matter whether they were supplied as RACF defaults, through the ACS routines, or by the user through JCL or dynamic allocation parameters.

**Note:** If the data set owner is a revoked user ID, the allocation fails.

## Controlling access to the DFP segment

You can use field-level access checking to control a user's ability to add, delete, modify, or access information in any field of the DFP segment of a user, group, or data set profile. To implement field-level access checking, issue RACF commands as described in the following sections.

### Activating the FIELD class

First, activate the FIELD general resource class (if it is not already active). To activate this class, issue the SETROPTS command with the CLASSACT operand as follows:

```
SETROPTS CLASSACT(FIELD)
```

## Defining profiles for field-level access checking

After you activate the FIELD class, you can define profiles that allow you to control access to fields in the DFP segment of user, group, or data set profiles. (Note that you can allow other users to define such profiles by assigning them the CLAUTH attribute for the FIELD class.) To define a profile for field-level access checking, issue the RDEFINE command and specify the class name as FIELD, the appropriate profile name, and the universal access authority (UACC) as follows:

```
RDEFINE FIELD profile-name UACC(access-authority)
```

When you specify *profile-name*, use the format shown in the following examples.

### Controlling access to all fields in the DFP segment of user profiles

You can define a profile that lets you control access to *all* of the fields in the DFP segment of all user profiles. Before you define this profile, generic profile checking for the FIELD class must be active. If generic profile checking is not active, issue the SETROPTS GENERIC(FIELD) command.

To define this profile, issue the RDEFINE command with a generic profile name. For example, enter:

```
RDEFINE FIELD USER.DFP.* UACC(NONE)
```

**Note:** When you specify a UACC of NONE, you prevent all users from accessing the DFP segment in all user profiles, including their own. Likewise, if you specify a UACC of READ, you allow all users to read the information contained in all fields of the DFP segment for all user profiles.

If the FIELD class is not yet RACLISTed, you must enter the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD)
```

If the FIELD class is already RACLISTed, you must refresh the profiles with the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD) REFRESH
```

### Controlling access to a specific field in the DFP segment of user profiles

You can define a profile that allows you to control access to a *specific* field in the DFP segment of all user profiles by issuing the RDEFINE command and specifying *profile-name* as shown in the following example:

```
RDEFINE FIELD USER.DFP.DATACLAS UACC(NONE)
```

This command defines a profile in the FIELD general resource class that protects, with a UACC of NONE, the DATACLAS field in the DFP segment of all user profiles. For more information, see [“Field-level access checking” on page 200](#).

If the FIELD class is not yet RACLISTed, you must enter the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD)
```

If the FIELD class is already RACLISTed, you must refresh the profiles with the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD) REFRESH
```

### Controlling access to all fields in the DFP segment of group profiles

You can define a profile that allows you to control access to *all* fields in the DFP segment of all group profiles. Before you define the following profile, generic profile checking for the FIELD class must be active. If it is not active, issue the SETROPTS GENERIC(FIELD) command before you define the generic

profile. To define this profile, issue the RDEFINE command and specify GROUP.DFP.\* for *profile-name* as shown in the following example:

```
RDEFINE FIELD GROUP.DFP.* UACC(NONE)
```

**Note:** When you specify a UACC of NONE, you prevent all users from accessing the DFP segment in all group profiles, including their current connect group. Likewise, if you specify a UACC of READ, you allow all users to read the information contained in all fields of the DFP segment for all group profiles.

If the FIELD class is not yet RACLISTed, you must enter the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD)
```

If the FIELD class is already RACLISTed, you must refresh the profiles with the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD) REFRESH
```

### ***Controlling access to a specific field in the DFP segment of group profiles***

You can define a profile that allows you to control access to a *specific* field in the DFP segment of all group profiles by issuing the RDEFINE command and specifying *profile-name* as shown in the following example:

```
RDEFINE FIELD GROUP.DFP.STORCLAS UACC(NONE)
```

This command defines a profile in the FIELD general resource class that protects, with a UACC of NONE, the STORCLAS field in the DFP segment of all group profiles. For more information, see [“Field-level access checking”](#) on page 200.

If the FIELD class is not yet RACLISTed, you must enter the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD)
```

If the FIELD class is already RACLISTed, you must refresh the profiles with the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD) REFRESH
```

### ***Controlling access to all fields in the DFP segment of data set profiles***

You can define a profile that allows you to control access to *all* fields in the DFP segment of all data set profiles. Before you define this profile, generic profile checking for the FIELD class must be active. If generic profile checking is not active, issue the SETROPTS GENERIC(FIELD) command. To define this profile, issue the RDEFINE command and specify DATASET.DFP.\* for *profile-name* as shown in the following example:

```
RDEFINE FIELD DATASET.DFP.* UACC(NONE)
```

**Note:** When you specify a UACC of NONE, you prevent all users from accessing the DFP segment in all data set profiles, including data set profiles that they own. Likewise, if you specify a UACC of READ, you allow all users to read the information contained in all fields of the DFP segment for all data set profiles.

If the FIELD class is not yet RACLISTed, you must enter the following command after you define or alter the profile:

```
SETROPTS RACLIST(FIELD)
```

If the FIELD class is already RACLISTed, you must refresh the profiles with the following command after you define or alter the profile:



```
SETRPTS RACLIST(FIELD) REFRESH
```

### Controlling access to a specific field in the DFP segment of data set profiles

You can define a profile that allows you to control access to a *specific* field in the DFP segment of all data set profiles by issuing the RDEFINE command and specifying *profile-name* as shown in the following example:

```
RDEFINE FIELD DATASET.DFP.RESOWNER UACC(NONE)
```

This command defines a profile in the FIELD general resource class that protects, with a UACC of NONE, the RESOWNER field in the DFP segment of all data set profiles. For more information, see [“Field-level access checking” on page 200](#).

If the FIELD class is not yet RACLISTed, you must enter the following command after you define or alter the profile:

```
SETRPTS RACLIST(FIELD)
```

If the FIELD class is already RACLISTed, you must refresh the profiles with the following command after you define or alter the profile:

```
SETRPTS RACLIST(FIELD) REFRESH
```

### Creating the access list for field-level access checking

After you define a profile to protect a resource in the FIELD class, you can create entries in the resource's access list using the PERMIT command. The following example shows how to create an entry that gives user DFPADMIN the authority to alter the SMS management class (MGMTCLAS field) in the profiles of all DFP users. Note that UPDATE authority is sufficient to change a value in a field of the DFP segment.

```
PERMIT USER.DFP.MGMTCLAS CLASS(FIELD) ID(DFPADMIN) ACCESS(UPDATE)
```

You can also specify the value &RACUID with the ID operand on the PERMIT command. When you enter this value on the PERMIT command, you allow all users access to the specified field within the DFP segment of their own user profiles. For example, if you issue the following command, you allow all users to read the DATAAPPL field in the DFP segment of their own user profiles.

```
PERMIT USER.DFP.DATAAPPL CLASS(FIELD) ID(&RACUID) ACCESS(READ)
```

## Controlling the use of other SMS resources

You can use RACF to control access to the following SMS resources:

- The Interactive Storage Management Facility (ISMF), including:
  - The entire ISMF component
  - Individual ISMF applications
  - ISMF functions, line operators, and commands
- The execution of functions, options, and commands, including:
  - Catalog functions for SMS data sets
  - DFSMSdss functions on data sets
  - Using SMS to activate a configuration
- SMS data sets, including:
  - Control data sets
  - Source data sets for ACS routines



- The test library for ACS routines

For information on how to use RACF to protect these resources, see the following documents:

- [\*z/OS DFSMS Using Data Sets\*](#), which shows the sequence of RACF commands you need to issue to protect the various SMS resources
- [\*z/OS DFSMSdfp Storage Administration\*](#) and [\*z/OS DFSMS Managing Catalogs\*](#), which show the names of the profiles you need to define to protect the various SMS resources.



---

## Chapter 20. RACF and TSO/E

This topic describes using RACF with TSO/E.

---

### TSO/E administration considerations

---

In order for users to log on to TSO, they must have an entry in the SYS1.UADS data set or a TSO segment defined in their RACF user profile. For more information, see [“The TSO segment in user profiles” on page 54](#).

**Note:** A TSO installation can write a TSO logon pre-prompt exit to bypass checking SYS1.UADS for user attribute information. For more information, see [z/OS TSO/E Customization](#).

You can move TSO user attribute information from SYS1.UADS to the RACF database. (SYS1.UADS contains an entry for each TSO user that describes the attributes that regulate the user's access to the system.) When you move this TSO information into the RACF database, it is stored in the TSO segment of the user's profile. When a user logs on to TSO, it uses the information contained in the TSO segment to build a session for the user.

Moving the TSO user information to the RACF database eliminates the need to maintain an entry in SYS1.UADS for each TSO user. However, you *must* maintain entries in SYS1.UADS for certain users (such as IBMUSER and system programmers).

For example, if you need to deactivate RACF to perform maintenance on the RACF database, users authorized to perform this maintenance must be able to log on to the system. When RACF is inactive, TSO checks entries in SYS1.UADS to authorize access to the system. When RACF is active, logon verification can produce an error during RACF processing. However, the logon can proceed by an alternative method (for example, UADS). This error occurs if the installation does not use the RACF database to store security-related information for a particular user, but it does use an alternative method (such as UADS) for the logon application to perform user verification.

**Note:** You can use the RACONVRT EXEC to help you convert SYS1.UADS entries to RACF user profiles. The RACONVRT EXEC creates a CLIST that contains multiple members. Each member contains RACF commands needed to add information read from the SYS1.UADS data set to the RACF database.

Be sure to inspect *all* members before running them. In particular, all of the ADDUSER commands that RACONVRT generates connect the users to group SYS1. *Be sure to modify the default groups before running this member.* You should also check the completeness and accuracy of the conversion that is performed by the RACONVRT EXEC. For more information on using the RACONVRT EXEC, see [z/OS TSO/E Customization](#).

---

### Protecting TSO resources

---

You can use RACF to protect certain TSO resources. These resources include TSO logon procedures, account numbers, and performance groups. In addition, you can protect resources called TSO user authorities, whose settings determine whether a user can issue certain authorized TSO commands. Examples of TSO user authorities include ACCT, JCL, MOUNT, OPER, RECOVER, PARMLIB, TESTAUTH, and CONSOLE. For detailed information about the TSO resources you can protect with RACF, see [z/OS TSO/E Customization](#).

If you are defining TSO segments in user profiles, you must protect these TSO resources, using the following general resource classes:

- TSOPROC (for protecting TSO logon procedures)
- ACCTNUM (for protecting TSO account numbers)
- PERFGPR (for protecting TSO performance groups)
- TSOAUTH (for protecting TSO user authorities)

The following access authorities apply to these resources:

#### NONE

No access allowed.

#### READ

For TSOPROC, ACCTNUM, and PERFGPR, allows users to specify the logon procedure, account number, or performance group when logging on.

For TSOAUTH, gives the user the authority to issue the associated authorized TSO command.

For PARMLIB, allows the user to issue the PARMLIB LIST command.

For TESTAUTH, allows the user to invoke a program in authorized state.

#### UPDATE

For PARMLIB, allows the user to issue the PARMLIB UPDATE command. For the other profiles, UPDATE is the same as READ.

#### CONTROL

Same as READ.

#### ALTER

Allows users to change the profile, if the profile is discrete.<sup>16</sup>

To control the use of TSO resources, issue RACF commands in the following sequence:

1. Activate the TSO general resource classes:

```
SETROPTS CLASSACT(TSOPROC ACCTNUM PERFGPR TSOAUTH)
```

**Considerations when activating the TSO resource classes:** Assume that you have defined a user profile for user SMITH that contains a TSO segment.

- If you do not activate the TSOPROC and ACCTNUM classes, user SMITH cannot log on to TSO because RACF cannot check SMITH's authority to use the logon procedure and account number specified on the logon panel. TSOPROC and ACCTNUM *must* be active so that users whose profiles contain TSO segments can log on to TSO.
  - If you do not activate the PERFGPR class and user SMITH specifies a performance group on the logon panel, SMITH cannot log on to TSO because RACF cannot check SMITH's authority to access the specified performance group. However, SMITH can log on to TSO when the performance group is deleted from the logon panel. Activate the PERFGPR class if your installation intends to use TSO performance groups.
  - If you do not activate the TSOAUTH class, user SMITH can log on to TSO but will not have any assigned TSO user authorities such as JCL or MOUNT. Activate the TSOAUTH class and give SMITH READ access authority to the appropriate resources in the TSOAUTH class if your installation is specifying user authorities when defining users to the system.
2. Create profiles to protect TSO resources. The following example shows how to define logon procedure LOGPROC1 to the TSOPROC resource class and assign it a UACC of READ. (A UACC of READ grants all users the ability to use the logon procedure.)

```
RDEFINE TSOPROC LOGPROC1 UACC(READ)
```

To protect a TSO resource so that a limited number of users can access it, you can define it and specify a UACC of NONE. Then you can create an access list containing only those users who require access to the resource. The following example shows how to define a logon procedure, LOGPROC2, in the TSOPROC resource class and protect it with a UACC of NONE.

```
RDEFINE TSOPROC LOGPROC2 UACC(NONE)
```

**Considerations for creating profiles for TSO resources:**

<sup>16</sup> More information about ALTER authority and how to limit it can be found in [Chapter 9, “Limiting ALTER access authority in discrete profiles,”](#) on page 259.

- For the TSOPROC class, the profile name must be the name of the logon procedure itself (no generic characters are allowed).
- For the ACCTNUM class, the profile name can be up to 39 characters long.

You should create at least one profile in the ACCTNUM class.

If you want a particular user to log on without an account number, you must ensure that the user has no access to any ACCTNUM profile. This means that you cannot specify UACC(READ) for *any* ACCTNUM profile. Also, a user can have access to an ACCTNUM profile by means of a connect group. If a user has access to one or more account numbers, the first such account number that RACF encounters when searching the RACF database becomes that user's default account number and is saved in the TSO segment of the user's profile. You can find out which account number is used by issuing the following command:

```
SEARCH CLASS(ACCTNUM) USER(userid)
```

The first account number listed is used. For example, you if you want to allow only two account numbers, D1001 and D1002, and you want to ensure that users log on with at least one of them, create the following profiles:

```
RDEFINE ACCTNUM D1001 UACC(READ)
RDEFINE ACCTNUM D1002 UACC(READ)
RDEFINE ACCTNUM ** UACC(NONE)
```

**Note:** Because of the order in which RACF searches the RACF database, account number D1001 is the default assigned to any user who logs on with a blank account number. To determine the search order in which profiles are used, issue SEARCH or RLIST command for the class. For example:

```
SEARCH CLASS(ACCTNUM)
```

- For the PERFGROUP class, the profile name must be the number of the performance group itself (no generic characters are allowed).
- For the TSOAUTH class, you should consider creating discrete profiles for each TSO attribute. The following examples assume that only a few users should be able to request mounts, but that every user (except those specifically disallowed) should be able to submit batch jobs:

```
RDEFINE TSOAUTH MOUNT UACC(NONE)
RDEFINE TSOAUTH JCL UACC(READ)
```

3. Use the PERMIT command to allow users and groups to use the TSO resources. The following example shows how to allow users USERA and USERB to specify logon procedure LOGPROC2 when they log on using TSO:

```
PERMIT LOGPROC2 CLASS(TSOPROC) ID(USERA USERB) ACCESS(READ)
```

4. Activate SETROPTS RACLIST processing for the TSO general resource classes:

```
SETROPTS RACLIST(TSOPROC ACCTNUM PERFGROUP TSOAUTH)
```

For more information on SETROPTS RACLIST processing, see [“SETROPTS options to activate in-storage profile processing”](#) on page 125.

**Note:** If SETROPTS RACLIST processing is already activated for the TSO general resource classes, you must refresh SETROPTS RACLIST processing:

```
SETROPTS RACLIST(TSOPROC ACCTNUM PERFGROUP TSOAUTH) REFRESH
```

For more information on refreshing SETROPTS RACLIST processing, see [“Refreshing profiles for SETROPTS RACLIST processing”](#) on page 129.

## Authorization checking for protected TSO resources

When a user logs on to TSO, TSO requests RACF authorization checking for protected TSO resources such as account numbers and logon procedures. For example, suppose that during logon, user SMITH requests the use of account number 12345. If SMITH is authorized to use account number 12345, RACF grants the request. If SMITH is not authorized to use account number 12345, the following occur:

- A message is sent to the operator console indicating that user SMITH has been denied access to a RACF-protected resource.
- An SMF record is generated indicating that RACF failed an attempt to access a protected resource (unless your installation has specified an alternative auditing option for account numbers).
- User SMITH is prompted to enter a valid account number.

RACF performs authorization checking in this manner for protected TSO resources in the TSOPROC, ACCTNUM, and PERFGPR classes. For resources in the TSOAUTH class, RACF performs authorization checking but no messages are sent to the operator console and no SMF records are generated.

## Field-level access checking for TSO

You can use RACF to control which users can access fields in the TSO segments of RACF profiles through *field-level access checking*. For more information on field-level access checking, see [“Field-level access checking”](#) on page 200.

## Controlling the use of the TSO SEND command

You can control whether users can receive messages sent with the TSO SEND command.

### Note:

1. When the SMESAGE class is active and a profile does not exist for the specified user, the SEND command completes normally.
2. When the SECLABEL class is active, the receiver of the message must pass the security label authorization check based on the receiver's current security label and the security label of the message (which was set by the sender's current security label at the time that the sender issued the TSO SEND command.)

To control the use of the TSO SEND command, do the following:

1. Create profiles in the SMESAGE class:

```
RDEFINE SMESAGE userid-of-receiver UACC(NONE)
```

2. Give users the appropriate access authority:

```
PERMIT RECV1 CLASS(SMESAGE) ID(SENDER1) ACCESS(READ)
PERMIT RECV2 CLASS(SMESAGE) ID(SENDER2) ACCESS(NONE)
```

where SENDER1 and SENDER2 are valid RACF user IDs or group names, and RECV1 and RECV2 are valid RACF user IDs. The *first* PERMIT in the example allows SENDER1 to send messages to RECV1. The *second* PERMIT in the example prevents SENDER2 from sending messages to RECV2.

3. When you are ready to start using the security provided by these profiles, activate both the DIRAUTH class and the SMESAGE class, and activate SETROPTS RACLIST processing for the SMESAGE class. SETROPTS RACLIST processing helps ensure high performance when access authorities are checked. You can do these actions in one command:

```
SETROPTS CLASSACT(SMESAGE DIRAUTH) RACLIST(SMESAGE)
```

**Note:** Any time you make a change to an SMESAGE profile, you must also refresh SETROPTS RACLIST processing for the SMESAGE class for the change to take effect. For example:

## Restricting spool access by TSO users

Activating the JESSPOOL class provides protection against unauthorized spool access through the use of the TSO OUTPUT and RECEIVE commands, as shown in [Table 35 on page 511](#).

Only authorized users (that is, the creator of a data set or those granted access through JESSPOOL profiles) have the authority to access SYSOUT data sets when using the TSO OUTPUT command.

To allow users to use the TSO OUTPUT command, create profiles in the JESSPOOL class and grant users access that they need. For more information, see [“Defining profiles for SYSIN and SYSOUT data sets” on page 484](#).

*Table 35. TSO command usage when RACF protection is enabled*

TSO command	RACF protection provided
OUTPUT	Prevents unauthorized users from reading data sets or changing the attributes of data sets on spool.
RECEIVE	Prevents unauthorized users from receiving transmitted data sets that are classified at a level higher than the receiver's current authority.

## TSO commands that relate to RACF

This topic summarizes the TSO commands that relate to RACF. For complete information on logging on to TSO and on TSO commands, see [z/OS TSO/E User's Guide](#) and [z/OS TSO/E System Programming Command Reference](#).

- Fields on the TSO/E logon panel
  - PASSWORD and NEW PASSWORD fields: Users should keep their passwords in confidence. Users should also know how to change passwords when logging on, and should know how to specify new passwords that comply with the syntax rules.

Users should be aware that the RACF might prevent them from changing their passwords too frequently if the SETROPTS PASSWORD(MINCHANGE) interval is set. Also, if mixed-case passwords are allowed, users must supply each character of their passwords in the same case as was used when the passwords were set.

  - GROUP IDENT field: Specify this field only if list-of-groups processing is *not* in effect and if the user wants the job to run with a group other than the user's default group.
  - SECLABEL field: Specify this field to log on with a security label other than the user's current security label.
  - Users with the OIDCARD attribute must supply a valid operator identification card during logon.
- Operands on the TSO LOGON command:
  - NEWPASSWORD (to specify a new password to replace the current password)
  - GROUP (to specify the group name to which the user is connected during the terminal session).
- Settings on the TSO PROFILE command:
  - PREFIX (used in determining the RRSFLIST data set name)
  - INTERCOM (determines whether RRSF uses SEND to send messages to the command issuer for directed commands).
- Sending and receiving messages: Depending on how security labels are used on your system, and on how SETROPTS options are set, users might need to be aware of their current security label when they send or receive messages from other users. For more information, see [“Controlling the use of the TSO SEND command” on page 510](#).

- Submitting and cancelling jobs: Users have flexibility with regard to which jobs they can work with using the TSO SUBMIT and CANCEL commands. For more information, see [“Surrogate job submission” on page 467](#) and [“Controlling who can cancel jobs by job name” on page 464](#).

## Using TSO when RACF is deactivated

---

While RACF is deactivated as a result of issuing the RVARY command, only users defined in the SYS1.UADS data set can still log on to TSO, but RACF does not do any processing. (For example, a user defined in SYS1.UADS who has the ADSP attribute can allocate DASD data sets, but RACF is not called to define discrete profiles for those data sets.)

If the TSO user has the WTPMSG option active, the user receives messages in the session that indicate the identity of the resource being defined or accessed.

Therefore, you should notify users who have the ADSP attribute when RACF is not active so that they are aware that profiles for newly created data sets are not being defined to RACF.

When RACF is reactivated, you should advise RACF users to log off TSO and log on again to ensure they are connected to their proper RACF group.



## Chapter 21. RACF and z/OS UNIX

This topic describes using RACF with z/OS UNIX. This topic describes factors to consider when using RACF to manage group identifiers (GIDs) and user identifiers (UIDs). It also describes how to map GIDs and UIDs to RACF group names and user IDs.

The z/OS UNIX security functions provided by RACF include user validation, file access checking, privileged user checking, and user limit checking. z/OS UNIX users are defined with RACF commands. When a job starts or a user logs on, the user ID and password are verified by RACF. When an address space requests an z/OS UNIX function for the first time, RACF:

1. Verifies that the user is defined as a z/OS UNIX user.
2. Verifies that the user's current connect group is defined as a z/OS UNIX group.
3. Initializes the control blocks needed for subsequent security checks.

### Additional reading:

- See *z/OS UNIX System Services Planning* for complete information on setting up and using RACF in the z/OS UNIX environment.
- See *z/OS Security Server RACF Auditor's Guide* for complete information on auditing in the RACF environment.
- See *z/OS Planning for Multilevel Security and the Common Criteria* for information about using security labels for z/OS UNIX files and directories.

**Note:** RACF program control does not control programs that are executed in any way that bypasses MVS contents supervision, such as load modules contained in z/OS UNIX files. Therefore, loading a program from a z/OS UNIX file prevents you from opening a data set in a PADS (program access to data sets) environment, and prevents you from loading a program from an MVS library if you only have EXECUTE authority. You should use program control to restrict access to any programs, such as these, that provide facilities for bypassing MVS contents supervision. For more information, see [Chapter 13, “Protecting programs,”](#) on page 301.

## Defining group identifiers (GIDs)

You can assign a group identifier (GID) to a RACF group by specifying a GID value in the OMVS segment of the RACF group profile. When a GID is assigned to a group, all users connected to this group as their default group who have a user identifier (UID) in their user profile can use z/OS UNIX functions and can access z/OS UNIX files based on the GID and UID values assigned. If a user's current connect group is not their default group, a GID must also be assigned to the current connect group.

The following command defines a GID for an existing RACF group:

### Example:

```
ALTGROUP SECADMIN OMVS(GID(336))
```

For more information about GIDs, see [“The OMVS segment in group profiles”](#) on page 87 and [“Defining UIDs and GIDs”](#) in *z/OS UNIX System Services Planning*.

Although the same GID can be assigned to multiple RACF groups, it is not recommended. If you assign the same GID to multiple groups, control at an individual group level is lost because the GID is used in z/OS UNIX security checks. RACF groups that have the same GID assignment are treated as a single group during z/OS UNIX security checks.

You can enforce identity uniqueness when assigning UNIX identifiers. For more on controlling GID uniqueness, refer to [“Controlling the use of shared UNIX identities”](#) on page 516. A unique GID can be defined automatically using the AUTOGID operand, as described in [“Enabling automatic assignment of unique UNIX identities”](#) on page 518.

For special considerations about using the RACF list-of-groups checking (GRPLIST) option for access to z/OS UNIX file system resources, such as z/OS UNIX files, see [“GRPLIST considerations for z/OS UNIX”](#) on page 112.

## Defining user identifiers (UIDs)

You can assign a user identifier (UID) to a RACF user by specifying a UID value in the OMVS segment of the RACF user profile. When assigning a UID to a user, make sure that the user's default group has an assigned GID. If the user specifies a group during logon or on a batch job, this current connect group must also have an assigned GID. A user with a UID and a default group (and current connect group, if applicable) with a GID can use z/OS UNIX functions and access z/OS UNIX files based on the assigned UID and GID values. If a UID and GID are not available as described, the user cannot use z/OS UNIX functions.

The following command defines a UID, and other OMVS segment information, for an existing RACF user:

### Example:

```
ALTUSER KAMAL OMVS(UID(122649) HOME('/') PROGRAM('/bin/sh'))
```

For more information about UIDs, see [“The OMVS segment in user profiles”](#) on page 52 and [“Defining UIDs and GIDs”](#) in *z/OS UNIX System Services Planning*.

Although you can assign the same UID to multiple users, it is not recommended. However, it might be necessary for some cases, such as superusers. If you assign the same UID to multiple users, control at an individual user level is lost because the UID is used in z/OS UNIX security checks. Users with the same UID assignment are treated as a single user during z/OS UNIX security checks.

You can enforce identity uniqueness when automatically assigning UNIX identifiers. For more on controlling UID uniqueness, refer to [“Controlling the use of shared UNIX identities”](#) on page 516. A unique UID can be defined using the AUTOUID operand, as described in [“Enabling automatic assignment of unique UNIX identities”](#) on page 518.

## Listing UIDs and GIDs

You can list the RACF users and groups associated with UIDs and GIDs using the following methods:

1. ISPF shell. See [z/OS UNIX System Services User's Guide](#) for information about using the ISPF shell.
2. RACF database unload utility (IRRDBU00). See [“Using the RACF database unload utility \(IRRDBU00\)”](#) on page 371 for information.
3. If you use UNIXMAP profiles to associate RACF users and groups with UIDs and GIDs, you can also use RLIST command. For example:

- To see the RACF groups that are associated with GID 237, enter:

```
RLIST UNIXMAP G237 ALL
```

- To see the RACF user IDs that are associated with UID 0, enter:

```
RLIST UNIXMAP U0 ALL
```

- To see all RACF groups and user IDs associated with GIDs and UIDs, enter:

```
RLIST UNIXMAP * ALL
```

For information about the UNIXMAP class, see [“Using the UNIXMAP class and Virtual Lookaside Facility \(VLF\)”](#) on page 524.

4. For installations at AIM stage 2 or higher, you can list a set of users or groups with a specific UID or GID, for example using '223' for the UID value and '55' for the GID value, enter:

```
SEARCH CLASS(USER)UID(223)
SEARCH CLASS(GROUP) GID(55)
```

## Superuser authority

You can assign superuser authority in three ways:

- Using resource profiles in the UNIXPRIV class (preferred method).
- Using the BPX.SUPERUSER resources in the FACILITY class.
- Assigning a UID of 0 (least desirable method).

You might choose to assign a UID of 0 to multiple RACF user IDs. However, you should minimize the number of users you assign the UID of 0 because a user with a UID of 0 can perform any z/OS UNIX function and passes all z/OS UNIX security checks.

**Guideline:** Instead of assigning a UID of 0, set z/OS UNIX user limits and manage superuser privileges through UNIXPRIV profiles. See [“Using UNIXPRIV class profiles to manage z/OS UNIX privileges”](#) on page 527 for more information.

For additional details, see [Superusers in z/OS UNIX](#) in *z/OS UNIX System Services Planning*.

Users running with the trusted or privileged attribute (for example, started tasks or jobs assigned by a RACROUTE REQUEST=VERIFY exit) are considered z/OS UNIX superusers even if their assigned UID is a value other than 0.

## Setting z/OS UNIX user limits

You can control the amount of resources consumed by certain z/OS UNIX users by setting individual limits for these users. The resource limits for the majority of z/OS UNIX users are specified in the BPXPRMxx member of SYS1.PARMLIB. These limits apply to all users except those with UID 0 (superuser authority). Rather than assigning superuser authority to application servers and other users so they can exceed BPXPRMxx limits, you can individually set higher limits to these users. Setting user limits allows you to minimize the number of assignments of superuser authority at your installation and reduces your security risk.

You can specify z/OS UNIX user limits by choosing options on the ADDUSER or ALTUSER commands. The limits are stored in the OMVS segment of the user profile. The following limits can be set in the OMVS user segment:

### **ASSIZEMAX**

Maximum address space size (RLIMIT\_AS)

### **CPUTIMEMAX**

Maximum CPU time (RLIMIT\_CPU)

### **FILEPROCMAX**

Maximum number of files per process

### **MEMLIMIT**

Maximum number of bytes of non-shared memory per user

### **MMAPAREAMAX**

Maximum memory map size

### **PROCUSERMAX**

Maximum number of processes per UID

### **SHMEMMAX**

Maximum number of bytes of shared memory per user

### **THREADSMAX**

Maximum number of threads per process

Once you have set individual user limits for users who require higher resource limits, you should consider removing their superuser authority. You should also reevaluate your installation's BPXPRMxx limits and consider reducing these limits. See [z/OS UNIX System Services Planning](#) for more information.

## Protected user IDs

You can define *protected* user IDs for started procedures associated with z/OS UNIX, such as the kernel, the initialization started procedure, and important daemons that are critical to the availability of your z/OS UNIX system. This prevents these user IDs from being revoked through inadvertent or malicious incorrect passwords and password phrases, or from being used for other purposes, such as logging on to the system. For more information, see [“Defining protected user IDs” on page 74](#).

## Controlling the use of shared UNIX identities

---

When you allow users to share UIDs, you lose the ability to control user access at an individual level. Users of a shared UID are treated as the same user during z/OS UNIX security checks.

**Guideline:** Avoid using shared (non-unique) UIDs and GIDs because they result in the loss of user accountability and decrease security. If shared UIDs and GIDs already exist at your installation, make an effort to minimize their use. Use the IRRDBU00 reports called "UIDS" and "GIDS" to find occurrences of shared IDs, and change them to unique IDs where appropriate.

If you want to implement automatic assignment of unique IDs, you must prevent the sharing of UNIX UIDs and GIDs. For details, see [“Enabling automatic assignment of unique UNIX identities” on page 518](#).

## Sharing IDs

By default, RACF does not prevent the sharing of UIDs and GIDs among any number of users or groups. However, you can enforce unique UNIX identifiers by defining a profile called SHARED.IDS in the UNIXPRIV class.

### Rules:

1. You must define the SHARED.IDS profile to enable each method of automatic assignment of unique UNIX identities. (See [“Automatically assigning unique IDs using RACF commands” on page 518](#) and [“Automatically assigning unique IDs through UNIX services” on page 520](#).)

2. To control uniqueness for automatic assignment of unique IDs using *RACF commands*, the RACF database must be at least at stage 2 of application identity mapping (AIM).

To control uniqueness for automatic assignment of unique IDs by *UNIX services*, the RACF database must be at AIM stage 3.

If you attempt to assign a UID or GID while the SHARED.IDS profile is defined but the RACF database is not at least at AIM stage 2, the command fails and message IRR52176I is issued.

For details about using the IRRIRA00 utility to advance the RACF database to AIM stage 2 or stage 3, see [z/OS Security Server RACF System Programmer's Guide](#)

3. RACF can enforce uniqueness of the UIDs and GIDs assigned using RACF TSO commands, RACF ISPF panels, or the R\_admin callable service (IRRSEQ00). RACF also assigns unique IDs through the following SAF callable services when FACILITY class profile BPX.UNIQUE.USER is defined.

- getUMAP (IRRSUM00)
- getGMAP (IRRSUM00)
- initUSP (IRRSIU00)

RACF does not enforce uniqueness of UIDs and GIDs assigned by installation programs that invoke the ICHEINTY or RACROUTE macros.

4. The maximum number of user IDs that can share a UID (or groups that share a GID) is 129 assuming a length of 8 characters for each. More user IDs or groups can share if the average length is less than 8 characters each. Once this limit is reached, you might consider combining user ID functions, such as those of started tasks or daemons, to reduce the number of user IDs sharing the same UID. Another option is to administer UNIXPRIV profiles that grant superuser authorities to reduce your need to share UID 0. For more information, see [“Using UNIXPRIV class profiles to manage z/OS UNIX privileges” on page 527](#).

## Defining the SHARED.IDS profile in the UNIXPRIV class

To control the use of shared IDs, define a profile called SHARED.IDS in the UNIXPRIV class. You must define this profile to enable each method of automatic assignment of unique UNIX identities. (See [“Automatically assigning unique IDs using RACF commands”](#) on page 518 and [“Automatically assigning unique IDs through UNIX services”](#) on page 520.)

Generic characters cannot be used in the profile name. Because the UNIXPRIV class must be RACLISTed, you must refresh the class after defining or altering the SHARED.IDS profile. If the UNIXPRIV class is not already active and RACLISTed, use the following commands to implement the SHARED.IDS profile:

### Example:

```
RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)
```

If the UNIXPRIV class is already active and RACLISTed, issue the following commands to implement the SHARED.IDS profile:

### Example:

```
RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

Once you define the SHARED.IDS profile, the default behavior of the ADDUSER, ALTUSER, ADDGROUP, and ALTGROUP commands is changed for the UID and GID options of the OMVS operand. Any attempt to assign an ID already in use fails with message IRR52174I being issued. Similarly, if you attempt to assign the same ID to a group of names on a single command, the command fails with message IRR52185I being issued.

Once you define the SHARED.IDS profile, if you want to make an exception to the enforcement of UNIX identity uniqueness, you must use the SHARED operand.

## Using the SHARED operand

Once you define the SHARED.IDS profile, if you want to make an exception and create a shared ID (as might be the case for UID 0), you must use the SHARED operand when you add or modify the OMVS segment of a user or group.

### Examples:

```
ADDUSER SUPERONE OMVS(UID(0) SHARED HOME(/) PROGRAM(/bin/echo)) NOPASSWORD
ALTGROUP DUDES OMVS(GID(99) SHARED)
```

To specify the SHARED operand, you must have the SPECIAL attribute or at least READ authority to the SHARED.IDS profile in the UNIXPRIV class.

**Example:** To authorize another user to create a user or group with a shared UNIX ID, issue the following commands:

```
PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(userid) ACCESS(READ)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

If specified, the SHARED operand is ignored when any of the following conditions are true:

- The SHARED.IDS profile is not RACLISTed.
- The UID or GID operand is omitted.
- The specified UID or GID value is unique.
- The specified UID or GID value is identical to the current UID or GID value.

## Enabling automatic assignment of unique UNIX identities

**Guideline:** Assign a unique UID for each user and a unique GID for each group that needs access to z/OS UNIX functions and resources. Assigning unique IDs rather than shared IDs improves overall security and increases user accountability.

If you choose not to define unique IDs for each user of UNIX functions, you can enable RACF to automatically generate unique UIDs and GIDs for you. There are two methods for automatically assigning unique IDs and you can use both methods together on the same system:

- **Method 1:** Enable RACF to automatically assign unique IDs when you issue the following RACF commands with the OMVS operand:
  - ADDUSER and ALTUSER commands
 

Specify the OMVS(AUTOUID) option to have RACF assign a unique UID to the user and store the UID in the OMVS segment of the user profile.
  - ADDGROUP and ALTGROUP commands
 

Specify the OMVS(AUTOGID) option to have RACF assign a unique GID to the group and store the GID in the OMVS segment of the group profile.

To use this method, the RACF database must be at least at AIM stage 2. For implementation details, see [“Automatically assigning unique IDs using RACF commands”](#) on page 518.

- **Method 2:** Enable RACF to automatically assign unique IDs when users without OMVS segments access the system to use certain UNIX services. This method provides unique IDs for users who need them to access UNIX functions and resources, and requires no administrative intervention each time a unique ID is assigned.

You can also use this method to automatically add common information to the OMVS segment of the users who are assigned unique UIDs.

To use this method, the RACF database must be at least at AIM stage 3. For implementation details, see [“Automatically assigning unique IDs through UNIX services”](#) on page 520.

### Automatically assigning unique IDs using RACF commands

RACF can automatically generate a unique ID value in the OMVS segment of a user or group upon your request. Do this by defining a profile called BPX.NEXT.USER in the FACILITY class (see [“Setting up the BPX.NEXT.USER profile”](#) on page 519) and then specifying the following command options:

- OMVS(AUTOUID) option of the ADDUSER and ALTUSER commands
- OMVS(AUTOGID) option of the ADDGROUP and ALTGROUP commands

#### Examples:

```
ADDUSER MARCY OMVS(HOME(/u/marcy) PROGRAM(/bin/sh) AUTOUID)
ALTUSER COLDEN OMVS(AUTOUID)
ADDGROUP DACKS OMVS(AUTOGID)
ALTGROUP FORTY6RS OMVS(AUTOGID)
```

Upon successful command completion, informational message IRR52177I is issued to indicate the assigned value.

#### Example:

```
IRR52177I User MARCY was assigned an OMVS UID value of 5344.
```

For the ALTUSER and ALTGROUP commands, the AUTOUID and AUTOGID options cannot be used to change the ID value if one exists for the user. However, it is not considered an error if the existing ID is unique, meaning it is not shared. If it is not unique, the command fails and message IRR52178I is issued.

If you attempt to use the AUTOUID or AUTOGID option with a list of users or groups, the command will fail with message IRR52184I being issued.

**Example (incorrect):**

```
ADDUSER (TOM DICK HARRY) OMVS(AUTOUID)
```

**Notes:**

- The RACF database must be at least at stage 2 of application identity mapping (AIM).
- Implementing SHARED.IDS and BPX.NEXT.USER is a prerequisite to successful automatic assignment of unique UNIX identities.
- The AUTOUID and AUTOUID operands cannot be specified with the SHARED operand. Doing so results in command failure and message IRR52186I being issued.
- AUTOUID is ignored if UID or NOUID is specified.
- AUTOUID is ignored if GID or NOGID is specified.

For more information on these commands, refer to the [z/OS Security Server RACF Command Language Reference](#).

**Setting up the BPX.NEXT.USER profile**

BPX.NEXT.USER is a FACILITY class profile that is used by RACF to derive unused UID and GID values. Note that the FACILITY class does not have to be active for RACF to use BPX.NEXT.USER. When creating the BPX.NEXT.USER profile, generic characters cannot be used in the name.

The APPLDATA field contains the starting UID or GID value or range of values separated by a forward slash (/). The starting value is the value RACF attempts to use in ID assignment, after determining that the ID is not in use. If it is in use, the value is incremented until an appropriate value is found.

For example, to have RACF start automatic assignment with a UID value of 1 and a GID value of 0, issue:

**Example:**

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('1/0')
```

When the maximum value of 2147483647 is reached, subsequent automatic ID assignment attempts fail and message IRR52181I is issued.

The starting value used is chosen at your discretion. For example, if UID values 0 - 2000 are already in use, and GID values 0 - 200 are already in use, you should use a UID starting value of 2001 and a GID starting value of 201.

**Example:**

```
RALTER FACILITY BPX.NEXT.USER APPLDATA('2001/201')
```

Specifying NOAUTO as a qualifier in the APPLDATA, or omitting the qualifier, prevents automatic ID assignment. For example, if you use employee serial numbers as the convention for assigning UIDs and do not want to use automatic assignment, but want to use automatic GID assignment starting at 3000, issue:

**Example:**

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('NOAUTO/3000')
```

Ranges can be useful in an RRSF environment. Specify a starting and ending value separated by a dash (–) if you want to include a range of values. Both values must be valid UID or GID values and the second must be greater than the first. Ranges can be specified independently for UIDs or GIDs.

**Examples:**

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('50000-80000/3000-10000')
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('50000/3000-10000')
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('50000-80000/3000')
RDEFINE FACILITY BPX.NEXT.USER APPLDATA('NOAUTO/3000-10000')
```



**Notes:**

- You cannot specify blanks in the APPLDATA string.
- Syntax checking of APPLDATA does not occur until AUTOUID and AUTOGID operands are specified on the ADDUSER, ALTUSER, ADDGROUP, and ALTGROUP commands.
- If you have defined BPX.NEXT.USER with incorrect APPLDATA, issuing AUTOUID or AUTOGID fails with message IRR52187I being issued.
- You can change the APPLDATA values at any time.
- After successful automatic ID assignment, RACF updates the APPLDATA starting value with either the next potential value or end of range.

## Automatically assigning unique IDs through UNIX services

**Guideline:** Assign a unique UID for each user and a unique GID for each group that needs access to z/OS UNIX functions and resources. You can accomplish this when you use AUTOUID and AUTOGID keywords on the user and group commands, as described in [“Automatically assigning unique IDs using RACF commands”](#) on page 518.

However, when you have a large number of users without OMVS segments who need access to z/OS UNIX services, such as FTP, you might choose not to assign UNIX identities in advance of their need to use the services. In these cases, use this method to enable RACF to automatically assign unique UIDs and GIDs at the time they are needed, when users without OMVS segments access certain z/OS UNIX services.

Many z/OS UNIX services, either directly or indirectly, invoke the following SAF callable services to retrieve the UID associated with a user or to retrieve the GID associated with a group:

- `initUSP` (IRRSIU00) callable service: Initialize USP
- `getUMAP` (IRRSUM00) callable service: Get UID-to-user-ID mapping
- `getGMAP` (IRRSUM00) callable service: Get GID-to-group-name mapping

RACF automatically assigns unique identities when z/OS UNIX invokes these SAF callable services to initialize the user security environment or determine a UID or GID, and *all* of the following requirements are met:

1. The RACF database is enabled for application identity mapping (AIM) stage 3. Your system programmer can use the IRRIRA00 utility to convert the RACF database to AIM stage 3. For details, see [z/OS Security Server RACF System Programmer's Guide](#).
2. The UNIXPRIV class profile SHARED.IDS is defined, and the UNIXPRIV class is active and RACLISTed.
3. The FACILITY class profile BPX.NEXT.USER is defined and its APPLDATA field has valid ID values or ranges.
4. The FACILITY class profile BPX.UNIQUE.USER is defined.
5. No OMVS segment is defined in the user or group profile.

When RACF generates and returns a new unique UID, it saves that value in the new OMVS segment of the user profile. Similarly, when RACF generates and returns a new unique GID, it saves that value in the new OMVS segment of the group profile. This ensures that the UID or GID remains assigned to the same user or group for all future uses of z/OS UNIX services.

RACF assigns unique UIDs and OMVS segments for users independently from the GIDs and OMVS segments it assigns for the user's current connect group, based on what the callable service requires. For instance, when the `initUSP` callable service calls RACF for a unique ID, a UID might be needed for the user, but the user's current connect group might already have a GID. Conversely, the callable service might require a GID for the user's current connect group but not a UID for the user.

At your option, RACF can also propagate common program, home, and other OMVS attributes to first-time z/OS UNIX users. To do so, you can define a user profile to serve as a model for OMVS segment information. When you specify the profile name in the APPLDATA field of the BPX.UNIQUE.USER profile in the FACILITY class, RACF extracts the OMVS information (other than the UID) from the model profile and saves it in the user profile at the same time it assigns the user's unique UID.



## Steps for automatically assigning unique IDs through UNIX services

**Before you begin:** Ensure that your plan to maintain UNIX access control lists (ACLs) and GID memberships includes the new unique UIDs and GIDs generated by this method.

Perform the following steps to enable RACF to automatically assign unique UIDs and GIDs for users who use z/OS UNIX services:

1. See your system programmer to ensure that your RACF database is enabled for AIM stage 3.

For details about using the IRRIRA00 utility to advance the RACF database to AIM stage 3, see [z/OS Security Server RACF System Programmer's Guide](#).

- 
2. Define the SHARED.IDS profile, if not already defined, in the UNIXPRIV class and activate and RACLIST the UNIXPRIV class. For instructions, see [“Defining the SHARED.IDS profile in the UNIXPRIV class” on page 517](#).

- 
3. Define the BPX.NEXT.USER profile in the FACILITY class, if not already defined. For instructions, see [“Setting up the BPX.NEXT.USER profile” on page 519](#).

- 
4. (Optional) Define a user profile to use as a model profile from which RACF can extract OMVS segment information. (You will specify the name of this profile in the APPLDATA field of the BPX.UNIQUE.USER profile in the FACILITY class in Step [“5” on page 522](#).)

### Guidelines:

- Define the model profile to ensure that users who are automatically assigned unique UIDs are assigned adequate OMVS information to enable them to use UNIX services.
- Omit UID for this profile. No UID is required for its intended purpose.
- Use this user profile only as the model profile for the BPX.UNIQUE.USER profile. Do not use the user ID for any other purpose.
  - Limit the use of this user ID by assigning the RESTRICTED and NOPASSWORD attributes.
  - Grant no access authority to the user ID. Do not add the user ID to RACF access lists or connect it to RACF groups that might grant resource access.
- You can specify the string &RACUID in the HOME directory path name to have RACF substitute the user ID in the path name when the OMVS segment is created. If you specify &RACUID in uppercase, RACF substitutes the user ID in uppercase. If you specify any character in the string &RACUID in lowercase, RACF substitutes the user ID in lowercase.
  - Only the first occurrence of the string is substituted.
  - If you are sharing the RACF database with a release of z/OS earlier than V2R1 that does not have APAR OA42554 installed and that uses BPX.UNIQUE.USER to assign OMVS segments, the &RACUID string is not replaced when an OMVS segment is created on that system.
  - If the substitution would result in a home directory path name that exceeds the maximum length of 1023 characters, substitution does not occur.

**Example:** The following command defines a model profile that contains a HOME value in the OMVS segment.

```
ADDUSER BPXMODEL NAME('OMVS model user profile')
OMVS(HOME('/tmp') PROGRAM('/bin/sh'))
NOPASSWORD RESTRICTED
```

**Example:** The following command defines a model profile that substitutes the user ID in lowercase in the HOME value.

```
ADDUSER BPXMODEL NAME('OMVS model user profile')
      OMVS(HOME('/u/&racuid') PROGRAM('/bin/sh'))
      NOPASSWORD RESTRICTED
```

If the user TANIA has an OMVS segment created as a result of BPX.UNIQUE.USER processing, the value assigned to the HOME operand is /u/tania.

- 
5. Define the BPX.UNIQUE.USER profile in the FACILITY class and specify the name of the model profile in the APPLDATA field.

**Example:**

```
RDEFINE FACILITY BPX.UNIQUE.USER APPLDATA('BPXMODEL')
```

**Rule:** Specify no generic characters in the BPX.UNIQUE.USER profile name.

If you do not want to propagate any OMVS information from a model profile, do not specify APPLDATA.

**Example:**

```
RDEFINE FACILITY BPX.UNIQUE.USER
```

- 
6. If the FACILITY class is RACLISTed, activate your new FACILITY profiles by refreshing the FACILITY class.

**Example:**

```
SETROPTS RACLIST(FACILITY) REFRESH
```

You need not activate and RACLIST the FACILITY class to enable automatic assignment of unique IDs. However, if the FACILITY class is already RACLISTed, you must refresh the class.

---

You have now enabled RACF to automatically assign unique IDs for users without OMVS segments when they use z/OS UNIX services. All users are now able to access z/OS UNIX services because they are automatically assigned a UID when they attempt to access a z/OS UNIX service for the first time.

If you want to prevent certain users from being able to access z/OS UNIX services, define an OMVS segment with no UID for those users. This prevents their user IDs from being automatically assigned a UID. When they attempt to use a z/OS UNIX service, the dub will fail, and a daemon will be unable to switch to these user IDs.

**Example:**

```
ALTUSER TSOADM1 OMVS(NOUID)
```

## RRSF considerations for automatic ID assignment

RACF does two things to facilitate automatic ID assignment in a RACF remote sharing facility (RRSF) environment in order to prevent different nodes from arriving at the same ID values independently for different users and then propagating these updates on the network.

1. RACF suppresses propagation of its own internal updates to the BPX.NEXT.USER profile. This prevents RACF from altering the BPX.NEXT.USER profile on other RRSF nodes when you are using automatic direction of application updates for the FACILITY class.
2. RACF alters the command image on the source node before propagating it out to other RRSF nodes. RACF inserts the generated ID value into the command image so (from the perspective of the target node) an explicit ID assignment is being requested. This protects you when automatic command direction is in effect for user and group profiles.

**Example:** Suppose you issue the following command on the source node in an RRSF environment:

```
ADDUSER SEYMOUR OMVS(AUTOUID HOME(/u/seymour) PROGRAM(bin/sh))
```

If RACF processes this ADDUSER command and arrives at a UID value of 4120 through unique ID processing, RRSF propagates the following command image to the target node:

```
ADDUSER SEYMOUR OMVS(AUTOUID UID(4120) HOME(/u/seymour) PROGRAM(/bin/sh))
```

As a result, no unique ID processing is done at the target node because it was already done at the source node. When the ADDUSER command executes at the target node, the AUTOUID option is ignored because UID is specified.

**Guidelines:** To avoid ID collisions when multiple nodes on your RRSF network are enabled for automatic assignment of unique IDs, take one of the following actions:

- Use non-overlapping ID ranges on each of the RRSF nodes to avoid conflicts when an automatic ID request is made on a given node and propagated to any other node.

Because you do not want APPLDATA updates being propagated between nodes, be careful when defining and altering BPX.NEXT.USER if automatic command direction is active for the FACILITY class. Using the ONLYAT operand (on the RALTER and RDEFINE commands) when you change BPX.NEXT.USER prevents propagation of a node's APPLDATA. ONLYAT must be used whether you are creating the BPX.NEXT.USER profile on a local or remote node.

**Example:** To define the BPX.NEXT.USER profile on the local node, issue:

```
RALTER FACILITY BPX.NEXT.USER
  APPLDATA('10000-19999/10000-19999')
  ONLYAT(.MYID)
```

**Example:** To define the BPX.NEXT.USER profile on a remote node called NODEB, issue:

```
RALTER FACILITY BPX.NEXT.USER
  APPLDATA('20000-29999/20000-29999')
  ONLYAT(NODEB.MYID)
```

- Ensure that user and group updates (specifically, those involving UID and GID requests) be performed on a single node, and propagated to other RRSF nodes from this node. Though you do not have to be concerned with the contents of BPX.NEXT.USER on any node other than the source node (whether or not automatic command direction or automatic direction of application updates is being used), all nodes should be running with SHARED.IDS implemented.

## Special RRSF considerations for automatic unique IDs

When you implement unique ID assignment using the BPX.UNIQUE.USER profile, RACF updates the OMVS segments of user and group profiles. If your installation synchronizes profiles in the USER and GROUP classes, you should enable automatic direction of application updates for these classes.

**Example:**

```
SET AUTOAPPL
RDEFINE RRSFDATA AUTODIRECT.*.USER.APPL UACC(READ)
RDEFINE RRSFDATA AUTODIRECT.*.GROUP.APPL UACC(READ)
```

If you use &RACUID in the home directory string when you define the OMVS segment of the model user ID, and this update gets propagated to a system earlier than z/OS V2R1 that does not have APAR OA42554 installed, substitution of user IDs for &RACUID does not occur when new OMVS segments are assigned on that system. For information about using &RACUID in the home directory string, see [“Steps for automatically assigning unique IDs through UNIX services” on page 521.](#)

## z/OS UNIX performance considerations

Associating RACF user IDs and groups to UIDs and GIDs has important performance considerations. It is important to use the Virtual Lookaside Facility (VLF) classes IRRUMAP and IRRGMAP and the UNIXMAP class to improve performance by avoiding sequential searches of the RACF database for UID and GID associations. If your installation shares the RACF database with only systems running z/OS, you might be able to achieve improved performance without using UNIXMAP. However, before you can avoid using UNIXMAP, you must ask your system programmer if your installation has reached stage 3 of application identity mapping by running the IRRIRA00 conversion utility. If your installation is new to RACF, you will automatically take advantage of application identity mapping at the stage 3 level without running the IRRIRA00 conversion utility, and you will not need to use UNIXMAP to achieve improved performance.

### Converting to stage 3 of application identity mapping

Your system programmer can convert your RACF database to stage 3 of application identity mapping using the IRRIRA00 conversion utility. See *z/OS Security Server RACF System Programmer's Guide* for information about running the IRRIRA00 conversion utility. In stage 3, RACF locates application identities, such as UIDs and GIDs, for users and groups by checking VLF and using an alias index that is automatically maintained by RACF. This process allows RACF to more efficiently handle authentication and authorization requests from applications such as z/OS UNIX than was possible using other methods, such as the UNIXMAP class. After your installation reaches stage 3 of application identity mapping, you will no longer have UNIXMAP class profiles on your system, and you can deactivate the UNIXMAP class.

### Using the UNIXMAP class and Virtual Lookaside Facility (VLF)

It is important to use Virtual Lookaside Facility (VLF) and the UNIXMAP class to improve performance. You might also need to use VLF and UNIXMAP class if your system programmer has not yet converted your systems for stage 3 of application identity mapping. See *z/OS Security Server RACF System Programmer's Guide* for information about converting to stage 3 by running the IRRIRA00 conversion utility.

VLF (and the associated VLF classes IRRUMAP and IRRGMAP) and the UNIXMAP class are used to map UIDs to RACF user IDs and GIDs to RACF group names. Both VLF and the UNIXMAP class can be either active or inactive. Table 36 on page 524 shows how these states affect performance. It is recommended that both the UNIXMAP class and VLF remain active, and that the VLF classes IRRUMAP and IRRGMAP should be defined to VLF.

*Table 36. The UNIXMAP class and VLF: Effects on performance for installations that have not reached stage 3 of application identity mapping*

State	Performance
<b>Active</b> UNIXMAP class  <b>Active</b> VLF	Running in this state at all times will give you the best performance.
<b>Active</b> UNIXMAP class  <b>Inactive</b> VLF	If VLF is inactive, requests for UID-to-user-ID mapping and GID-to-group-name mapping must access a UNIXMAP class profile in the database, which degrades performance. Running with VLF inactive should be done only when you need to stop VLF to make changes to it.
<b>Inactive</b> UNIXMAP class  <b>Active</b> VLF	If the UNIXMAP class is inactive, requests for UID-to-user-ID mapping and GID-to-group-name mapping must search the entire RACF database when the UID or GID specified is not found in VLF. Running in this state degrades performance severely. The inactive state for the UNIXMAP class is provided as a migration aid. After migration is complete, you should never need to run with the UNIXMAP class inactive.

Table 36. The UNIXMAP class and VLF: Effects on performance for installations that have not reached stage 3 of application identity mapping (continued)

State	Performance
<b>Inactive</b> UNIXMAP class	Running with both VLF inactive and the UNIXMAP class inactive causes requests for UID-to-user-ID mapping and GID-to-group-name mapping to default to searching the RACF database on each request. Running in this state significantly degrades performance of these functions. It could also affect other systems in a complex sharing the RACF database because of the increased I/O to the database. It is recommended that you never run in this state.
<b>Inactive</b> VLF	

You have the option to cache additional z/OS UNIX security information in VLF. This capability allows RACF to avoid accessing the RACF database when called to create a security environment for z/OS UNIX users. To use the cached user security (USP) packet, the IRRSMAP class must be defined to VLF. For more information, see [z/OS Security Server RACF System Programmer's Guide](#).

For information about the effect of certain RACF commands on VLF, see [“RACF commands for flushing a VLF cache”](#) on page 356.

For more information on VLF, see:

- [z/OS MVS Planning: Operations](#)
- [z/OS MVS Initialization and Tuning Guide](#)
- [z/OS MVS Initialization and Tuning Reference](#)

## Activating the UNIXMAP class

The UNIXMAP class is not used for UID and GID lookups until you activate it at your installation. It should be left inactive until the steps listed below are performed to initially populate the UNIXMAP class with information that might already exist in user and group profiles in the database having OMVS segments. z/OS UNIX can be active while the initial population takes place. Once the UNIXMAP class is initially populated, it should be activated.

### How to initially populate the UNIXMAP class

If your installation already uses z/OS UNIX, and has OMVS segments defined in group or user profiles, you should perform the following steps. If you do not use z/OS UNIX, you do not need to perform these steps.

To initially populate the UNIXMAP class, do the following:

1. Quiesce administrative activity against users and groups.
2. Run the database unload utility (IRRDBU00) against the database.
3. Read instructions at the beginning of the REXX migration exec (in the IRR30858 member of SYS1.SAMPLIB) concerning what data sets are to be used in your environment. After reading the exec and modifying it appropriately, run it against the database unload utility output. It produces a file containing RDEFINE and PERMIT commands that will populate the UNIXMAP class. Do not execute this command file yet.
4. Issue SETROPTS NOADDCREATOR. This is very important because you do not want the ID of the user who runs the command file produced in Step [“3”](#) on page 525 on the access list of all the profiles in this new class.
5. Execute the command file produced in Step [“3”](#) on page 525. When you execute this file, you might see messages ICH408I and ICH10102I, indicating that some profile is already defined to the UNIXMAP class. This occurs if a UID maps to more than one user ID or if a GID maps to more than one group name.
6. If SETROPTS ADDCREATOR was in effect prior to Step [“4”](#) on page 525, issue SETR ADDCREATOR now to restore that setting.

7. Issue SETROPTS CLASSACT(UNIXMAP). The UNIXMAP profiles will now be used to do UID and GID lookups. To maintain performance, it is recommended that the UNIXMAP class remain active.

Administrative activity can now be resumed against users and groups. From this point, RACF automatically keeps the UNIXMAP profiles synchronized with the user and group profiles.

## Using UNIXMAP class profiles to map UIDs and GIDs

For each UID that is defined in the OMVS segment of a user profile, a general resource profile named *Uuid* in the UNIXMAP class is automatically created. The access list of the *Uuid* profile contains all user IDs that have been assigned this UID.

For each GID that is defined in the OMVS segment of a group profile, a general resource profile named *Ggid* in the UNIXMAP class is automatically created. The access list of the *Ggid* profile contains all groups that have been assigned this GID.

These mapping profiles are used to provide a cross reference to user and group profiles. They provide RACF with a performance-sensitive method of returning information for a given UID or GID when requested by z/OS UNIX or application programs.

RACF automatically maintains these mapping profiles when UIDs and GIDs are added, changed, or deleted. The UNIXMAP class does not have to be active for this to happen. RACF does this by modifying UNIXMAP class profiles appropriately when ADDUSER, ALTUSER, DELUSER, ADDGROUP, ALTGROUP, or DELGROUP commands are issued. When RACF creates UNIXMAP profiles as a result of an ADDUSER, ALTUSER, ADDGROUP or ALTGROUP command, the user ID of the command issuer becomes the owner of the UNIXMAP profile.

For example, if the following command is issued:

```
ADDUSER ORTIZ OMVS(UID(13))
```

RACF creates a UNIXMAP profile named U13 with ORTIZ contained on the access list. If the following command is subsequently issued:

```
ALTUSER ORTIZ OMVS(UID(55))
```

RACF deletes the U13 profile and creates a U55 profile with ORTIZ contained on the access list.

In general, you should not alter these profiles. However, it is possible that they might get inadvertently deleted, or damaged by database corruption. If a profile is deleted, or if the user is not contained in its access list, RACF will not be able to retrieve information for the UID or GID that the profile represented. RACF will be unable to locate the mapping profile and will send z/OS UNIX a return code indicating that the UID or GID is not known.

If this happens, an authorized user needs to repair the damage. First, see if the user name associated with the UID or the group name associated with the GID can be determined from a message displayed by RACF. For example, suppose you received an error message associated with user ORTIZ. You should display the UID associated with the user profile for ORTIZ by entering:

```
LISTUSER ORTIZ OMVS NORACF
```

If, for example, LISTUSER displays a UID of 13, you would then enter:

```
RDEFINE UNIXMAP U13 UACC(NONE)
PERMIT U13 CLASS(UNIXMAP) ACCESS(NONE) ID(ORTIZ)
PERMIT U13 CLASS(UNIXMAP) ID(your-userid) DELETE
```

If your environment has the SETROPTS NOADDCREATOR option in effect, the second PERMIT command is not necessary because RDEFINE does not put the profile creator on the access list.

If you are unable to determine the user name or group name from a RACF message, look at the output from the database unload utility (IRRDBU00) to find the user ID or group associated with a given UID or GID. The mapping profiles should then be added, changed, or deleted as appropriate to be consistent.

## Using UNIXPRIV class profiles to manage z/OS UNIX privileges

You can define profiles in the UNIXPRIV class to grant RACF authorization for certain z/OS UNIX privileges. These privileges are automatically granted to all users with z/OS UNIX superuser authority. By defining profiles in the UNIXPRIV class, you can specifically grant certain superuser privileges with a high degree of granularity to users who do not have superuser authority. This allows you to minimize the number of assignments of superuser authority at your installation and reduces your security risk.

Resource names in the UNIXPRIV class are associated with z/OS UNIX privileges. You must define profiles in the UNIXPRIV class protecting these resources in order to use RACF authorization to grant z/OS UNIX privileges. The UNIXPRIV class must be active and SETROPTS RACLIST must be in effect for the UNIXPRIV class. Global access checking is not used for authorization checking to UNIXPRIV resources.

See *z/OS UNIX System Services Planning* for a list of the resource names available in the UNIXPRIV class, the z/OS UNIX privilege associated with each resource, and the level of access required to grant the privilege.

### Example of authorizing superuser privileges

The following examples apply to superuser privileges, except the privilege associated with the CHOWN.UNRESTRICTED resource (see [“Using the CHOWN.UNRESTRICTED profile” on page 528](#)). For example, these are the steps to authorize selected users to transfer ownership of any file.

1. Define a profile in the UNIXPRIV class to protect the resource called SUPERUSER.FILESYS.CHOWN. For example:

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.CHOWN UACC(NONE)
```

**Note:** Generic profile names are generally permitted for resources in the UNIXPRIV class, though there are certain exceptions such as the CHOWN.UNRESTRICTED resource. These examples are documented in their appropriate sections. If you want to authorize all file-system privileges, you can use generics and define a profile called SUPERUSER.FILESYS.\*\*.

2. Authorize selected users or groups as appropriate:

```
PERMIT SUPERUSER.FILESYS.CHOWN CLASS(UNIXPRIV)
      ID(appropriate-groups-and-users) ACCESS(READ)
```

3. Activate the UNIXPRIV class, if it is not currently active at your installation:

```
SETROPTS CLASSACT(UNIXPRIV)
```

**Note:** If you do not activate the UNIXPRIV class and activate SETROPTS RACLIST processing for the UNIXPRIV class, only superusers will be allowed to transfer ownership of any file.

4. You *must* activate SETROPTS RACLIST processing for the UNIXPRIV class, if it is not already active. For a complete description of this function, see [“SETROPTS RACLIST processing” on page 127](#).

```
SETROPTS RACLIST(UNIXPRIV)
```

**Note:** If SETROPTS RACLIST processing is already in effect for the UNIXPRIV class, you must refresh SETROPTS RACLIST processing in order for new or changed profiles in the UNIXPRIV class to take effect.

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

### Allowing z/OS UNIX users to change file ownerships

On z/OS UNIX systems, RACF enforces the rules for the POSIX constant called `_POSIX_CHOWN_RESTRICTED`. This means that, by default, only superusers can change the ownership of any file to any UID or GID on the system, and that general users can only change the ownership of files that they own, and only to one of their own associated GIDs. However, by defining a profile called



CHOWN.UNRESTRICTED in the UNIXPRIV class, you can allow selected users and groups to transfer ownership of files they own to any UID or GID on the system.

**Guideline:** For a more secure implementation, do not define the CHOWN.UNRESTRICTED profile.

You can define an additional profile in the UNIXPRIV class protecting a resource called SUPERUSER.FILES.CHOWN to authorize selected z/OS UNIX users to transfer ownership of any file to any UID or GID. See [“Example of authorizing superuser privileges” on page 527](#) for an example of authorizing users using the SUPERUSER.FILES.CHOWN resource.

## Using the CHOWN.UNRESTRICTED profile

To allow z/OS UNIX users to transfer ownership of files that they own, create a discrete profile in the UNIXPRIV class that is called CHOWN.UNRESTRICTED. Then add the users to the access list with the appropriate access level. Restrict this ability to users that are trusted.

The CHOWN.UNRESTRICTED profile must be a discrete profile. Any matching generic profiles are ignored.

To allow z/OS UNIX users to transfer ownership for files they own, perform the following steps:

1. Define the discrete profile in the UNIXPRIV class called CHOWN.UNRESTRICTED:

```
RDEFINE UNIXPRIV CHOWN.UNRESTRICTED UACC(NONE)
```

2. Add the user or group to the access list with the appropriate access level:

- UPDATE access is required to change ownership to UID 0.
- READ access is required to change ownership to any other UID value, or to the GID of a group to which the user is not connected.

For example:

```
PERMIT CHOWN.UNRESTRICTED CLASS(UNIXPRIV) ID(GRPX) ACCESS(READ)
```

3. Activate the UNIXPRIV class, if it is not currently active at your installation:

```
SETROPTS CLASSACT(UNIXPRIV)
```

**Note:** If you do not activate the UNIXPRIV class and activate SETROPTS RACLIST processing for the UNIXPRIV class, only superusers are allowed to transfer ownership of files to others.

4. You must activate SETROPTS RACLIST processing for the UNIXPRIV class, if it is not already active. For a complete description of this function, see [“SETROPTS RACLIST processing” on page 127](#).

```
SETROPTS RACLIST(UNIXPRIV)
```

**Note:** If SETROPTS RACLIST processing is already in effect for the UNIXPRIV class, you must refresh SETROPTS RACLIST processing in order for the CHOWN.UNRESTRICTED profile to take effect.

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

## Allowing z/OS UNIX users to read or search directories

Sometimes z/OS UNIX administrators need the ability to read and search all file system directories to manage file ownerships and permissions. It is not necessary to give such administrators RACF AUDITOR or ROAUDIT authority to provide this ability when directory permission bits and access lists do not explicitly allow access. Instead, you can define a UNIXPRIV profile covering SUPERUSER.FILES.DIRSRCH to control such access. This permission is complementary to administrator authorities provided by SUPERUSER.FILES.CHOWN and SUPERUSER.FILES.CHANGEPERMS.

**Note:** Use caution when permitting users to the DIRSRCH profile if you employ the strategy of protecting files by disallowing user access to the parent directory. Users with DIRSRCH profile permission can read



and search all directories, their access to files in all subdirectories is determined by the defined file permissions and access lists.

DIRSRCH profile permission does NOT override FSACCESS file system or security label protection.

To allow z/OS UNIX users to read and search all file system directories, regardless of file permission bits or access lists, create a profile in the UNIXPRIV class protecting a resource that is called SUPERUSER.FILESYS.DIRSRCH. Then permit users and groups with at least READ access performing the following steps.

1. Define a profile in the UNIXPRIV class.

**Example:**

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.DIRSRCH UACC(NONE)
```

2. Add the user or group to the access list with at least READ access.

**Example:**

```
PERMIT SUPERUSER.FILESYS.DIRSRCH CLASS(UNIXPRIV) ID(USER01 GRPX) ACCESS(READ)
```

3. If the UNIXPRIV class is not already active, activate and RACLIST it.

**Example:**

```
SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)
```

4. If the UNIXPRIV class is already active and RACLISTed, refresh it.

**Example:**

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

You have now given directory read and search permission to the specified users and groups.

## Configuring the group owner for new UNIX files

When a new UNIX file is created on z/OS, by default, the owning UID is initialized from the effective UID of the creating process, and the owning GID is copied from the parent directory. The POSIX standard allows the owning GID to be taken either from the parent directory, or from the effective GID of the creating process.

Many versions of UNIX and Linux® use the set-gid bit of the parent directory to determine how to set a new object's group owner. If the parent's set-gid bit is on, then the group owner is set to that of the parent directory. Otherwise, it is set from the effective GID of the process. Further, the set-gid bit for a new directory is inherited from the parent directory.

This behavior can be configured on z/OS by defining the FILE.GROUPOWNER.SETGID profile in the UNIXPRIV class.

### Using the FILE.GROUPOWNER.SETGID profile

To specify that the group owner of a new z/OS UNIX file is to come from the effective GID of the creating process, you need to set up a profile in the UNIXPRIV class called FILE.GROUPOWNER.SETGID.

#### ***Steps for setting up the FILE.GROUPOWNER.SETGID profile***

Perform the following steps to set up the FILE.GROUPOWNER.SETGID profile:

1. Define the FILE.GROUPOWNER.SETGID profile. Generic characters cannot be used in this profile name.

```
RDEFINE UNIXPRIV FILE.GROUPOWNER.SETGID
```

2. Activate the UNIXPRIV class if it is not currently active at your installation.

```
SETOPTS CLASSACT(UNIXPRIV)
```

3. Activate the SETROPTS RACLIST processing for the UNIXPRIV class if it is not already active.

```
SETOPTS RACLIST(UNIXPRIV)
```

If SETROPTS RACLIST processing is already in effect for the UNIXPRIV class, you must refresh SETROPTS RACLIST processing in order for the FILE.GROUPOWNER.SETGID profile to take effect.

```
SETOPTS RACLIST(UNIXPRIV) REFRESH
```



**Attention:** When a new file system is mounted, you must turn on the `set-gid` bit of its root directory if you want new objects within the file system to have their group owner set to that of the parent directory.

## Protecting file system resources

z/OS UNIX file system resources, such as z/OS UNIX files and directories, can be protected by permission bits that are stored within the file system itself in the file security packet (FSP) and by access control lists (ACLs) that are also stored in the file system.

Permission bits allow specification of read authority, write authority, or search authority for a directory. They also allow specification of read, write, or execute for a file. There are three sets of bits so that separate authorities can be specified for the owner of the file or directory, the owning group, and everyone else (like RACF's universal access authority, or UACC). The owner is represented by a UID. The owning group is represented by a GID. Access checking compares the user's UID and GID to the ones stored in the FSP.

Access control lists (ACLs) are used in conjunction with permission bits. ACLs provide a more granular level of access control for files and directories, allowing you to control access by individual UIDs and GIDs. Authorization checking for ACLs is done by RACF. However, you administer ACLs using z/OS UNIX commands, particularly the `setfacl` and `getfacl` commands. For several examples of using these commands to manage ACLs, see [z/OS UNIX System Services Planning](#). ACLs are automatically deleted whenever a file is deleted. This occurs even when a file system with ACLs is mounted on a downlevel system.

z/OS UNIX files and directories can also be protected using security labels. See [z/OS Planning for Multilevel Security and the Common Criteria](#) for more information.

## Administering ACLs

The z/OS UNIX `setfacl` command is used to create, modify, and delete ACLs. The z/OS UNIX `getfacl` command is used to display the contents of ACLs. To create and administer an ACL for a file, you must either be the file owner or you must have superuser authority by having UID(0) or READ access to SUPERUSER.FILESYS.CHANGEPERMS in the UNIXPRIV class.

You can also use `setfacl` to create *default* (or *model*) ACLs for directories. When new objects are created within the directory, the default ACL is automatically inherited by the new object. See [z/OS UNIX System Services Planning](#) for complete information on using ACLs.

You must activate the FSSEC class before ACLs can be used in access decisions. You can define and display ACLs while the FSSEC class is inactive, however they will not be used for authorization checking. Similarly, if you have defined *default* ACLs on directories, the ACLs will be inherited by new objects while the FSSEC class is inactive but they will not be used for authorization checking.

The following command can be used to activate the FSSEC class.

### Example:

```
SETOPTS CLASSACT(FSSEC)
```

When a security decision is needed, the file system calls RACF, supplying the ACL, if present, and the FSP. RACF provides authorization checking and auditing, and then returns control to the file system. See [“Authorization checking for RACF-protected resources”](#) on page 718 for details.

## Controlling access to file system resources for restricted users

Users with the RESTRICTED attribute cannot access protected resources they are not specifically authorized to access. (See [“Defining restricted user IDs”](#) on page 75 for more information.) However, the RESTRICTED attribute has no effect when a user accesses a z/OS UNIX file system resource; the file's "other" permission bits can allow access to users who are not explicitly authorized. To ensure that restricted users do not gain access to z/OS UNIX file system resources through "other" bits, you must perform the following steps.

### ***Steps for controlling access to file system resources for restricted users***

Perform the following steps to prevent restricted users from accessing file system resources based on the file's "other" bits:

1. Define a resource called RESTRICTED.FILESYS.ACCESS in the UNIXPRIV class with UACC(NONE). To prevent all restricted users, do not permit any users or groups.

#### **Example:**

```
RDEFINE UNIXPRIV RESTRICTED.FILESYS.ACCESS UACC(NONE)
```

- 
2. If needed, explicitly allow certain restricted users to access certain files using the usual authorization method of adding those users, or one of their groups, to the file's ACL using the `setfacl` command. (See [z/OS UNIX System Services Command Reference](#) for details.)

#### **Example:**

```
setfacl -m user:thabo:rwX MyFile
```

Authorization changes made using the `setfacl` command take effect immediately.

- 
3. If needed, grant exceptions to certain restricted users to allow them to gain access based on the file's "other" bits. Add those users, or one of their groups, to the access list with READ authority.

#### **Example:**

```
PERMIT RESTRICTED.FILESYS.ACCESS CLASS(UNIXPRIV) ID(RSTDUSER) ACCESS(READ)
```

Do not attempt to *deny* access to certain restricted users by defining this resource with UACC(READ) and then permitting those users with access of NONE. The UACC of a resource cannot be used to allow access when the user is restricted.

- 
4. Refresh the UNIXPRIV class to activate changes from Steps [“1”](#) on page 531 and [“3”](#) on page 531.

#### **Example:**

```
SETOPTS RACLIST(UNIXPRIV) REFRESH
```

For any given z/OS UNIX process, the result of the first check to the RESTRICTED.FILESYS.ACCESS resource will be cached for the life of the process. Therefore, subsequent authorization changes to this resource will not take effect for that process.

---

## Overriding SUPERUSER.FILESYS authority with ACLs

Any user who is not a superuser with UID (0) or the file owner and is denied access through the ACL can still access a file system resource if the user has sufficient authority to the SUPERUSER.FILESYS resource in the UNIXPRIV class. To prevent this, you can force RACF to use your ACL authorizations to override a user's SUPERUSER.FILESYS authority by performing the following steps.

### Steps for overriding SUPERUSER.FILESYS authority with ACLs

Perform the following steps to prevent users from using their SUPERUSER.FILESYS authority to access file system resources they are specifically unauthorized to access through the ACL:

1. Define a resource called SUPERUSER.FILESYS.ACLOVERRIDE in the UNIXPRIV class with UACC(NONE). To prevent all users, do not permit any users or groups.

#### Example:

```
RDEFINE UNIXPRIV SUPERUSER.FILESYS.ACLOVERRIDE UACC(NONE)
```

2. If needed, grant exceptions to certain users or groups to allow them to gain access based on their SUPERUSER.FILESYS authority. Add those users or groups to the access list with the same level of access they require for the SUPERUSER.FILESYS resource.

#### Example:

```
PERMIT SUPERUSER.FILESYS.ACLOVERRIDE CLASS(UNIXPRIV) ID(PER) ACCESS(READ)
```

See [z/OS UNIX System Services Planning](#) for details about authorizing users for the SUPERUSER.FILESYS resource.

3. Refresh the UNIXPRIV class to activate changes from Steps “1” on page 532 and “2” on page 532.

#### Example:

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

SUPERUSER.FILESYS.ACLOVERRIDE is checked only when a user's access was denied by a matching ACL entry based on the user's UID or one of the user's GIDs. If the user's access was denied by the file's permission bits, SUPERUSER.FILESYS is checked. See [“Authorizing access to z/OS UNIX files and directories” on page 728](#) for details.

## Restricting access to a zFS file system

You can restrict access to a z/OS File System (zFS) file system by defining a general resource profile in the FSACCESS class. This enables you to use RACF commands to restrict z/OS UNIX access to the specified zFS file system for most users while allowing selected users and groups to remain eligible to access the file system. This method supports an improved audit posture by enabling the RACF administrator to demonstrate a single point of control for restricting access to one or more file systems that might contain sensitive or personal data.

When you define an FSACCESS profile, you restrict access to the file system, which includes all of its files and directories, at only the file system level. By contrast, the z/OS UNIX administrator can use the **setfacl** command to control access at the file system level and to control access to any of its files and directories on an individual resource basis.

When a zFS file system is protected by an FSACCESS profile with UACC(NONE), only users and groups with UPDATE access authority or higher, and users with the AUDITOR or ROAUDIT attribute, are eligible to access the file system. Eligible users are then subject to the usual authorization checking, which includes

checking for superuser authority, ownership, permission bits, access control lists (ACLs), and UNIXPRIV authorities.

When a zFS file system is protected by an FSACCESS profile and a user has insufficient access authority to it, no further authorization checking is done, and z/OS UNIX access to the protected file system, including all of its files and directories, is denied. Note that while superuser authority can be used to mount a file system protected by an FSACCESS profile, it is insufficient authority to access it. Also, note that access authority to the MVS data set that contains the file system is unaffected when you define the FSACCESS profile.

You need not authorize UPDATE access for users with the AUDITOR or the ROAUDIT attribute. These users are exempt from the access restrictions enforced by the FSACCESS profile.

## Steps for restricting access to a zFS file system

**Before you begin:** For each zFS file system, ask the z/OS UNIX administrator for the name of the MVS data set where the file system is stored.

Perform the following steps to restrict access to a zFS file system.

1. Define a profile in the FSACCESS class to protect each zFS file system. The profile name is the name of the MVS data set that contains the file system.

### Example:

```
RDEFINE FSACCESS OMVS.ZFS.WEBSRV.TOOLS UACC(NONE)
```

If multiple file systems are stored in data sets with similar names, you can define a generic profile name to protect multiple file systems. Before you define a generic profile in the FSACCESS class, enable generics for the class, as follows.

### Example:

```
SETROPTS GENERIC(FSACCESS)
RDEFINE FSACCESS OMVS.ZFS.WEBSRV.** UACC(NONE)
```

- 
2. Authorize selected users and groups with UPDATE access.

### Example:

```
PERMIT OMVS.ZFS.WEBSRV.TOOLS CLASS(FSACCESS) ID(GROUPB USER19) ACCESS(UPDATE)
```

- 
3. Activate your profile changes in the FSACCESS class, as follows.

- If the FSACCESS class is not already active, activate and RACLIST it.

### Example:

```
SETROPTS CLASSACT(FSACCESS) RACLIST(FSACCESS)
```

- If the FSACCESS class is already active and RACLISTed, refresh it.

### Example:

```
SETROPTS RACLIST(FSACCESS) REFRESH
```

---

You have now restricted access to a zFS file system to only the specified users and groups.

## Restricting access to all zFS file systems

In addition to restricting access to individual zFS file systems, your installation might consider adding a *top* generic profile in the FSACCESS class to restrict access to all zFS file systems. The profile name might consist of two asterisks (\*\*) as the MVS data set name. By defining and activating a top generic profile, you will disallow access for all users to any zFS file system that is not protected by another FSACCESS profile.

**Important:** Before implementing a top generic profile in the FSACCESS class, work with the z/OS UNIX administrator and carefully plan your profile names and access lists.

**Example:**

```
RDEFINE FSACCESS ** UACC(NONE)
```

## Restricting execute access in a zFS or TFS file system

You can prevent users from executing any file in a z/OS File System (zFS) file system or Temporary File System (TFS) by defining a general resource profile in the FSEXEC class. This enables you to use RACF commands to restrict z/OS UNIX access to the specified file system for most users while allowing selected users and groups to remain eligible for execute access. This method supports an improved audit posture by enabling the RACF administrator to demonstrate a single point of control for restricting execute access to one or more file systems that might contain authorized code, or code of unknown origin.

When a file system is protected by an FSEXEC profile with UACC(NONE), only users and groups with UPDATE access authority or higher are eligible for execute file access. Eligible users are then subject to the usual authorization checking, which includes checking for superuser authority, ownership, permission bits, access control lists (ACLs), and UNIXPRIV authorities.

**Note:** FSEXEC restriction does not apply to file systems mounted with the '-s nosecurity' option.

When a file system is protected by an FSEXEC profile and a user has insufficient access authority to it, RACF denies file execution access regardless of other user authorizations. Superuser, auditor, or read-only auditor privilege does not override FSEXEC denial of access.

## Steps for restricting execute access in a zFS or TFS file system

**Before you begin:** For each file system you intend to protect, ask the z/OS UNIX administrator administrator for the FILESYSTEM name specified on the MOUNT statement.

Perform the following steps to restrict access.

1. Define a profile in the FSEXEC class to protect the file system. The profile name is the FILESYSTEM name specified on the MOUNT statement. Since profiles in the FSEXEC class are case sensitive, ensure the profile name on the RDEFINE command matches the letter case of the name on the MOUNT statement.

**Example:**

```
RDEFINE FSEXEC /tmp UACC(NONE)
```

If multiple file systems are have similar names, you can define a generic profile name to protect multiple file systems. Before you define a generic profile in the FSEXEC class, enable generics for the class, as follows.

**Example:**

```
SETROPTS GENERIC(FSEXEC)
RDEFINE FSEXEC OMVS.ZFS.ADMIN.** UACC(NONE)
```

- 
2. For selected users and groups who require execute access, authorize them with UPDATE access.

**Example:**

```
PERMIT OMVS.ZFS.ADMIN.** CLASS(FSEEXEC) ID(USER019 GROUPADM) ACCESS(UPDATE)
```

---

3. Activate your profile changes in the FSEEXEC class, as follows.

- If the FSEEXEC class is not already active, activate and RACLIST it.

**Example:**

```
SETROPTS CLASSACT(FSEEXEC) RACLIST(FSEEXEC)
```

- If the FSEEXEC class is already active and RACLISTed, refresh it.

**Example:**

```
SETROPTS RACLIST(FSEEXEC) REFRESH
```

---

You have now restricted execute access to a file system to only the specified users and groups.

## z/OS UNIX application considerations

---

z/OS UNIX supports two fundamental types of application servers:

- Multithreaded application servers
- Single-threaded application servers

A *multithreaded* application has multiple sequential flows of control. In this family of applications, the application server can process more than one unit of work at a time.

A *single-threaded* application has one sequential flow of control. In this family of applications, the application server processes one unit of work at a time.

z/OS UNIX provides the `pthread_security_np` callable service and support through the C runtime library that enables *unauthorized* multithreaded applications to create and delete a RACF security environment in a fashion that is mediated and controlled by the kernel and RACF.

**Note:** The term *unauthorized* refers to applications that are not APF-authorized and do not run in supervisor state or in a system storage protection key.

The `pthread_security_np` callable service enables multithreaded applications to customize the security environment of a thread, allowing it to execute under a different RACF identity than the server. You need to authorize the server to use the `pthread_security_np` service.

**For more information:**

- Administrative considerations of the z/OS UNIX `pthread_security_np` callable service are discussed in *z/OS UNIX System Services Planning*.
- Additional information regarding the `pthread_security_np` service is discussed in *z/OS UNIX System Services Programming: Assembler Callable Services Reference*. The C language support for this service is discussed in *z/OS C/C++ Runtime Library Reference*.

## Threads and security

An application that uses the `pthread_security_np` service can customize the RACF identity of a thread. The server initiates a thread that processes the client's request. If the server customizes the thread initiated for the client with the client's RACF identity, any resource access decisions to RACF protected resources are made using the client's RACF identity and authorizations.

Depending on the trust you place in an application, you have the option of enforcing whether to use both the application server's RACF identity *and* the RACF identity of the client in resource access control decisions.

You can choose one of the following:

- Only the RACF user ID of the client is used in local resource access control decisions made by RACF.
- Both the RACF user ID of the server and the RACF user ID of the client are used in local resource access control decisions.

The use of the `pthread_security_np` service is in part protected by the RACF FACILITY class profile BPX.SERVER.

- If the RACF user ID that is associated with an application server is permitted with UPDATE access to this profile, the application server is allowed to establish a thread-level (task-level) security environment for clients connecting to the server. With UPDATE authority to BPX.SERVER in the RACF FACILITY class, the server can act as a *surrogate* of the client. This means that the identity of the thread associated with the request from the server's client executes with the RACF user ID of the server's client.

The RACF identity of the client determines the type of access allowed to system resources (such as data sets) and z/OS UNIX resources (such as file system resources), which are accessed by the client's thread in the server.

- READ access allows the server to establish a thread-level security environment for the clients it services. However, the user ID of the server and the user ID of the client must be authorized to the resources the server accesses. A thread level security environment in which both the client's and server's identities are used in the access control decision, but a password was not supplied by the client, is called an *unauthenticated client* security environment.

Depending on the design and implementation of the client/server application, a client might need to supply an authenticator to the server.

For example, the client might be prompted to supply a password or a password substitute, such as a RACF PassTicket, to the server to prove its identity. If a RACF password or PassTicket is specified as an option on the `pthread_security_np` service, and the password or PassTicket is valid for the client user ID, *only* the RACF user ID of the client is used in rendering access control decisions.

This task level security environment created by an application server is called an *authenticated client* security environment. Because the client has trusted the application server sufficiently to supply a RACF password or PassTicket to the server, the server is granted the capability of acting as a surrogate for that client.

This capability enables you to determine:

- On behalf of which user IDs the server can act
- What resources the server can access when acting on behalf of one of its clients

Potentially, for additional security checking, two audit records can be produced to audit:

- The client accessing the resource
- The server accessing the resource on behalf of the client

If you choose to implement this additional security checking, you might need to authorize the identity associated with the application server to the resource profiles that protect the resources accessed by the server on behalf of its clients.

See [z/OS UNIX System Services Planning](#) for a complete description of the administrative planning steps and requirements for using the `pthread_security_service`.

## Application services and security

Through z/OS UNIX, the C runtime library, and RACF, the following services are available that enable application servers on z/OS to:

- Map a DCE identity to a RACF user ID; or map a RACF user ID to a DCE identity.



- Map an application identity, such as a Lotus Notes for z/OS short name or an Novell Directory Services for OS/390 user name, to a RACF user ID; or map a RACF user ID to an application identity.

See [“RACF support for NDS and Lotus Notes for z/OS” on page 251](#) for more information.

- Invoke RACF authorization services

The `convert_id_np` (BPX1CID) callable service is the z/OS UNIX service that converts a DCE principal's UUID pair (cell UUID and principal UUID) to the RACF user ID that has been cross linked with the UUID pair. This service also accepts a RACF user ID and returns the corresponding DCE UUIDs that have been cross-linked with the RACF user ID. This z/OS UNIX service is also supported through the C runtime library via the `__convert_id_np()` function call.

These services use the SAF callable service, `R_dceruid` (IRRSUD00), which accesses the appropriate profile information stored in the RACF database, to perform the identity conversion. The use of these identity mapping functions is RACF-protected. The `R_dceruid` callable service performs an authorization request to determine if the user ID associated with the application server is authorized to use the identity conversion service. Controlling the use of these conversion services is discussed in [“`R\_dceruid` \(IRRSUD00\) callable service” on page 608](#).

For more information about the `convert_id_np` (BPX1CID) callable service, see [z/OS UNIX System Services Programming: Assembler Callable Services Reference](#). The C language support for the `__convert_id_np()` is discussed in [z/OS C/C++ Runtime Library Reference](#).

## Application authorization service

The application developer might want to use RACF to control access to a set of resources that are managed by an application server. Through z/OS UNIX, the `auth_check_resource_np` (BPX1ACK) callable service enables application servers to invoke RACF authorization services. This service is also supported by the C runtime library through the `__check_resource_auth_np()` function call.

The service allows z/OS UNIX application servers to perform authorization checking for resources that are defined to RACF general resource classes. For more information on the `auth_check_resource_np` callable service, see [z/OS UNIX System Services Programming: Assembler Callable Services Reference](#).

## Restrictions of RACF client ACEE support

As the security administrator, you need to be aware of restrictions of the RACF client ACEE support, in which both the application server's RACF identity and the client's RACF identity are used in resolving access decisions.

- RACROUTE REQUEST=FASTAUTH processing does not check both the server and client RACF identities automatically.

Unauthorized application servers cannot use the RACROUTE REQUEST=LIST instruction to build in-storage profiles for RACF defined resources. Profiles must reside in storage before RACROUTE REQUEST=FASTAUTH can verify a user's access to a resource.

- The client/server relationship is not propagated from the application server.

If you have implemented access control to resources that use both the server's RACF identity and the client's RACF identity in an access control decision, application servers that you do not trust should be treated as *end points*. These servers should not be allowed to submit batch jobs or use the services of other servers that run exclusively under the identity of the client. You must ensure that applications servers that do not meet this criteria are not authorized to the profile BPX.SERVER in the RACF FACILITY class.

## Auditing z/OS UNIX security events

You can issue the SETROPTS LOGOPTIONS command to enable auditing options for z/OS UNIX security events by specifying the following classes:

- DIRACC

- DIRSRCH
- FSOBJ
- FSSEC
- IPCOBJ
- PROCACT
- PROCESS

No profiles can be defined in these classes. With the exception of FSSEC class, these classes are not used for authorization checking and need not be active to control auditing.

The FSSEC class, when active, also controls whether ACLs are used during authorization checks for z/OS UNIX files and directories.

Use the SETROPTS LOGOPTIONS command to specify logging options for any of these classes. Additionally, you can use the SETROPTS AUDIT command to control auditing of some events for the FSOBJ and the PROCESS classes. For more information on the SETROPTS command, see [z/OS Security Server RACF Command Language Reference](#).

**Note:** Audit records are always written for security decisions made during RACF callable services involving resources in these z/OS UNIX classes when the user has the UAUDIT attribute, regardless of the LOGOPTIONS and AUDIT settings.

For details about RACF auditing for z/OS UNIX security events, see [Auditing for z/OS UNIX System Services](#) in [z/OS Security Server RACF Auditor's Guide](#).

For a brief description of each class, see [“Supplied resource classes for z/OS systems”](#) on page 683.

## Chapter 22. RACF and digital certificates

This topic provides information on using RACF to manage digital certificates.

### Overview of digital certificates

In a client-server network environment, entities identify themselves with digital certificates using a public key protocol, such as Secure Sockets Layer (SSL). Public key protocols are based on asymmetric encryption, in which mathematical properties are used to produce an encryption *key pair*, a value formed by pairing a public key with a related private key. The public key, as implied by its name, is public information that can be disseminated freely. The private key, on the other hand, is private and should never be revealed to anyone other than the owner of the key pair.

### Public and private keys

A public key and a related private key are numerically associated with each other. Therefore, any data encrypted using one of the key values can only be decrypted using the other key value. Network protocols take advantage of this in the following ways:

- Data can be securely sent from one party to another if the sending party knows the public key of the receiving party. The sender encrypts the data with the public key before sending. Upon receipt, the receiving party recovers the data by decrypting it with the private key. Because the intended recipient is the only party that possesses the private key, only the intended recipient can recover the data.
- One party can digitally sign data by encrypting a copy of the data using her own private key. If the signer's public key is known, the signature can be verified by decrypting the signed data using the signer's public key. If the recovered data matches the expected value (the original data), then it is the data signed by the original party, not forged by another, because only the original party has the matching private key.

In practical terms, symmetric encryption algorithms, such as Data Encryption Standard (DES), perform much faster than asymmetric encryption algorithms. Therefore, public key protocols use a combination of symmetric and asymmetric encryption. For example, in SSL, the message data is symmetrically encrypted only after asymmetric encryption is used to exchange the symmetric encryption key. Also, to reduce the size of the message transmitted, the data to be digitally signed is compressed using a one-way hashing function before being encrypted with the signer's private key. The signature verifier then performs the same hashing function on the recovered data before comparing the signature.

### X.509 certificates

Public keys can be freely disseminated. In fact, the success of the various public key protocols requires a systematic and trustworthy way of distributing public keys and securely storing their associated private keys. The X.509 digital certificate is the packaging that enables the distribution of a single public key. The X.509 standard is the subsection of the International Telecommunication Union (ITU) X.500 directory standard that defines certificates.

The X.509 digital certificate is a data structure that contains, at minimum, the following fields:

- The distinguished name of the owner of the public key, also called the *subject's name*
- The distinguished name of the issuer of the certificate, also called the *issuer's name*
- The public key itself
- The time period during which the certificate is valid, also called the *validity period*
- The certificate's serial number as designated by the issuer
- The issuer's digital signature

In addition to these required fields, an X.509 certificate might contain one or more extensions that hold information about how the key is to be used (a KeyUsage extension) or how the certificate authority conducts its business (a CertificatePolicies extension).

In its simplest form, a digital certificate is a binding between a named *entity* (a person or device) and a public key. It is a declaration that, for example, party A owns public key 123. Digital certificates can be issued by *certificate authorities* or they can be *self-issued*. Certificate authorities (CAs) are often well-known commercial organizations or they can be local or internal organizations. When a certificate authority uses its private key to sign and issue a certificate, it makes the declaration that binds the entity (subject) to its public key. When an organization issues its own certificate with itself as subject and issuer, signing with its own private key, the certificate is called a *self-signed* certificate.

## Certificate hierarchies

Certificate authorities digitally sign the certificates they issue using their own private key. Thus, another party can verify the information in a certificate, including its extensions, by validating the signature on the certificate with the certificate authority's own public key. The other party gets the certificate authority's public key from a certificate issued to the certificate authority and does a signature check that might involve the public key from yet another certificate. The chain of verification can be quite long, depending on the *certificate hierarchy*. Figure 44 on page 540 is an example of a simple certificate hierarchy.

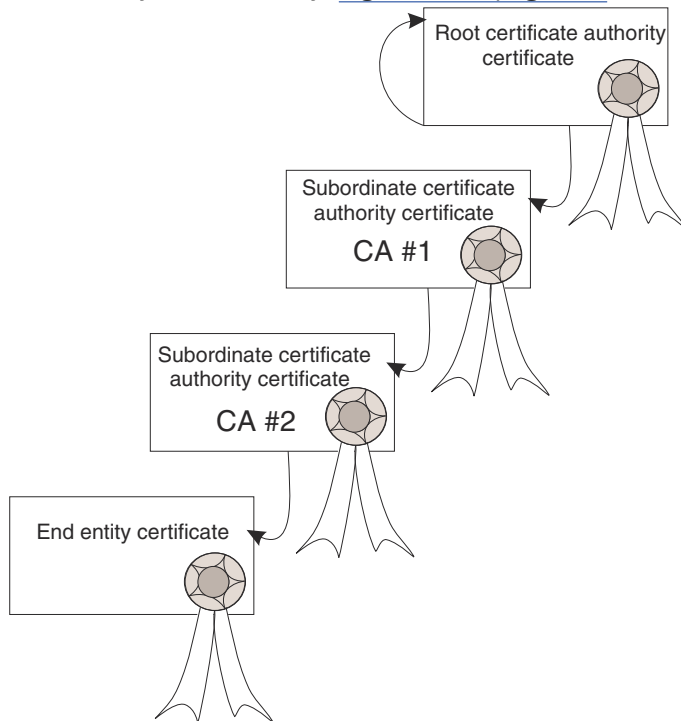


Figure 44. Example of a simple certificate hierarchy

Figure 44 on page 540 is a representation of a certificate hierarchy containing four entities where the end-entity certificate is issued by subordinate certificate authority (CA #2). The certificate of CA #2 is issued by subordinate certificate authority (CA #1). The certificate of CA #1 is issued by the root certificate authority. The certificate of the root certificate authority is self-issued, meaning that its certificate is signed by its own private key (a *self-signed* certificate).

The chain of signature verification begins with the end-entity certificate. The public key of CA #2 is used to verify the signature of the end-entity certificate. If the signature is valid, the public key of CA #1 is used to verify the signature of the CA #2 certificate. If the signature is valid, the public key of the root certificate is used to verify the signature of the CA #1 certificate. Finally, the signature of the root certificate is verified using its own public key.

Signature verification for the self-signed root certificate simply provides assurance that the root certificate is unaltered. It does not guarantee that the information in the certificate, or the certificate

authority itself, is trustworthy because anyone can create a self-signed certificate and claim to be a certificate authority. You must establish trust in your own selected set of certificate authorities and individual certificates *before* using public key protocols.

Your selected set of trusted certificates or CAs might be referred to using various terms, such as your *trusted roots*, *trusted signers*, or, simply, your *trust policy*. RACF supports your trust policies through RACF key rings. For details, see [“RACF and key rings” on page 567](#).

RACF, like other security software that supports digital certificates, has a method for supplying a predefined set of trusted root certificates. RACF includes a base set of well-known certificate authority certificates that are added to the RACF database whenever the system is IPLed. Unlike with other security software, these certificates are initially disabled. They are supplied as a convenience so that you need not define them yourself. However, you must determine which of these certificate authorities to trust, if any, and you must enable trust for those. For details, see [“Supplied digital certificates” on page 589](#).

**Remember:** You must select and enable trust for the certificate-authority certificates supplied with RACF before using them. Perform [“Steps to begin using a supplied CA certificate” on page 589](#).

## Public key algorithms

There are four asymmetric public-key algorithms in use today, with more likely to be developed in the future. The public key algorithms in use today are:

- Rivest-Shamir-Adleman (RSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Digital Signature Algorithm (DSA)
- Diffie-Hellman key agreement protocol

RSA, developed by RSA Laboratories, is by far the most popular algorithm and supports digital signatures and data encryption. Elliptic Curve Cryptography (ECC) is a newer algorithm that offers shorter keys that achieve comparable strengths when compared with longer RSA keys. DSA implements the Digital Signature Standard (DSS) published by the National Institute of Standards and Technology (NIST) and is used for digital signatures only. Diffie-Hellman keys cannot be stored in RACF but they can be retrieved from a PKCS #11 token.

## Certificate formats

X.509 digital certificates come in various formats for handling by system administrators and end users. It is important to understand these various formats because each is handled in a different way. RACF fully supports all of these forms, except as specified.

### Single binary certificate

In its base form, a digital certificate is a binary data structure containing the fields listed in [“X.509 certificates” on page 539](#). It is encoded using Distinguished Encoding Rules (DER), a platform-independent standard for encapsulating data. As with other binary data, remember to transfer a binary certificate in binary format, for example using binary FTP, when you move it to or from a z/OS system.

It is not necessary for you to examine the contents of a certificate. However, you can peek at them using a text editor because certificates do not contain private keys and are usually not encrypted in any way. If you peek at a data set containing a binary certificate on a z/OS or other EBCDIC platform, the contents appear unintelligible because none of the data is encoded in EBCDIC. On a Windows or other ASCII platform, some string data might be intelligible if it is encoded in ASCII.

### PKCS #7 binary certificate package

The PKCS #7 binary certificate package, based on the Public Key Cryptographic Standards (PKCS) published by RSA Laboratories, is a package used to distribute one or more certificates, or an entire chain of certificates such as the chain depicted in [Figure 44 on page 540](#). Most commercial certificate authorities return multiple requested certificates using the PKCS #7 format as a convenience rather than

distributing certificates individually. When used for distribution purposes, the PKCS #7 package as a whole is neither signed nor encrypted. As with the single binary certificate, the PKCS #7 package does not contain any private keys.

## PKCS #12 binary certificate package

The PKCS #12 binary certificate package is a password-encrypted package that can contain nearly any type of data. In its common form, the PKCS #12 package is similar to a PKCS #7 certificate chain with a private key included. In this form, it is the only form of PKCS #12 package that RACF supports. It can be used to migrate an end-entity certificate and its private key, with the signing certificate chain, from one platform to another. It can also be used by a certificate authority to distribute a certificate chain and one private key when the certificate authority generates the private key.

## Base64-encoded certificates

The binary certificate and the PKCS #7 and #12 binary packages can be additionally encoded using the base64 algorithm. Base64 encoding is a mechanism to convert binary data into text so that it can be easily transported as text, such as within an e-mail. When converting from binary to text, each three bytes of binary are converted into four characters from the following set: a–z, A–Z, 0–9, \, and +. When you peek at a base64-encoded certificate on any platform, it looks similar to the following:

```
-----BEGIN CERTIFICATE-----
MIICYzCCAcygAwIBAgIBADANBgkqhkiG9w0BAQUFADAUMQswCQYDVQQGEwJVUzEM
MAoGA1UEChMDSUJNMREwDwYDVQQLEwhMb2NhbcBDQTAeFw050TEyMjIwNTAwMDBa
Fw0wMDEyMjMwNDU5NTlaMC4xCzAJBgNVBAYTA1VMTQwwCgYDVQQKEwNJQk0xETAP
BgNVBAsTCExvY2FsIENBMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQD2bZEo
7xGaX2/0GHkzNFZv1xBou9v1Jmt/PDiTMPve8r9FeJAQ0QdvFST/0JPQYD20rH0b
imdDLgNdNynmyRoS2S/IIInfpmf69iyc2G0TPyRvmHI0ZbdCd+YBHQi1adkj17ND
cwj6S14tVurFX73zx0sNoMS79q3tuXKrDsxeuWIDAQABo4GQMIGNMEsGCVUdDwGG
+EIBDQQ+EzxHZW51cmF0ZWQgYnkGdGh1IFNlY3VyZVdheSBTZWN1cm10eSBTZXJ2
ZXIzZm9yIE9TLzM5MCAoUkFDRikwDgYDVVR0PAQH/BAQDAgAGMA8GA1UdEwEB/wQF
MAMBAf8wHQYDVVR0OBByEFJ3+ocRyCTJw067dLSw/na1x6YMA0GCSqGSIb3DQEB
BQUAA4GBAMaQzt+zaj1GU77yzl8iMBXgdQrwsZZWJo5exnAucJAEYQZm0fyLiM
D6oYq+ZnfM0n8G/Y79q8nhwvuxpY0nRSAXFp6xSkI0eZtJMY1h00LKp/JX3Ng1
svZ2agE126JHsQ0bhzN5TKsYfbwFTwfjdWAGy6Vf1nYi/r0+ryMO
-----END CERTIFICATE-----
```

**Remember:** Base64-encoded certificates are text. When you transfer them to or from a z/OS system, you must transfer them as text to ensure that the ASCII translation *to* or *from* EBCDIC takes place. Using ASCII (not binary) FTP or a text *cut-and-paste* will suffice. Be sure to include the BEGIN and END lines in your transfer.

## Using certificates with z/OS client/server applications

When you combine a driving application, such as z/OS HTTP Server with middleware that supports a secure protocol, such as SSL, and the secure certificate management functions of RACF, you can implement a secure certificate environment on z/OS. For implementation details about setting up z/OS HTTP Server and SSL, see the following references:

- WebSphere Application Server IBM Documentation ([www.ibm.com/docs/en/was](http://www.ibm.com/docs/en/was))
- *z/OS Cryptographic Services System SSL Programming*

## The secure handshake

A network protocol where a z/OS application is playing the role of the client or the server is shown in Figure 45 on page 543. Each party, both client and server, has its own certificate, a matching private key, and a list of trusted certificate-authority certificates. When the client needs to authenticate itself to the server to be able to perform a transaction, both the server and client need to verify one another. The protocol for a secure handshake for mutual verification begins with the parties exchanging certificates. Each party then separately validates the other's certificate to make sure that its signature is valid, that the subject name in the certificate is correct, and that the certificate originated from a trusted certificate authority. If successful, each party must prove to the other that it owns the private key that matches its public key certificate. This step establishes *proof of possession* and can be accomplished by having each

party sign a known unique value, such as a hash of the message traffic between the two parties. If each signature can be validated using the associated public key, the proofs are successful. The final step in this handshake is for one of the parties to generate a random symmetric key, encrypt it using the other party's public key, and send it to the other party. This random symmetric key can then be used to encrypt the data for the remainder of the session. Once the secure handshake is complete, secure transactions can be safely handled in the z/OS environment between this client and server.

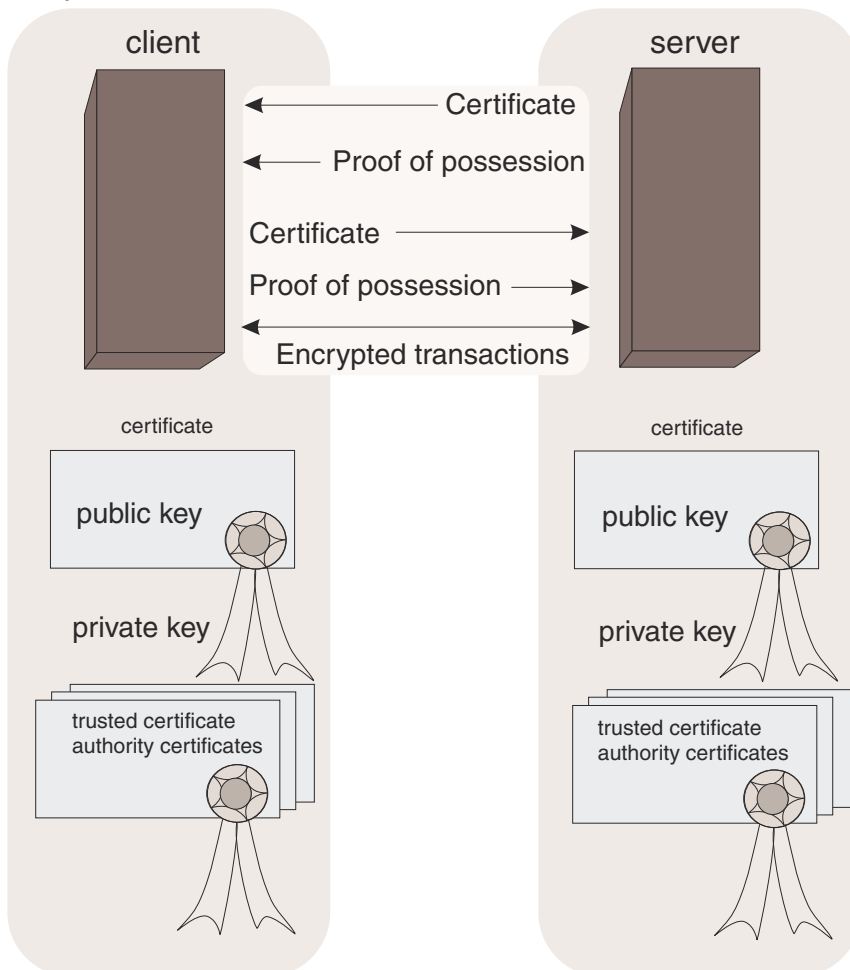


Figure 45. A high-level view of a secure z/OS handshake using a public key network protocol

## Planning your certificate environment

The following questions and answers represent decisions on which you will base your z/OS certificate environment.

### What is the name of my network entity?

When creating a certificate, the subject name in the certificate is the name of the entity it represents. The subject's name is an X.500 distinguished name that consists of various qualifier-value pairs. For example, if you are creating a certificate for a Web server with the domain name `systems.abc.com` within the Systems division of a company in the United States, the distinguished name might be as follows: `CN=systems.abc.com, OU=Systems Division, O=ABC, C=US`.

### What type of encryption will I choose for my public and private key pair?

RSA is the default. Choose DSA only when you have a specific need for DSA.

### Which user ID will be the owner of my certificate?

The user ID of the daemon or started task associated with your application will be the owner of the certificate.

### What nickname or label will this certificate to be known by?

A certificate label can be up to 32 characters in length and must be unique to each owning user ID.



### How big will my private key be?

The larger the key, the greater its strength and the slower it operates. Be aware that some larger keys might be export-controlled.

### Where will I generate and store my private key?

RACF gives you the ability to generate keys in hardware or software. Keys generated by Integrated Cryptographic Service Facility (ICSF) hardware are more secure but require that the hardware is always present and enabled. (RACF cannot back up ICSF keys.) By contrast, keys generated and stored as software keys are least secure. This is because they are stored in the RACF database in a *masked* format that is less secure than being encrypted by ICSF. The best compromise is to generate your keys in software, save them to an encrypted PKCS #12 data set, and then store them in the ICSF PKA key data set (PKDS).

### Which organization will issue my certificate? Who will be my certificate authority?

If you intend to operate a commercial Web application, choose a well-known commercial certificate authority. If you have no need for a well-known certificate authority and need only a small number of certificates, choose RACF as a small-scope certificate authority. If you need to issue a large number of certificates, consider using commercial software, such as z/OS Cryptographic Services PKI Services. (To find out more about PKI Services, see [z/OS Cryptographic Services PKI Services Guide and Reference](#).)

### Will I specify a certificate validity period?

If you plan to create a certificate that you will not replace with an externally signed one, then you should specify the validity period. RACF's default validity period is one year from the date of issue but this period is usually not long enough for a CA certificate. If you request a certificate from an external certificate authority, you need not specify the validity period. The external certificate authority sets the validity period.

### Which certificate authorities will I trust?

You need to decide which certificate authorities you will consider trusted enough to identify parties involved in the network protocol with your application. You need to trust at least one certificate authority, the one that issued the certificate for your application. Any others you choose are based on the needs of your application and its users. The set of certificate authorities that your application considers trusted comprise your application's trust policy, or RACF key ring.

## Setting up your certificate environment

After you complete your planning decisions, you can begin setting up your z/OS certificate environment. In general, this is the sequence of activity involved in preparing for one entity, or application, to use a secure network protocol. All of the following activities are described using RACF command functions where applicable. (For RACDCERT command syntax, see [z/OS Security Server RACF Command Language Reference](#).) At your option, you can choose to supplement your activities with support from other software or external organizations.

1. If you chose RACF as your certificate authority, use the RACDCERT GENCERT command to generate your certificate authority (CA) certificate, the associated public, and the private key pair in RACF. Set the certificate's validity period because the one-year default value is usually not long enough for a CA certificate.
2. Use the RACDCERT GENCERT command to generate your application's end-entity certificate, and the associated public and private key pair, in RACF.
  - If you are using RACF as your certificate authority, sign the application certificate with your RACF certificate authority certificate.
  - If you are using an external certificate authority, create a self-signed certificate in RACF as a *placeholder* and use the RACDCERT GENREQ command to generate a certificate request, based on the placeholder certificate, to send to your external certificate authority. The certificate request (header line, footer line, and all data between them) is sent to the certificate authority who signs the certificate and returns it to you. Upon receipt, use RACDCERT ADD to replace the self-signed certificate with your new CA-signed certificate.



If you peek at a request data set before you send it to the certificate authority, you will notice the following header and footer lines. (Certificate requests are always DER-encoded and then base64-encoded, like base64-encoded certificates.)

```
-----BEGIN NEW CERTIFICATE REQUEST-----
:
-----END NEW CERTIFICATE REQUEST-----
```

3. Establish the trust policy for your application. (For details, see [“RACF and key rings”](#) on page 567.)
  - Use the RACDCERT ADDRING command to define a key ring in RACF and associate it with your application's user ID.
  - Use the RACDCERT CONNECT command to connect certificates to the key ring. Be sure to connect your trusted certificate authority certificates and the certificate that represents your application.

## Enabling client login using certificates

Without digital certificates, RACF users of traditional client/server applications authenticate themselves to servers by presenting their user IDs and passwords. Successful authentication adds a security *context*, a control block called the accessor environment element (ACEE), to the user's address space. Subsequently, units of work initiated by the client are tagged with the client's identity, or security context. In this environment, the client's user ID and password provide identification and authentication.

In the z/OS digital certificate environment, the secure handshake protocol depicted in Figure 45 on page 543 accomplishes identification and authentication when the client presents its certificate as identification and its *proof-of-possession* as authentication. The client's ACEE is created when the application invokes the SAF callable service called `initACEE` (IRRSIA00) to determine the client's user ID based on information in the client's certificate.

## Certificate mapping

When RACF is called by `initACEE` to determine the user ID to associate with the client certificate, it does so based on your installation's set of certificate mapping rules. RACF can only determine the user ID for a given certificate if a rule covering that certificate was created prior to the certificate's use.

RACF provides the following mechanisms for defining certificate mapping rules:

- One-to-one certificate to user ID association
- Certificate name filtering
- The `hostIdMappings` certificate extension

### ***One-to-one certificate to user ID association***

Whenever you generate a certificate using the RACDCERT GENCERT command, RACF registers it to a user ID and adds it to the RACF database. You can also store a previously generated certificate and register it to a user ID using the RACDCERT ADD command. These methods establish a direct one-to-one association, or mapping, between each certificate and one specific user ID. You can create direct mappings for each of your users by simply adding individual certificates for each user to the RACF database. However, the administrative cost of this approach might only be feasible for you when handling a limited number of certificates.

Registered certificates are stored in certificate profiles. These profiles contain an exact copy of the certificate and, for user IDs on this system, the private key, if it exists. Certificates stored in this way can be used to simply associate a certificate with a user ID or they can be gathered into a collection, or *key ring*, for use by other applications as part of a secure network protocol. For details, see [“Using the RACDCERT command to administer certificates”](#) on page 548 and [“RACF and key rings”](#) on page 567.

### ***Certificate name filtering***

For some applications, directly mapping each client certificate to a user ID is neither practical nor desirable. An alternative is to create one or more certificate name filters using the RACDCERT MAP

command. A certificate name filter allows you to associate many certificates with one user ID, based on rules concerning portions of the subject's or issuer's distinguished names in the certificate, such as the subject's corporate affiliation or department. With carefully chosen certificate name filters, a large number of client certificates can be mapped to a limited number of user IDs with very little administrative cost.

This benefit is limited to some degree by a loss of granularity in access control. For example, if you create a certificate name filter to map the certificates of all company employees in the Systems division to user ID SDUSER, then all such employees are given the resource authorizations of the user ID SDUSER. However, you retain full auditing accountability because the subject's and issuer's distinguished names in the client's certificate appears in every audit record created on behalf of the client's unit of work.

This mapping option is explored in detail in [“Certificate name filtering” on page 571](#).

### ***The hostIdMappings certificate extensions***

The hostIdMappings certificate extension is used to communicate the user's host identity for one or more host systems. The extension contains a sequence of host name and user ID value pairs. (Each pair can also have an encrypted password, but this field is not used by RACF.) When RACF is called to create an ACEE from a certificate containing a hostIdMappings extension, RACF examines the extension to determine the appropriate user ID for the ACEE. For more information how RACF uses this extension, see [“Using a hostIdMappings extension” on page 605](#).

When you use hostIdMappings extensions, you need not create certificate profiles nor name filters prior to using certificates. However, as with all other extensions in a certificate, the hostIdMappings extension is created by the certificate's issuer at the time the certificate is generated. If you operate as your own certificate authority and you know the respective user IDs of your clients at the time their certificates are created, using hostIdMappings extensions is your lowest administrative cost option.

**Restriction:** PKI Services for z/OS supports the creation of hostIdMappings extensions. However, other commercial certificate-authority software might not support them, so check with your software vendor.

## Using RACF to manage digital certificates

---

You can use RACF to create, register, store, and administer digital certificates and their associated private keys, and build certificate requests that can be sent to a certificate authority for signing. You can also use RACF to manage key rings of stored digital certificates. Digital certificates and key rings are managed in RACF primarily by using the RACDCERT command or by using an application that invokes the R\_data lib callable service (IRRSDL00 or IRRSDL64) or the initACEE callable service (IRRSIA00).

The R\_data lib callable service provides an application programming interface to the CDSA (Common Data Security Architecture) data library functions, and is used by secure sockets layer (SSL) and System SSL to establish secure sessions between servers. The initACEE callable service can be used to manage digital certificates for RACF-authenticated users.

RACF has three categories for managing digital certificates:

### **User certificate**

A certificate that is associated with a RACF user ID and is used to authenticate the user's identity. The RACF user ID can represent a traditional user or be assigned to a server or started procedure.

### **Certificate-authority certificate**

A certificate that is associated with a certificate authority and is used to verify signatures in other certificates.

### **Site certificate**

A certificate that is associated with an off-platform server or other network entity, such as a peer VPN server. This category of certificate can also be used to share a single certificate and its private key among multiple RACF user IDs. When used for sharing, a certificate might be referred to as a *placeholder* certificate.

**Note:** You can use the RDATA LIB class to allow other IDs to access the private key belonging to another ID without using Site certificate.

## Size considerations for public and private keys

RACF has restrictions for the size of the private key for certificates that have associated private keys.

For NISTECC keys, valid key sizes are 192, 224, 256, 384, and 521 bits. For BPECC keys, valid key sizes are 160, 192, 224, 256, 320, 384, and 512 bits.

For DSA keys, the minimum key size is 512. For RSA keys, the minimum size for clear RSA keys and secure RSA keys on the public key data set (PKDS) is 512 bits. The minimum size for secure RSA keys on the token key data set (TKDS) is 1024 bits and the size must be a multiple of 256. The maximum key size is determined by United States export regulations and is controlled by RACF and non-RACF code in z/OS. Depending on the installation, non-RACF code might enforce a lower maximum size.

**Maximum key sizes:** The maximum key size for a private key depends on key type, as follows:

Private key type	Maximum key size
RSA key that is stored in the RACF database	4096 bits
RSA key that is stored in the ICSF TKDS as a secure key	4096 bits
RSA key that is stored in the ICSF PKDS as a CRT key token	4096 bits
DSA key	2048 bits
RSA key that is stored in the ICSF PKDS as an ME key token	1024 bits
NISTECC key	521 bits
BPECC key	512 bits

Currently, the standard sizes for RSA keys are as follows:

Key size	Key strength
512 bits	Low-strength key
1024 bits	Medium-strength key
2048 bits	High-strength key
4096 bits	Very high-strength key

**Key strength considerations:** Shorter keys of the ECC type, which are generated when you specify NISTECC or BPECC, achieve comparable key strengths when compared with longer RSA keys.

RSA, NISTECC, and BPECC keys of the following sizes are comparable in strength:

RSA key size	NISTECC key size	BPECC key size
1024 bits	192 bits	160 or 192 bits
2048 bits	224 bits	224 bits
3072 bits	256 bits	256 or 320 bits
7680 bits	384 bits	384 bits
15360 bits	521 bits	512 bits

**Hashing algorithm used for signing:** RACF signs certificates using a set of secure hash algorithms that are based on the SHA-1 or SHA-2 hash functions. When the signing key is a DSA type, the SHA-1 algorithm is used for keys of all sizes. When the signing key is an RSA, NISTECC, or BPECC type, the size of the signing key determines the hashing algorithm that is used for signing, as follows:

Hashing algorithm used for signing	Signing key size		
	RSA	NISTECC	BPECC
SHA-1	Less than 2048 bits	—	—
SHA-256	2048 bits or longer	192, 224, or 256 bits	160, 192, 224, 256, or 320 bits
SHA-384	—	384 bits	384 bits
SHA-512	—	521 bits	512 bits

## Using the RACDCERT command to administer certificates

The RACDCERT command is used to store and maintain digital certificate information in RACF, and should be used for all maintenance of certificate profiles and related user profile fields. For more information on these formats, see [z/OS Security Server RACF Command Language Reference](#).

The RACDCERT command can be used to perform the following functions:

- List information about the certificates for a specified RACF-defined user ID, or your own user ID.
- Add a certificate and associate it with a specified RACF-defined user ID, or your own user ID, and set the TRUST status.
- Alter the TRUST status or label for a certificate.
- Delete a certificate.
- Add or remove a certificate from a key ring.
- Create, delete, or list a key ring.
- Generate a public/private key pair and certificate, replicate a digital certificate with a new public/private key pair, or retire the use of an existing private key.
- Write (export) a certificate or certificate package to a data set.
- Create a certificate request.
- Create, alter, delete, or list a certificate name filter (user ID mapping).
- Add, delete, or list a z/OS PKCS #11 token.
- Bind a certificate to a z/OS PKCS #11 token.
- Remove (unbind) a certificate from a z/OS PKCS #11 token.
- Import a certificate (with its private key, if present) from a z/OS PKCS #11 token and add it to RACF.
- List the content of a certificate and its issuers' certificates and list information that can cause the certificate to be unusable.
- Check whether a data set contains a valid chain of certificates and whether they have been installed in RACF.

The RACDCERT command is your primary administrative tool for managing digital certificates using RACF. Authority to use the RACDCERT command is controlled through resources in the FACILITY class or the RDATA LIB class. The RACDCERT command is used to manage resources in the following classes:

### DIGTCERT

Profiles in the DIGTCERT class contain information about digital certificates, as well as the certificate itself and the private key, if any. For more information, see [“DIGTCERT general resource profiles” on page 565](#).

### DIGTRING

Profiles in the DIGTRING class contain information about key rings and the certificates that are part of each key ring. Key rings are named collections of the personal, site and certificate-authority certificates associated with a specific user. For more information, see [“DIGTRING general resource profiles” on page 568](#).

**DIGTNMAP**

Profiles in the DIGTNMAP class contain information about certificate name filters. For more information, see [“DIGTNMAP general resource profiles” on page 573](#).

**USER**

Profiles in the USER class contain information about digital certificates that are associated with the user.

This information is used by the RACDCERT command in its processing and by the DELUSER command to clean up certificate-related resources owned by the user ID being deleted.

**Restriction:** Profiles in the DIGTCERT, DIGTRING, and DIGTNMAP classes are automatically maintained through RACDCERT command processing. You cannot administer profiles in these classes using the RDEFINE, RALTER, and RDELETE commands. These commands do not operate with profiles in the DIGTCERT, DIGTRING, and DIGTNMAP classes. Because these profiles contain lowercase characters, the SEARCH FILTER and RLIST commands are not intended for use and will deliver unpredictable results.

You need not activate the DIGTCERT, DIGTCRIT, and DIGTRING classes to use resources in those classes. However, performance is improved when you activate and RACLIST the DIGTCERT and DIGTCRIT classes. See [“RACLISTing the DIGTCERT class” on page 566](#) and [“RACLISTing the DIGTCRIT class” on page 579](#).

See [z/OS Security Server RACF Command Language Reference](#) for more information about the RACDCERT command.

See [“RRSF considerations for digital certificates” on page 431](#) for information about propagating updates made by the RACDCERT command to other nodes in an RRSF network.

## Controlling the use of the RACDCERT command

Authority to the IRR.DIGTCERT.*function* resource in the FACILITY class allows a user to issue the RACDCERT command. To issue the RACDCERT command, users must have one of the following authorities:

- The SPECIAL attribute
- Sufficient authority to resource IRR.DIGTCERT.*function* in the FACILITY class.
  - READ access to IRR.DIGTCERT.*function* to issue the RACDCERT command for themselves.
  - UPDATE access to IRR.DIGTCERT.*function* to issue the RACDCERT command for others.
  - CONTROL access to IRR.DIGTCERT.*function* to issue the RACDCERT command for SITE and CERTAUTH certificates. (This authority also has other uses.)
  - CONTROL access to IRR.DIGTCERT.LIST to issue the RACDCERT command with the LISTCHAIN keyword.
- Sufficient authority to the following resources in the RDATA LIB class, if IRR.RACDCERT.GRANULAR is defined.

**Note:** Do not define IRR.RACDCERT.GRANULAR before granular profiles are set up. Otherwise the following RACDCERT functions will fail the authorization check.

- For functions that involve certificates: READ authority to IRR.DIGTCERT.cert\_owner.cert\_label.UPD.function or IRR.DIGTCERT.cert\_owner.cert\_label.LST.function, where function is ADD , ALTER , DELETE , EXPORT , GENCERT , GENREQ , IMPORT , REKEY or ROLLOVER.
- For functions that involve key ring: READ authority to ring\_owner.ring\_name.UPD.function, where function is ADDRING or DELRING.
- For functions that involve key ring and certificate: READ authority to IRR.DIGTCERT.cert\_owner.cert\_label.LST.function, and ring\_owner.ring\_name.UPD.function, where function is CONNECT or REMOVE.

For detailed information about the RACDCERT command and the authority required to execute each command, see [z/OS Security Server RACF Command Language Reference](#).

Note that users who have insufficient authority to the IRR.DIGTCERT.LIST resource can use the RACDCERT CHECKCERT command to display some digital certificate information if they have READ authority to the data set containing the certificate. The output they receive reflects only the certificate information contained in the data set. Because they lack sufficient authority to the IRR.DIGTCERT.LIST resource, they will not receive certificate information contained in the RACF database, such as the TRUST status, the LABEL, or the RACF user ID associated with the certificate. For an example of this output, see [“Examples of checking digital certificate information” on page 559](#).

Unlike the other RACDCERT functions, there is only one access level for LISTCHAIN, which is CONTROL. Only users who have CONTROL authority to the IRR.DIGTCERT.LIST resource can use the RACDCERT LISTCHAIN command to display information about the certificates in the chain.

### **Examples of controlling the use of the RACDCERT command using the FACILITY class**

Effective use of RACDCERT requires that its privileges be carefully controlled. However, end-users and application administrators should be allowed some flexibility in defining their security characteristics. These guidelines might prove useful.

- The ability to add certificate authorities and site certificates should be allowed to only a small set of trusted people.
- End users should be permitted to add, delete, and modify the contents of their own key rings and add, delete, and alter their own certificates.
- Help desk personnel should be allowed the ability to list certificates and rings.

Assume that your system administrators, who are the only ones who are allowed to add, alter, or delete certificate-authority certificates or site certificates, are in the group WEBADMIN. Furthermore, assume that your help desk personnel are in the group HELPDESK. The commands in [Figure 46 on page 551](#) show one method of controlling access to RACDCERT functions. Note that similar authorizations can be defined to allow system administrators and help desk personnel to manage certificate name filters.

```

RDEFINE FACILITY IRR.DIGTCERT.ADD          UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ADDRING      UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ALTER        UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ALTMAP       UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.BIND         UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.CONNECT      UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.DELETE       UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.DELMAP      UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.DELRING     UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.EXPORT       UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.EXPORTKEY   UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENCERT     UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.GENREQ      UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LIST        UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTCHAIN    UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTMAP     UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.LISTRING    UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.MAP         UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.REKEY       UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.REMOVE      UACC(NONE)
RDEFINE FACILITY IRR.DIGTCERT.ROLLOVER    UACC(NONE)

PERMIT IRR.DIGTCERT.ADD          CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ADDRING      CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.ALTER        CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ALTMAP       CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.BIND         CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.CONNECT      CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.DELETE       CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.DELMAP      CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.DELRING     CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.EXPORT       CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.EXPORTKEY   CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.GENCERT     CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.GENREQ      CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LIST        CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTCHAIN    CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTMAP     CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.LISTRING    CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.MAP         CLASS(FACILITY) ID(WEBADMIN) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.REKEY       CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.REMOVE      CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.ROLLOVER    CLASS(FACILITY) ID(WEBADMIN) ACCESS(CONTROL)

PERMIT IRR.DIGTCERT.ADD          CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.ADDRING      CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.ALTER        CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.ALTMAP       CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.BIND         CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.CONNECT      CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.DELETE       CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.DELMAP      CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.DELRING     CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.EXPORT       CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.EXPORTKEY   CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.GENCERT     CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.GENREQ      CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.LIST        CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTMAP     CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTRING    CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.MAP         CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.REKEY       CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.REMOVE      CLASS(FACILITY) ID(*) ACCESS(READ)
PERMIT IRR.DIGTCERT.ROLLOVER    CLASS(FACILITY) ID(*) ACCESS(READ)

PERMIT IRR.DIGTCERT.LIST        CLASS(FACILITY) ID(HELPPDESK) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTCHAIN    CLASS(FACILITY) ID(HELPPDESK) ACCESS(CONTROL)
PERMIT IRR.DIGTCERT.LISTMAP     CLASS(FACILITY) ID(HELPPDESK) ACCESS(UPDATE)
PERMIT IRR.DIGTCERT.LISTRING    CLASS(FACILITY) ID(HELPPDESK) ACCESS(UPDATE)

```

Figure 46. Controlling access to RACDCERT functions under the FACILITY class

## Examples of controlling the use of the RACDCERT command using the RDATA LIB class

By using the granular control (enabled by defining the profile IRR.RACDCERT.GRANULAR in the RDATA LIB class), you can enforce a naming convention for the certificates and the key rings in your system and segregate the administration of them. For example:

- To enforce the rule that the label for a certificate used for tcpip must start with the string TCPIP you can use:

```
RDEFINE RDATA LIB IRR.DIGTCERT.*.TCPIP*.UPD.GENCERT UACC(NONE) for all certificate owners with all certificates that start with the string TCPIP
```

or

```
RDEFINE RDATA LIB IRR.DIGTCERT.certificate_owner.TCPIP_SYS1.UPD.GENCERT UACC(NONE) for a specific certificate owner with all certificate name TCPIP_SYS1.
```

- To enforce the rule that the name for a key ring used for servers must start with the string SERVER you can use:

```
RDEFINE RDATA LIB *.SERVER*.UPD.ADDRING UACC(NONE) for all ring owners with all ring names that start with the string SERVER
```

or

```
RDEFINE RDATA LIB ring_owner.SERVERABC.UPD.ADDRING UACC(NONE) for a specific ring owner with all ring names that start with the string SERVER.
```

- To allow system administrators to create certificates with labels that start with TCPIP and create key rings with names that start with SERVER:

```
PERMIT IRR.DIGTCERT.*.TCPIP*.*.GENCERT CLASS(RDATA LIB) ID(SYSADMIN) ACCESS(READ)
```

```
PERMIT *.SERVER*.UPD.ADDRING CLASS(RDATA LIB) ID(SYSADMIN) ACCESS(READ)
```

- To allow web server administrators to connect the TCPIP\_TEST certificate to the SERVERABC key ring:

```
PERMIT IRR.DIGTCERT.*.TCPIP*.*.CONNECT CLASS(RDATA LIB) ID(WEBADMIN) ACCESS(READ)
```

```
PERMIT *.SERVER*.UPD.CONNECT CLASS(RDATA LIB) ID(WEBADMIN) ACCESS(READ)
```

- To enforce the CA certificate of PKI Services (label LOCAL\_PKI\_CA) can only be used by that PKI daemon PKISRVD, but not by any administrators to sign other certificates:

```
RDEFINE RDATA LIB IRR.DIGTCERT.CERTIFAUTH.LOCAL_PKI_CA.UPD.GENCERT UACC(NONE)
```

```
PERMIT IRR.DIGTCERT.CERTIFAUTH.LOCAL_PKI_CA.UPD.GENCERT CLASS(RDATA LIB) ID(PKISRVD) ACCESS(READ)
```

## Examples of adding digital certificate information

1. User RACFADM with SPECIAL authority requests the addition of a digital certificate for user NET2. RACFADM has placed the digital certificate in data set 'RACFADM.NET2.CERT' and wants the status of the certificate to be trusted. RACFADM issues the following RACDCERT command:

```
RACDCERT ID(NET2) ADD('RACFADM.NET2.CERT') TRUST
```

2. User RACFADM with SPECIAL authority requests the addition of a digital certificate for user NETBOY. RACFADM has placed the digital certificate in data set 'RACFADM.NETBOY.CERT' and wants the status of the certificate to be trusted. In addition, RACFADM wants to associate the saved certificate for user NETBOY with a label called 'Savings Account'. RACFADM issues the following RACDCERT command:

```
RACDCERT ID(NETBOY) ADD('RACFADM.NETBOY.CERT') TRUST WITHLABEL('Savings Account')
```

## Examples of listing digital certificate information

1. User RACFADM with SPECIAL authority requests the listing of user ID GEORGEM's digital certificate information by issuing the RACDCERT command with the LIST operand. User ID GEORGEM has three certificates, one of which is not associated with any key rings. [Figure 47 on page 553](#) shows the output of the following command:



**Note:** Since the Certificate Fingerprint is unique to each certificate, you may use it to verify that it is the same certificate.

#### RACDCERT ID(GEORGEM) LIST

```

Digital certificate information for user GEORGEM:

Label: New Cert Type - Ser # 00
Certificate ID: 2QfHxdbZx8XU1YWmQMOfmaNA46iXhUBgQOKFmUB7QPDw
Status: TRUST
Start Date: 2010/04/18 03:01:13
End Date: 2025/02/13 03:01:13
Serial Number:
>00<
Issuer's Name:
>OU=Internet Demo CertAuth.O=The Cert Software Inc.<
Subject's Name:
>OU=Internet Demo CertAuth.O=The Cert Software Inc.<
Signing Algorithm: sha256RSA
Key Type: RSA Mod-Exp
Key Size: 2048
Private Key: YES
PKDS Label: IRR.DIGTCERT.GEORGEM.SY1.BD7103108611F42F
Certificate Fingerprint (SHA256):
  C2:B4:5C:1C:E4:71:DF:D3:32:F3:08:9B:85:42:E9:46:
  17:D8:93:D7:BE:84:4E:11:A7:93:7E:E5:9D:A6:43:14
Ring Associations:
Ring Owner: GEORGEM
Ring:
>GEORGEMsNewRing01<
Ring Owner: GEORGEM
Ring:
>GEORGEMsRing<

Label: New Type Cert - VsignC1
Certificate ID: 2QfHxdbZx8XU1YWmQ00o14VAw4WZo0BgQ0WiiYeVw/FA
Status: TRUST
Start Date: 2010/04/22 23:23:26
End Date: 2028/01/15 23:23:26
Serial Number:
>3511A552906FE7D029A44019D411FC3E<
Issuer's Name:
>OU=Class 1 Public Primary Certification Authority.O=VeriSign, Inc..C=<
>US<
Subject's Name:
>CN=Gwillink.OU=RedoakCA.L=Clymer.SP=NY.C=US<
Signing Algorithm: sha256RSA
Key Type: RSA
Key Size: 512
Private Key: YES
Certificate Fingerprint (SHA256):
  BA:F6:4A:2C:C4:91:DF:D3:32:F3:08:9B:65:42:E9:36:
  17:D8:93:C7:BE:94:4E:10:A7:93:7E:E2:8D:D6:51:A2
Ring Associations:
Ring Owner: GEORGEM
Ring:
>GEORGEMsNewRing01<

Label: New Type Cert - VsignC2
Certificate ID: 2QfHxdbZx8XU1YWmQ00o14VAw4WZo0BgQ0WiiYeVw/JA
Status: NOTRUST
Start Date: 2010/03/19 15:39:52
End Date: 2030/03/19 15:39:52
Serial Number:
>50D35294912F79D315E32B31AC8548F0<
Issuer's Name:
>OU=Class 2 Public Primary Certification Authority.O=VeriSign, Inc..C=<
>US<
Subject's Name:
>CN=Steve Rater.OU=StorageCA.L=Corry.SP=PA.C=US<
Signing Algorithm: sha256RSA
Key Type: NIST ECC
Key Size: 256
Private Key: NO
Certificate Fingerprint (SHA256):
  AA:B6:5A:1C:C4:91:DF:D3:32:F3:08:9B:85:42:E9:46:
  17:D8:93:D7:BE:94:4E:11:A7:93:7E:E2:9D:B6:23:16
Ring Associations:
*** No rings associated ***

```

Figure 47. Output from the RACDCERT LIST command

2. User RACFADM with SPECIAL authority requests the listing of user ID GEORGEM's key rings by issuing the RACDCERT command with the LISTRING operand. User ID GEORGEM has three key rings with

certificates and one key ring which has no certificates. [Figure 48 on page 554](#) shows the output of the following command:

```
RACDCERT ID(GEORGEM) LISTRING
```

Digital ring information for user GEORGEM:

```
Ring:
>GEORGEMsNewRing01<
Certificate Label Name      Cert Owner      USAGE      DEFAULT
-----
New Cert Type - Ser # 00    ID(GEORGEM)     PERSONAL    YES
New Type Cert - VsignC1     ID(GEORGEM)     CERTAUTH    NO
New Type Cert - VsignC2     ID(GEORGEM)     SITE        NO
65                           ID(JOHNHP)      PERSONAL    NO

Ring:
>GEORGEMsRing<
Certificate Label Name      Cert Owner      USAGE      DEFAULT
-----
GEORGEM's Cert # 48         ID(GEORGEM)     PERSONAL    NO
GEORGEM's Cert # 84         ID(GEORGEM)     PERSONAL    NO
New Cert Type - Ser # 00    ID(GEORGEM)     PERSONAL    YES

Ring:
>GEORGEMsRing#2<
Certificate Label Name      Cert Owner      USAGE      DEFAULT
-----
GEORGEM's Cert # 84         ID(GEORGEM)     PERSONAL    NO
GEORGEM's Cert # 48         ID(GEORGEM)     PERSONAL    NO

Ring:
>GEORGEMsRing#3<
*** No certificates connected ***
```

*Figure 48. Output from the RACDCERT LISTRING command*

3. User NETBOY requests the listing of his Savings Account digital certificate to ensure it has been defined, and that it is marked trusted. He has READ authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the RACDCERT command with the LIST operand, specifying the label to identify his certificate. [Figure 49 on page 554](#) shows the output of the following command:

```
RACDCERT LIST(LABEL('Savings Account'))
```

```
Digital certificate information for user NETBOY:
Label: Savings Account
Certificate ID: 2QbVxePC1ujigaWJlYeiQMGDg5aklaNA
Status: TRUST
Start Date: 2010/11/10 00:00:00
End Date: 2026/11/10 23:59:59
Serial Number:
>5D666C20207A6638727A413872D8413B<
Issuer's Name:
>OU=BobsBank Savers.O=BobsBank.L=Internet<
Subject's Name:
>CN=S.S.Smith.OU=Digital ID Class 1 - NetScape.OU=BobsBank Class 1 - S<
>avingsAcct.O=BobsBank.L=Internet<
Signing Algorithm: sha256RSA
Key Type: Brainpool ECC
Key Size: 192
Private Key: YES
Certificate Fingerprint (SHA256):
D2:E5:3C:24:C4:91:DF:D3:32:F3:08:9B:85:12:E9:46:
11:D8:93:D7:BE:94:4E:10:A7:93:7E:E1:9C:D9:65:17
Ring Associations:
*** No rings associated ***
```

*Figure 49. Output from the RACDCERT LIST command with LABEL*

4. User RACFADM with SPECIAL authority uses the RLIST DIGTCERT \* command to request the listing of all DIGTCERT profiles. This RLIST command lists information about the profiles that contain digital certificates, rather than information about the certificates themselves. (Use the RACDCERT LIST

command to list detailed information about certificates.) [Figure 50 on page 555](#) shows a partial sample of the output of the following command:

```
RLIST DIGTCERT *
```

The RLIST command lists the universal access value for a profile in the DIGTCERT class differently based on the TRUST status of the digital certificate contained in the profile:

Trust status	Universal access
Trusted	ALTER
Untrusted	???????

[Figure 50 on page 555](#) shows the listing of a profile containing a certificate-authority certificate that was supplied with your RACF system. For more information about these certificates, see [“Supplied digital certificates” on page 589](#).

```
RLIST DIGTCERT *
CLASS      NAME
-----
DIGTCERT   00.personal-basic@thawte.com.CN=ThawtePersonalBasicCA.OU=Certific
ationServicesDivision.O=ThawteConsulting.L=CapeTown.SP=WesternCape.C=ZA

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----
  00    IBMUSER      ???????              NONE         NO

INSTALLATION DATA
-----
NONE

APPLICATION DATA
-----
irrcerta

AUDITING
-----
FAILURES(READ)

NOTIFY
-----
NO USER TO BE NOTIFIED

:
```

*Figure 50. Output from the RLIST DIGTCERT command*

5. User RACFADM with SPECIAL authority uses the SEARCH CLASS(DIGTCERT) command to find the names of all DIGTCERT profiles. (For detailed listings of certificate information, use the RACDCERT LIST command.) [Figure 51 on page 556](#) shows sample output from the following command:

```
SEARCH CLASS(DIGTCERT)
```

[Figure 51 on page 556](#) shows several listings of profiles containing certificate-authority certificates that are supplied with your RACF system. For more information, see [“Supplied digital certificates” on page 589](#).

```
SEARCH CLASS(DIGTCERT)

00.personal-basic@thawte.com.CN=ThawtePersonalBasicCA.OU=CertificationServicesDivision.O=ThawteConsulting.L=CapeTown.SP=WesternCape.C=ZA

00.personal-freemail@thawte.com.CN=ThawtePersonalFreemailCA.OU=CertificationServicesDivision.O=ThawteConsulting.L=CapeTown.SP=WesternCape.C=ZA

00.personal-premium@thawte.com.CN=ThawtePersonalPremiumCA.OU=CertificationServicesDivision.O=ThawteConsulting.L=CapeTown.SP=WesternCape.C=ZA

00BA5AC94C053B92D6A7B6DF4ED053920D.OU=Class2PublicPrimaryCertificationAuthority.O=VeriSign,Inc..C=US

00E49EFD33AE80ECFA5113E19A4240232.OU=Class3PublicPrimaryCertificationAuthority.O=VeriSign,Inc..C=US

01.premium-server@thawte.com.CN=ThawtePremiumServerCA.OU=CertificationServicesDivision.O=ThawteConsultingcc.L=CapeTown.SP=WesternCape.C=ZA

01.server-certs@thawte.com.CN=ThawteServerCA.OU=CertificationServicesDivision.O=ThawteConsultingcc.L=CapeTown.SP=WesternCape.C=ZA

02AD667E4E45FE5E576F3C98195EDDC0.OU=SecureServerCertificationAuthority.O=RSADataSecurity,Inc..C=US

325033CF50D156F35C81AD655C4FC825.OU=Class1PublicPrimaryCertificationAuthority.O=VeriSign,Inc..C=US

3381F595.CN=IntegrionCertificationAuthorityRoot.O=IntegrionFinancialNetwork.C=US

33820AD2.CN=IBMWorldRegistryCertificationAuthority.O=IBMWorldRegistry.C=US

:
```

Figure 51. Output from the SEARCH CLASS(DIGTCERT) command

## Examples of listing digital certificate chain information

Use the LISTCHAIN keyword on the RACDCERT command to display information about a certificate owned by a user ID, SITE, or CERTAUTH, and its issuers' certificates owned by CERTAUTH in a chain of certificates.

1. User WEBADM has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. She issues the RACDCERT command shown to see information about a certificate and the issuers' certificates that are owned by CERTAUTH in a chain of certificates.

```

RACDCERT ID(CHOI) LISTCHAIN(LABEL('samplecert'))

Certificate 1:
Digital certificate information for user CHOI:

  Label: samplecert
  Certificate ID: 2QbmxsPI1smJl140FmaPy
  Status: TRUST
  Start Date: 2011/10/20 00:00:00
  End Date: 2025/10/20 23:59:59
  Serial Number:
    >05<
  Issuer's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
  Subject's Name:
    >CN=samplecert.0=Test.SP=Poughkeepsie.C=US<
  Subject's AltNames:
    IP: 127.0.0.5
    EMail: choi at us.ibm.com
    Domain: www.ibm.com
  Signing Algorithm: sha256RSA
  Key Usage: HANDSHAKE
  Key Type: RSA
  Key Size: 2048
  Private Key: Yes
  PKDS Label: SAMPLECERT
  Certificate Fingerprint (SHA256):
    3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2
  Ring Associations:
    Ring Owner: CHOI
    Ring:
      >testing<

Certificate 2:
Digital certificate information for CERTAUTH:

  Label: sampleCA
  Certificate ID: 2PabcsPI1smJl140FmaPx
  Status: TRUST
  Start Date: 2010/03/22 00:00:00
  End Date: 2028/10/22 23:59:59
  Serial Number:
    >02<
  Issuer's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
  Subject's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
  Signing Algorithm: sha256RSA
  Key Usage: CERTSIGN
  Key Type: RSA
  Key Size: 2048
  Private Key: Yes
  PKDS Label: SAMPLECA
  Certificate Fingerprint (SHA256):
    A3:5A:89:FA:B4:81:DF:D3:32:B3:18:9B:80:42:E9:36:
    14:D8:93:D7:EE:94:4E:10:A7:93:4E:F2:6D:76:83:B5
  Ring Associations:
    Ring Owner: CHOI
    Ring:
      >testing<

Certificate 3:
Digital certificate information for CERTAUTH:

  Label: MasterCA
  Certificate ID: 2KbmxsPI1smJl140FmaPm
  Status: TRUST
  Start Date: 2008/04/20 00:00:00
  End Date: 2038/04/20 23:59:59
  Serial Number:
    >00<
  Issuer's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
  Subject's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
  Signing Algorithm: sha256RSA
  Key Usage: CERTSIGN
  Key Type: RSA
  Key Size: 4096
  Private Key: Yes
  PKDS Label: MASTERCA
  Certificate Fingerprint (SHA256):
    5E:22:79:FA:B4:81:DF:D3:32:E3:18:9B:85:42:E9:36:
    13:D8:93:D7:EE:94:4E:10:A7:93:4E:F1:6D:92:93:BA
  Ring Associations:
    Ring Owner: CHOI
    Ring:
      >testing<

Chain information:
  Chain contains 3 certificate(s), chain is complete
  Chain contains ring in common: CHOI/testing

```

2. User WEBADM has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. She issues the RACDCERT command shown to see information about a certificate and the issuers' certificates that are owned by CERTAUTH in a chain of certificates. One certificate is expired, and one certificate is a NOTRUST certificate.

```
RACDCERT ID(CHOI) LISTCHAIN(LABEL('samplecert'))

Certificate 1:
Digital certificate information for user CHOI:

Label: samplecert
Certificate ID: 2QbmxsPI1smJl40FmaPy
Status: TRUST
Start Date: 2010/10/20 00:00:00
End Date: 2011/10/20 23:59:59
Serial Number:
>05<
Issuer's Name:
>CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=samplecert.0=Test.SP=Poughkeepsie.C=US<
Subject's AltNames:
IP: 127.0.0.5
Email: choi at us.ibm.com
Domain: www.ibm.com
Signing Algorithm: sha256RSA
Key Usage: HANDSHAKE
Key Type: RSA
Key Size: 2048
Private Key: Yes
PKDS Label: SAMPLECERT
Certificate Fingerprint (SHA256):
3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2
Ring Associations:
Ring Owner: CHOI
Ring:
>testing<

Certificate 2:
Digital certificate information for CERTAUTH:

Label: sampleCA
Certificate ID: 2PabcsPI1smJl40FmaPx
Status: NOTRUST
Start Date: 2010/03/22 00:00:00
End Date: 2025/10/22 23:59:59
Serial Number:
>02<
Issuer's Name:
>CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 2048
Private Key: Yes
PKDS Label: SAMPLECA
Certificate Fingerprint (SHA256):
A3:5A:89:FA:B4:81:DF:D3:32:B3:18:9B:80:42:E9:36:
14:D8:93:D7:EE:94:4E:10:A7:93:4E:F2:6D:76:83:B5
Ring Associations:
Ring Owner: CHOI
Ring:
>testing<

Certificate 3:
Digital certificate information for CERTAUTH:

Label: MasterCA
Certificate ID: 2KbmxsPI1smJl40FmaPm
Status: TRUST
Start Date: 2008/04/20 00:00:00
End Date: 2038/04/20 23:59:59
Serial Number:
>00<
Issuer's Name:
>CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 4096
Private Key: Yes
PKDS Label: MASTERCA
Certificate Fingerprint (SHA256):
5E:22:79:FA:B4:81:DF:D3:32:E3:18:9B:85:42:E9:36:
13:D8:93:D7:EE:94:4E:10:A7:93:4E:F1:6D:92:93:BA
Ring Associations:
Ring Owner: CHOI
Ring:
>testing<

Chain information:
Chain contains 3 certificate(s), chain is complete
Chain contains ring in common: CHOI/testing
Chain contains NOTRUST certificate(s)
Chain contains expired certificate(s)
```

3. User WEBADM has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. She issues the RACDCERT command shown to see information about a certificate and the issuers' certificates that are owned by CERTAUTH in a chain of certificates. The chain is incomplete, and there is no common ring.

```

Certificate 1:
Digital certificate information for user CHOI:

Label: samplecert
Certificate ID: 2QbmxsPI1smJl40FmaPy
Status: TRUST
Start Date: 2010/10/20 00:00:00
End Date: 2025/10/20 23:59:59
Serial Number:
>05<
Issuer's Name:
>CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=samplecert.0=Test.SP=Poughkeepsie.C=US<
Subject's AltNames:
IP: 127.0.0.5
Email: choi at us.ibm.com
Domain: www.ibm.com
Signing Algorithm: sha256RSA
Key Usage: HANDSHAKE
Key Type: RSA
Key Size: 2048
Private Key: Yes
PKDS Label: SAMPLECERT
Certificate Fingerprint (SHA256):
3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2
Ring Associations:
Ring Owner: CHOI
Ring:
>testring<

```

```

Certificate 2:
Digital certificate information for CERTAUTH:

```

```

Label: sampleCA
Certificate ID: 2PabcsPI1smJl40FmaPx
Status: TRUST
Start Date: 2010/03/22 00:00:00
End Date: 2028/10/22 23:59:59
Serial Number:
>02<
Issuer's Name:
>CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 2048
Private Key: Yes
PKDS Label: SAMPLECA
Certificate Fingerprint (SHA256):
A3:5A:89:FA:B4:81:DF:D3:32:B3:18:9B:80:42:E9:36:
14:D8:93:D7:EE:94:4E:10:A7:93:4E:F2:6D:76:83:B5
Ring Associations:
Ring Owner: WAI
Ring:
>testring2<

```

```

Chain information:
Chain contains 2 certificate(s), chain is incomplete
Chain contains no ring in common

```

## Examples of checking digital certificate information

1. User NETADMN has a digital certificate in a data set, and is uncertain who it belongs to, and whether or not it has been defined. The digital certificate is in data set 'NETADMN.SOMEONZ.CERT'. NETADMN has UPDATE authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the following RACDCERT, and the output he receives indicates that it has already been defined for user GTM:

```
RACDCERT CHECKCERT('NETADMN.SOMEONZ.CERT')
```

Digital certificate information for user GTM:

```
Label: LABEL000000001
Certificate ID: 2QPH49TH49RAw4WZo4mGiY0Bo4VA
Status: NOTRUST
Start Date: 2010/11/11 00:00:00
End Date: 2028/11/11 23:59:59
Serial Number:
    >84<
Issuer's Name:
    >CN=BobsBank Class 2<
Subject's Name:
    >loanOf@BobsBank.com.CN=G.T.Miles.T=President.OU=Loans.O=BobsBank,INC<
    >..SP=NY.L=Internet.C=USA<
Signing Algorithm: sha256RSA
Key Type: RSA
Key Size: 2048
Private Key: NO
Certificate Fingerprint (SHA256):
    2B:3A:77:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:C2:54
```

2. User USERA finds a digital certificate and is uncertain who it belongs to, and whether or not it has been defined to RACF. The digital certificate is contained in data set 'NETADMN.SOMEONZ.CERT' and is associated with user GTM. USERA has READ authority to the data set 'NETADMN.SOMEONZ.CERT'. He issues the following RACDCERT command. The output he receives reflects only the certificate information contained in the data set, and does not include certificate information contained in the RACF database. Note that the listing contains the same level of information that NETADMN receives in Example “3” on page 560.

```
RACDCERT CHECKCERT('NETADMN.SOMEONZ.CERT')
```

```
Start Date: 2010/11/11 00:00:00
End Date: 2028/11/11 23:59:59
Serial Number:
    >84<
Issuer's Name:
    >CN=BobsBank Class 2<
Subject's Name:
    >loanOf@BobsBank.com.CN=G.T.Miles.T=President.OU=Loans.O=BobsBank,INC<
    >..SP=NY.L=Internet.C=USA<
Signing Algorithm: sha1RSA
Certificate Fingerprint (SHA256):
    2B:3A:77:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:C2:54
```

3. User NETADMN has a digital certificate in a data set, and is uncertain who it belongs to, and whether or not it has been defined. The digital certificate is in data set 'NETADMN.SOMEELSZ.CERT'. NETADMN has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the following RACDCERT command, and the output he receives indicates that the certificate is not associated with a user ID.

```
RACDCERT CHECKCERT('NETADMN.SOMEELSZ.CERT')
```

```
Start Date: 2010/03/18 14:58:37
End Date: 2028/03/17 14:58:37
Serial Number:
    >79<
Issuer's Name:
    >CN=BobsBank Class 2<
Subject's Name:
    >brchMGR@BobsBank.com.CN=J. Miles.T=Manager.OU=Branch2.O=BobsBank,INC<
    >..SP=NY.L=Internet.C=USA<
Signing Algorithm: sha1RSA
Certificate Fingerprint (SHA256):
    A4:8A:57:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:85:B3:43
```

4. User NETADMN has a chain of digital certificates in a data set, and wants to know if the digital certificates are defined to RACF. The digital certificates are in data set 'NETADMN.SOMECHN.CERT'.



NETADMN has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the following RACDCERT command, and the output he receives indicates that the certificates are not in RACF, because the **Label**, **Certificate ID**, and **Status** fields are not shown for any of them.

```
RACDCERT CHECKCERT('NETADMN.SOMECHN.CERT')

Certificate 1:
Start Date: 2011/10/20 00:00:00
End Date: 2028/10/20 23:59:59
Serial Number:
>05<
Issuer's Name:
>CN=sampleCA.O=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=samplecert.O=Test.SP=Poughkeepsie.C=US<
Subject's AltNames:
IP: 127.0.0.5
EMail: choi at us.ibm.com
Domain: www.ibm.com
Signing Algorithm: sha1RSA
Key Usage: HANDSHAKE
Key Type: RSA
Key Size: 2048
Certificate Fingerprint (SHA256):
3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2

Certificate 2:
Start Date: 2010/03/22 00:00:00
End Date: 2030/10/22 23:59:59
Serial Number:
>02<
Issuer's Name:
>CN=MasterCA.O=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=sampleCA.O=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 2048
Certificate Fingerprint (SHA256):
3B:7C:86:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:4D:88:F6:99

Certificate 3:
Start Date: 2008/04/20 00:00:00
End Date: 2038/04/20 23:59:59
Serial Number:
>00<
Issuer's Name:
>CN=MasterCA.O=Test.SP=Poughkeepsie.C=US<
Subject's Name:
>CN=MasterCA.O=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 4096
Certificate Fingerprint (SHA256):
54:C6:78:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
15:D8:93:D7:EE:94:4E:11:A7:93:6E:F2:4D:86:E2:A4

Chain information:
Chain contains 3 certificate(s), chain is complete
```

5. User NETADMN has a chain of digital certificates in a data set, and wants to know if the digital certificates are defined to RACF. The digital certificates are in data set 'NETADMN.SOMECHN.CERT'. NETADMN has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the following RACDCERT command, and the output he receives indicates that only the end-entity certificate is in RACF, because the **Label**, **Certificate ID**, and **Status** fields are shown for that certificate but not the others. The output also shows that the end-entity certificate has expired, because the end date is before the current date.

```
RACDCERT CHECKCERT('NETADMN.SOMECHN.CERT')
```

Certificate 1:

Digital certificate information for user CHOI:

```
Label: samplecert
Certificate ID: 2QbmxsPI1smJl40FmaPy
Status: TRUST
Start Date: 2010/10/20 00:00:00
End Date: 2019/10/20 23:59:59
Serial Number:
    >05<
Issuer's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
    >CN=samplecert.0=Test.SP=Poughkeepsie.C=US<
Subject's AltNames:
    IP: 127.0.0.5
    EMail: choi at us.ibm.com
    Domain: www.ibm.com
Signing Algorithm: sha1RSA
Key Usage: HANDSHAKE
Key Type: RSA
Key Size: 2048
Private Key: Yes
PKDS Label: SAMPLECERT
Certificate Fingerprint (SHA256):
    3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2
```

Certificate 2:

```
Start Date: 2010/03/22 00:00:00
End Date: 2030/10/22 23:59:59
Serial Number:
    >02<
Issuer's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 2048
Certificate Fingerprint (SHA256):
    A3:5A:89:FA:B4:81:DF:D3:32:B3:18:9B:80:42:E9:36:
    14:D8:93:D7:EE:94:4E:10:A7:93:4E:F2:6D:76:83:B5
```

Certificate 3:

```
Start Date: 2008/04/20 00:00:00
End Date: 2038/04/20 23:59:59
Serial Number:
    >00<
Issuer's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Subject's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
Signing Algorithm: sha256RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 4096
Certificate Fingerprint (SHA256):
    5E:22:79:FA:B4:81:DF:D3:32:E3:18:9B:85:42:E9:36:
    13:D8:93:D7:EE:94:4E:10:A7:93:4E:F1:6D:92:93:BA
```

Chain information:

```
Chain contains 3 certificate(s), chain is complete
Chain contains expired certificate(s)
```

6. User NETADMN has a chain of digital certificates in a data set, and wants to know if the digital certificates are defined to RACF. The digital certificates are in data set 'NETADMN.SOMECHN.CERT'. NETADMN has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the following RACDCERT command, and the output he receives indicates that the certificates are not in RACF, because the **Label**, **Certificate ID**, and **Status** fields are not shown for any of them. The output also shows that the signature on certificate 2 is not valid.

```

RACDCERT CHECKCERT('NETADMN.SOMECHN.CERT')

Certificate 1:
  Start Date: 2011/10/20 00:00:00
  End Date:   2028/10/20 23:59:59
  Serial Number:
    >05<
  Issuer's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
  Subject's Name:
    >CN=samplecert.0=Test.SP=Poughkeepsie.C=US<
  Subject's AltNames:
    IP: 127.0.0.5
    Email: choi at us.ibm.com
    Domain: www.ibm.com
  Signing Algorithm: sha1RSA
  Key Usage: HANDSHAKE
  Key Type: RSA
  Key Size: 2048
  Private Key: No
  Certificate Fingerprint (SHA256):
    3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2

Certificate 2:
  Start Date: 2010/03/22 00:00:00
  End Date:   2030/10/22 23:59:59
  Serial Number:
    >02<
  Issuer's Name:
    >CN=MasterCA.0=Test.SP=Poughkeepsie.C=US<
  Subject's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
  Signing Algorithm: sha256RSA
  Key Usage: CERTSIGN
  Key Type: RSA
  Key Size: 2048
  Private Key: No
  Certificate Fingerprint (SHA256):
    A3:5A:89:FA:B4:81:DF:D3:32:B3:18:9B:80:42:E9:36:
    14:D8:93:D7:EE:94:4E:10:A7:93:4E:F2:6D:76:83:B5
IRRD302I Processing terminated. Problem found in certificate 2 in the
dataset.
IRRD112I The certificate that you are processing does not have a
valid signature.

```

7. User NETADMN has a chain of digital certificates in a data set, and wants to know if the digital certificates are defined to RACF. The digital certificates are in data set 'NETADMN.SOMECHN.CERT'. NETADMN has CONTROL authority to the FACILITY class resource IRR.DIGTCERT.LIST. He issues the following RACDCERT command, and the output he receives indicates that certificate 1 is not in RACF, because the **Label**, **Certificate ID**, and **Status** fields are not shown for it. The output also shows that the name on certificate 2 contains a character that is not valid. Certificate 2 is not displayed, and processing of the command stops.

```
Certificate 1:
  Start Date: 2011/10/20 00:00:00
  End Date:   2028/10/20 23:59:59
  Serial Number:
    >05<
  Issuer's Name:
    >CN=sampleCA.0=Test.SP=Poughkeepsie.C=US<
  Subject's Name:
    >CN=samplecert.0=Test.SP=Poughkeepsie.C=US<
  Subject's AltNames:
    IP: 127.0.0.5
    EMail: choi at us.ibm.com
    Domain: www.ibm.com
  Signing Algorithm: sha256RSA
  Key Usage: HANDSHAKE
  Key Type: RSA
  Key Size: 2048
  Private Key: No
  Certificate Fingerprint (SHA256):
    3B:4A:66:FA:B4:81:DF:D3:12:F3:18:9B:85:42:E9:36:
    15:D8:93:D7:EE:94:4E:11:A7:93:4E:F2:6D:88:51:A2
  IRRD302I Processing terminated. Problem found in certificate 2 in the
  dataset.
  IRRD182I Unexpected character encountered.
```

## Examples of altering digital certificate information

To alter information about a certificate, use the RACDCERT ALTER command. The only alterable information about a certificate is the TRUST status and the label of a certificate. See [z/OS Security Server RACF Command Language Reference](#) for more information.

### Using the TRUST option

User CERTCTL requests that the status of user ID NET1's certificate from VeriSign be changed from not trusted to trusted. Because NET1 has more than one certificate defined, the certificate label must be specified. User CERTCTL has UPDATE access to resource IRR.DIGTCERT.ALTER in the FACILITY class by RACFADM.

#### Example:

```
RACDCERT ID(NET1) ALTER(LABEL('NET1 Cert1')) TRUST
```

### Using the NOTRUST option

User CERTCTL requests that the status of user ID NET2's certificate from VeriSign be changed from trusted to not trusted. Because user NET2 has only one certificate defined, neither the serial number nor the issuer's distinguished name needs to be specified. User CERTCTL has been given UPDATE access to resource IRR.DIGTCERT.ALTER in the FACILITY class by RACFADM.

#### Example:

```
RACDCERT ID(NET2) ALTER NOTRUST
```

## Examples of deleting digital certificates

To delete user certificates, CA certificates and site certificates, use the RACDCERT DELETE command. For user certificates, you must uniquely identify the certificate you want deleted. Therefore, if the user has more than one certificate, you must provide either:

- LABEL, or
- SERIALNUMBER and ISSUERSDN

The RACDCERT command uses the DELETE operand in the following forms:

```
DELETE(LABEL('label-name'))
DELETE(SERIALNUMBER(serial-number) ISSUERSDN('issuer's-dn'))
```

When you delete a certificate that is connected to a key ring, the certificate is automatically removed from the key ring.

Because PKCS #11 tokens are managed by ICSF, not RACF, when you delete a certificate that is bound in a token, the equivalent certificate object in the token is unchanged.

For detailed syntax and usage information, see [z/OS Security Server RACF Command Language Reference](#).

## Deleting a user certificate

User CERTCTL deletes user ID NET1's certificate from VeriSign. Because NET1 has more than one certificate defined, the label must be specified. User CERTCTL has UPDATE access to resource IRR.DIGTCERT.ALTER in the FACILITY class by RACFADM.

### Example:

```
RACDCERT ID(NET1) DELETE(LABEL('NET1 Cert2'))
```

## Deleting a CA or SITE certificate

User CERTCTL deletes a local CA certificate and a site certificate. User CERTCTL has been given CONTROL access to resource IRR.DIGTCERT.ALTER in the FACILITY class by RACFADM.

### Examples:

```
RACDCERT CERTAUTH DELETE(LABEL('Local PKIX CA'))
RACDCERT SITE DELETE(LABEL('('Shared Server B')'))
```

## DIGTCERT general resource profiles

Each profile in the DIGTCERT class contains a digital certificate and information related to the certificate. The APPLDATA field contains the RACF user ID associated with this digital certificate. The UACC contains the status of the certificate. When a certificate's status is set to TRUST, the `initACEE` callable service allows the certificate to be mapped to the RACF user ID contained in the APPLDATA field.

**Important:** Do not enable generic profile checking for the DIGTCERT class by issuing the `SETROPTS GENERIC(DIGTCERT)` or `SETROPTS GENCMD(DIGTCERT)` command. DIGTCERT and DIGTRING do not support generic profile checking. If you have GENERIC or GENCMD turned on for the DIGTCERT class when the certificate is created or added, and its Issuer's Distinguished Name contains any generic characters (\*, & and %), a generic certificate profile will be created. You may check from the output of `SEARCH CLASS(DIGTCERT)` to see if a letter 'G' appears on that entry. This generic feature will cause some unexpected behavior when the certificate is being used by other programs. You need to delete the certificate and add it back after turning off GENERIC and GENCMD in the DIGTCERT class.

## DIGTCERT profile names

The name of a DIGTCERT profile is derived from the certificate's serial number and the issuer's distinguished name (IDN). Any character in either value that would not be valid in a RACF profile name, such as a blank, is replaced with the `¢` character (X'4A').

The maximum length of a DIGTCERT profile name is 246 characters. The format of the profile name is based on the combined length of the certificate's serial number and the issuer's distinguished name (IDN), including the period.

When the combined length of the value of *serial-number.issuer's-distinguished-name* is 246 characters or less, the name of the DIGTCERT profile uses the following format:

```
serial-number.issuer's-distinguished-name
```

**Example:** If the certificate's serial number is 41D87A3B05DE6FBD466C2069661E3872 and the issuer's distinguished name is OU=VeriSign Class1.0=VeriSign.L=Internet, the profile name of the DIGTCERT profile is as follows:

```
41D87A3B05DE6FBD466C2069661E3872.OU=VeriSignClass1.0=VeriSign.L=Internet
```

When the combined length of the value of *serial-number.issuer's-distinguished-name* exceeds 246 characters, the name of the DIGTCERT profile uses the following format, where the *certificate-hash* value is a hexadecimal representation of the certificate in a hashed form:

```
serial-number.<first-portion-of-IDN><certificate-hash><last-portion-of-IDN>
```

**Example:** If the certificate's serial number is 0E and the issuer's distinguished name is as follows, the resulting profile name is as shown:

### Issuer's distinguished name:

```
CN=Entrust Certification Authority - L1B.OU=(c) 2008 Entrust, Inc  
.OU=www.entrust.net/CPS is incorporated by reference.OU=CPS CON  
TAINS IMPORTANT LIMITATIONS OF WARRANTIES AND LIABILITY.OU=AND A  
DDITIONAL TERMS GOVERNING USE AND RELIANCE.OU=Entrust, Inc.C=US
```

### DIGTCERT profile name:

```
0E.CN:Entrust Certification Authority - L1B.OU:(c) 2008 Entrust,  
Inc..OU:www.entrust.net/CPS i de9f2c7fd25e1b3afad3e85a0bd17d9b10  
0db4b32fd4e1c67a2d28fced849ee1 ES AND LIABILITY.OU:AND ADDITIONA  
L TERMS GOVERNING USE AND RELIANCE.OU:Entrust, Inc.C=US
```

When a DIGTCERT profile name contains a certificate hash value, each occurrence of the equal sign (=) delimiter is replaced with a colon (:).

## Ownership of DIGTCERT profiles

The owner of a DIGTCERT profile is the user ID that issued the RACDCERT command to create the profile. The owner has no authority over the profile or the resources it protects. RACF does not use profile owner information for authorization or any other purpose. The profile owner of a DIGTCERT profile cannot be changed. Because it is unused, there is no need to change the owner.

The profile owner is not listed in RACDCERT LIST output. However, the user ID of the profile owner can be seen in the output of the RLIST DIGTCERT \* command (although RLIST is not intended for use with this class) and in the output of the database unload (IRRDBU00) utility.

## RACLISTing the DIGTCERT class

**Guideline:** Activate and RACLIST the DIGTCERT class if you use digital certificates with applications that require high performance, such as applications that access WebSphere® Application Server.

If the DIGTCERT class is not RACLISTed, digital certificates can still be used but performance might be impacted when applications that retrieve certificates from RACF must wait while RACF retrieves them from the RACF database instead of from virtual storage.

**Rule:** You must activate each class you want to RACLIST. For example:

```
SETROPTS RACLIST(DIGTCERT) CLASSACT(DIGTCERT)
```

After creating a new digital certificate, refresh the DIGTCERT class by issuing the SETROPTS RACLIST(DIGTCERT) REFRESH command. If you do not refresh the RACLISTed DIGTCERT profiles, RACF will still use the new digital certificate. However, performance might be impacted because applications that retrieve certificates from RACF will wait while RACF retrieves the new certificate from the RACF database.

**Restriction:** Any RACLISTed digital certificates that you alter, re-add or delete will not reflect your changes until you refresh the DIGTCERT class. This is because RACF uses RACLISTed profiles *before* profiles in the RACF database. Therefore, to make your changes effective, refresh the DIGTCERT class.

## RACF and key rings

A key ring is a collection of certificates that identify a networking trust relationship (also called a trust policy). In a client-server network environment, entities identify themselves using digital certificates. Server applications on z/OS that want to establish network connections to other entities can use RACF key rings and other related services to determine the trustworthiness of the client or peer entity.

A *virtual key ring* is the set of all certificates owned by a user ID. This set of certificates is used, like a real key ring, by a user or server application to determine the trustworthiness of a client or peer. Each RACF user ID is associated with a virtual key ring. In contrast to a real key ring, a virtual key ring is not added to RACF.

Each of the following commands list the contents of a virtual key ring:

### Examples:

```
RACDCERT ID(userid) LIST
RACDCERT CERTAUTH LIST
RACDCERT SITE LIST
```

The most common type is the CERTAUTH virtual key ring, which is used when an application validates the certificates of others but has no need for its own certificate and private key. See [“Using a virtual key ring”](#) on page 568 for an example.

System SSL and other security middleware use the R\_data1ib callable service (IRRSDL00 or IRRSDL64) to retrieve certificate information from RACF. In order for an application to retrieve certificates and private keys from RACF, both of the following conditions must be met:

- The certificates must be connected to a RACF key ring (a real or *virtual* key ring) or a z/OS PKCS #11 token. The key ring or token is the data store that R\_data1ib opens, reads, and closes as directed by the application.
- The application must have appropriate access authority to the key ring or the token. For authorization details, see [Usage Notes for R\\_data1ib \(IRRSDL00 or IRRSDL64\) in z/OS Security Server RACF Callable Services](#).

Applications can also use R\_data1ib callable service to manage keys rings (virtual key rings are not included). Authorized applications can create key rings and connect certificates to key rings. See [“R\\_data1ib \(IRRSDL00 or IRRSDL64\) callable service”](#) on page 606 for information about controlling applications that use this callable service.

The usage assigned to a certificate when it is connected to a key ring indicates its intended purpose. Personal certificates are to be used by the local server application to identify itself. Certificate-authority certificates are to be used to verify the peer entity's certificate. Peers with certificates issued by certificate authorities connected to the key ring are considered trusted network entities. There might be a few certificate validation applications that treat a certificate that is connected to a key ring with usage site as a valid certificate authority certificate to bypass the normal certificate verification tests; for example, an expired certificate can be considered trusted. The most popular exploiter of R\_data1ib, System SSL, does not make use of the site certificate.

**Restriction:** If the calling application does not specify the status of the certificates to be returned, certificates marked NOTRUST would not be retrieved using the R\_data1ib callable service even if they are connected to a key ring. RACF hides them from the calling application and does not indicate that they are connected to the key ring.

## DIGTRING general resource profiles

Key rings are associated with specific RACF user IDs. A RACF user ID can have more than one key ring. Key rings are managed using the RACDCERT command, and are maintained in the general resource class called DIGTRING.

RACF key rings provide an installation-wide method to share key rings across multiple servers. You can decentralize responsibility to manage key rings by granting access to resources in the FACILITY class or the RDATA LIB class. (See [“Examples of controlling the use of the RACDCERT command using the FACILITY class” on page 550.](#)) However, you can retain sole ability to connect certificates to key rings at your installation. This will allow you to implement and maintain a centralized security or trust policy toward certificate authorities. For example, you can establish key rings for servers that contain certificates from only approved certificate authorities. You can then delegate other key ring responsibilities to server administrators who will be able remove certificates from their key rings, but not add certificates from unapproved sources.

Key rings are identified by ring names that are 1 - 237 characters in length. Each key ring profile in the DIGTRING class contains references to those certificates that are part of that key ring. Profile names are in the form:

```
userid.ring-name
```

When you delete a user ID, DELUSER command processing deletes the user's key rings by deleting the associated resources in the DIGTRING class. The certificates referenced in the key ring are not deleted unless they too are associated with the user ID being deleted.

**Important:** Do not enable generic profile checking for the DIGTRING class by issuing the SETROPTS GENERIC(DIGTRING) or SETROPTS GENERIC(\*) command. Some classes, such as DIGTCERT and DIGTRING, do not support generic profile checking. These and other classes might already have profile names that contain generic characters (\*, &, and %). If a class already has profile names that contain generic characters, avoid issuing the SETROPTS GENERIC(classname) command for that class. Enabling generic profile checking for such a class prevents RACF from using previously defined profiles that contain generic characters in the name.

## Sharing a private key in a key ring

You can share a certificate among two or more servers (user IDs) by administering profiles in either the FACILITY or the RDATA LIB class. Sharing a certificate can save you the expense of purchasing a new certificate for each server and avoids the need to export and import certificate copies.

You can also share a certificate's private key among two or more servers (user IDs) by administering profiles in the RDATA LIB class. Sharing a private key requires a high degree of authority for each server involved. The key ring that contains the shared certificate must be protected. Also, each server must be configured to access the shared key ring and have sufficient access authority to read the private key with the R\_data lib callable service.

For an example of the required setup, see [“Scenario 7: Sharing one certificate among multiple servers” on page 599.](#)

## Using a virtual key ring

For applications using System SSL, such as z/OS FTP, or other middleware programs that read RACF key rings through the R\_data lib callable service, a virtual key ring can be specified in place of a real key ring, whenever a real key ring is expected. To include virtual key rings, the application user specifies an asterisk (\*) for the key ring name along with the ring owner's user ID using the form *ring-owner/\**. The ring owner user IDs for the certificates under CERTAUTH and SITE are in the form of \*AUTH\* and \*SITE\*.

### Example using the z/OS FTP client with TLS

A z/OS FTP client can use a virtual CERTAUTH key ring to authenticate the FTP server by following these steps:



1. The user specifies the following KEYRING directive in her FTP.DATA file:

```
KEYRING *AUTH*/*
```

2. The user directs FTP to use TLS by specifying **-a TLS** or **-r TLS** on the FTP command:

```
ftp -r TLS ftp.ibm.com
```

**Note:** The virtual key ring is used only to authenticate the FTP server when client authentication is not required.

## RACF and z/OS PKCS #11 tokens

Tokens are containers that hold digital certificates and keys. z/OS supports both clear and secure keys in the PKCS #11 tokens that are provided and managed by ICSF. You can use RACF in the following ways to define and manage certain certificate objects in a token (certificates, public keys, and private keys).

- You can use the following RACDCERT command functions:
  - ADDTOKEN: Defines a new empty token.
  - DELTOKEN: Deletes an existing token and all its contents.
  - LISTTOKEN: Displays information about the objects that are contained in the token.
  - BIND: Connects a RACF certificate, its public key, and (in some cases) its private key, to an existing token.
  - UNBIND: Removes a certificate and its keys from an existing token.
  - IMPORT: Adds a certificate to RACF from an existing token.

See *z/OS Security Server RACF Command Language Reference* for syntax and usage information about these functions of the RACDCERT command.

- You can authorize applications to use the R\_data1ib (IRRSDL00 or IRRSDL64) callable service to read and extract token information. For details, see *RACF Authorization for R\_data1ib (IRRSDL00 or IRRSDL64)* in *z/OS Security Server RACF Callable Services*.
- You can use resources in the CRYPTOZ class to control access to tokens. See *Controlling token access and key policy* in *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications*.

Because tokens are managed by ICSF, not RACF, other applications can use ICSF functions to change tokens without updating the certificate information in the RACF database. Similarly, RACF changes to digital certificates already bound to a token are not reflected in the token information that is maintained by ICSF. Therefore, the following restrictions apply:

### restrictions:

- Deleting, altering, or renewing a RACF certificate that is bound to a token has no effect on the equivalent token objects that are managed by ICSF.
- Deleting or altering a certificate object in a token has no effect on the following objects:
  - The equivalent RACF certificate.
  - The equivalent certificate objects in other tokens.

## Creating and populating PKCS #11 tokens

You can create and populate a PKCS #11 token in the following ways:

- Using the ADDTOKEN and BIND functions of the RACDCERT command.

See *“Steps for creating and populating tokens”* on page 570.

- Using the gskkyman utility.

See *z/OS Cryptographic Services System SSL Programming* for details.

- Executing an application that invokes the PKCS #11 API provided by ICSF.

See *z/OS Cryptographic Services ICSF Writing PKCS #11 Applications* for detailed information about tokens and applications that use them.

## Steps for creating and populating tokens

**Before you begin:** Add an end-entity certificate and, optionally, a private key for the server or application for which you are creating and populating the token.

For the purpose of this example, the following certificates are assumed to exist. (The commands to create them are not shown.)

1. A root CA certificate installed under CERTAUTH with the label `Local Root CA for Servers`.
2. An end-entity certificate and private key installed under user FTPSRV with the label `FTP Key`. This certificate was signed by the first certificate.
3. An end-entity certificate and private key installed under user WEBSRV with the label `Web Key`. This certificate was also signed by the first certificate.

Perform the following steps to create and populate PKCS #11 tokens. The examples in these steps show how to use PKCS #11 tokens as key stores for an FTP server and a Web server.

1. Create two new tokens using your installation's naming convention for tokens.

The naming convention used in this example requires the high-level qualifier of the token name to match the owning user ID. In this example, the user IDs associated with the FTP and Web servers are FTPSRV and WEBSRV.

### Examples:

```
RACDCERT ADDTOKEN(ftpshr.ftp.server.pkcs11.token)
RACDCERT ADDTOKEN(websrv.web.server.pkcs11.token)
```

- 
2. Bind the root CA certificate to the two new tokens.

### Examples:

```
RACDCERT BIND(CERTAUTH LABEL('Local Root CA for Servers')
  TOKEN(ftpshr.ftp.server.pkcs11.token))
RACDCERT BIND(CERTAUTH LABEL('Local Root CA for Servers')
  TOKEN(websrv.web.server.pkcs11.token))
```

- 
3. Bind the end-entity root certificates to their respective tokens. Define each certificate as the default in its token.

### Examples:

```
RACDCERT BIND(ID(FTPSTRV) LABEL('FTP Key')
  TOKEN(ftpshr.ftp.server.pkcs11.token) DEFAULT)
RACDCERT BIND(ID(WEBSTRV) LABEL('Web Key')
  TOKEN(websrv.web.server.pkcs11.token) DEFAULT)
```

You have now created and populated PKCS #11 tokens for the FTP server and the Web server.

To begin using the new tokens, the Web server administrator must now configure each server to use its new token.

For example, if the Web server uses IBM HTTP Server, a keyfile directive must be added to its `httpd.conf` file:

### Example:

```
keyfile *TOKEN*/WEBSRV.WEB.SERVER.PKCS11.TOKEN SAF
```

**Note:** In this example, the SAF option indicates to SSL that the keyfile is a SAF key ring, rather than a key database file. The leading characters \*TOKEN\*/ in the keyfile name indicate to SAF that the key ring is, in fact, a token.

For details about adding a keyfile directive, see the [WebSphere Application Server IBM Documentation](http://www.ibm.com/docs/en/was) ([www.ibm.com/docs/en/was](http://www.ibm.com/docs/en/was)).

Similarly for the FTP server, a KEYRING definition must be added to its configuration file (FTP.DATA):

**Example:**

```
KEYRING *TOKEN*/FTPSRV.FTP.SERVER.PKCS11.TOKEN
```

**Note:** The leading characters \*TOKEN\*/ in the key ring name indicate to SAF that the key ring is, in fact, a token.

## Certificate name filtering

As more and more users access your system from the Web, you face an increasing administrative burden to securely manage their digital certificates. *Certificate name filtering* is a method for administering large numbers of user certificates, without storing each certificate in the RACF database. Certificates managed using certificate name filtering:

- Require no individual administration to be registered or to be replaced when they expire.
- Occupy very little space in the RACF database.
- Can be used to allow several users to share the same user ID in a secure manner.
- Can be selectively mapped to different user IDs based on system and application criteria.
- Are logged on use with audit records that include the associated user ID and the certificate's full subject's and issuer's name.

Certificate name filters are used to determine the operational user ID when RACF is called to create a security context for a client login using a certificate, such as during SSL client authentication. Certificate name filters cannot be used in protocols where the client certificate or the client private key is required. Therefore, certificate name filters are ideally suited for use with SSL client authentication which requires that only the client's root certificate, not the client certificate, be stored in the RACF database.

**Note:** Certificate name filters are *unrelated* to distributed identity filters. (See Chapter 28, “Distributed identity filters,” on page 671). An installation might choose to implement either certificate name filters or distributed identity filters, both types of filters, or neither.

## Interpreting the X.500 directory information tree

When you use certificate name filtering, RACF provides the ability to allow several users to share the same user ID on your system based on the *subject's distinguished name* and the *issuer's distinguished name* as contained in X.509 certificates. The subject's distinguished names and issuer's distinguished names for three sample certificates are listed in [Table 37 on page 571](#) and are shown in the address form used by RACF:

Table 37. Subject's and issuer's distinguished names

Subject's distinguished name	Issuer's distinguished name
CN=Agneta Berglund.OU=Sales.OU=New York.OU=US.O=World Sales Corp	OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
CN=Hiro Ogura.OU=Admin.OU=New York.OU=US.O=World Sales Corp	OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
CN=Timo Kokkonen.OU=Sales.OU=Los Angeles.OU=US.O=World Sales Corp	OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet

The distinguished names contained in the certificates shown in Table 37 on page 571 are represented in the X.500 directory information tree shown in Figure 52 on page 572. For a list of the components of the subject's X.509 distinguished name, see the syntax of the RACDCERT command in [z/OS Security Server RACF Command Language Reference](#).

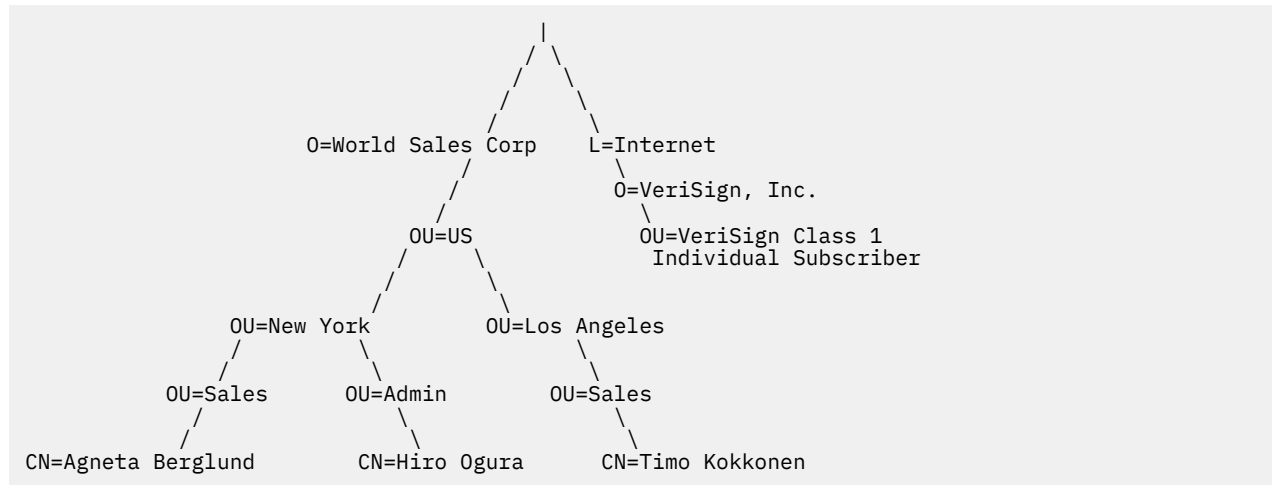


Figure 52. Example of an X.500 directory information tree

Now, let's look at the left branch of the tree in Figure 52 on page 572 as representing a hierarchical organization, with each level of the tree, or node, representing a different level within an organization. For example, Agneta works in the Sales department in New York for the US division of the World Sales Corporation. If viewed as a hierarchy of user groups, each level of the tree might represent increased access authority, with each group consisting of the groups below it.

For example, as an employee of World Sales, Agneta might have access to the internal phone numbers of all World Sales employees. As a member of the US division, she might also have access to the US division internal Web site, in addition to the phone numbers of all employees. Being in New York might allow her to run sales reports for the New York office, as well as to access the Web site and employee phone numbers. Being in the Sales department might allow her to place customer orders, in addition to all other access authorities.

You can associate a user ID with each node in a directory information tree using certificate name filtering. Each user ID can represent a number of users, each of whom has one or more digital certificates. Therefore, you can administer several certificates and the access authorities for several users, through a single user ID. For each node that you associate with a user ID, you create a certificate name filter that contains partial or full distinguished names, depending on where the node falls in the hierarchy.

## Creating certificate name filters

You create certificate name filters using the RACDCERT MAP command. *Certificate name filters* are used by RACF (specifically, the `initACEE` callable service) to analyze the subject's and issuer's distinguished names in a given certificate to determine the user ID to associate with it. You can create filters based on the full issuer's distinguished names in order to administer all certificates by a given issuer as a single user ID. You can also create filters based on portions of the subject's distinguished name, and a variety of filters based on certain combinations of partial and full distinguished names. See [“Types of certificate name filters”](#) on page 574.

### Example:

The RACDCERT MAP command shown in Figure 53 on page 573 creates a certificate name filter based on the full issuer's distinguished name. This filter associates the user ID `WEBUSER` to any user presenting a certificate issued by VeriSign Class 1, who does not have an individual certificate registered with RACF on your system.

```
RACDCERT ID(WEBUSER) MAP WITHLABEL('INTERNET OTHERS') TRUST
IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')
SETOPTS RACLIST(DIGTNMAP) REFRESH
```

Figure 53. Sample RACDCERT MAP command for creating an issuer's name filter

See [z/OS Security Server RACF Command Language Reference](#) for more information about the RACDCERT MAP command.

## Assigning user IDs to certificate name filters

You must define a RACF user ID for each user ID you associate with a certificate name filter. Because these user IDs are shared, consider assigning the PROTECTED and RESTRICTED attributes to each one.

### Example:

```
ALTUSER WEBUSER NOPASSWORD RESTRICTED
```

The PROTECTED attribute protects the user ID from being used to logon directly to the system and from being revoked through incorrect password and password phrase attempts. See [“Defining protected user IDs” on page 74](#).

The RESTRICTED attribute ensures that the user ID is not used to access protected resources it is not specifically authorized to access. See [“Defining restricted user IDs” on page 75](#).

## Activating certificate name filtering

When you create a certificate name filter, RACDCERT MAP processing automatically creates a mapping profile in the DIGTNMAP class to represent the new filter. The DIGTNMAP class must be active and SETOPTS RACLIST processing must be active for the DIGTNMAP class. Before creating any certificate name filters using the RACDCERT MAP command, you must issue the following command:

```
SETOPTS CLASSACT(DIGTNMAP) RACLIST(DIGTNMAP)
```

Once SETOPTS RACLIST processing is active for the DIGTNMAP class, you must refresh the DIGTNMAP class in order for new or changed certificate name filters to take effect. After creating or changing a certificate name filter, you must issue the following command:

```
SETOPTS RACLIST(DIGTNMAP) REFRESH
```

## DIGTNMAP general resource profiles

RACDCERT MAP processing automatically creates mapping profiles in the DIGTNMAP class for each certificate name filter you create. When you map a certificate name filter to a RACF user ID, both the filter and the user ID are stored in the mapping profile. DIGTNMAP profiles should not be administered using the RDEFINE, RALTER or RDELETE commands. These commands do not operate with the DIGTNMAP class.

The SEARCH FILTER and RLIST commands are not intended for use with profiles in the DIGTNMAP class and will deliver unpredictable results. These profiles can only be displayed using the RACDCERT LISTMAP command: For example:

```
RACDCERT ID(WEBUSER) LISTMAP
```

Based on the output of the RACDCERT LISTMAP command shown in [Figure 54 on page 574](#), there is one certificate name filter associated with the WEBUSER user ID.

```

Mapping information for user WEBUSER:
Label: INTERNET OTHERS
Status: TRUST
Issuer's Name Filter:
>OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet<
Subject's Name Filter:
><

```

Figure 54. Sample output from the LISTMAP command for an issuer's name filter

See [z/OS Security Server RACF Command Language Reference](#) for more information about the RACDCERT LISTMAP command.

## Types of certificate name filters

There are three basic types of certificate name filters, based on the X.509 distinguished names referenced in the filter definition. They are:

1. Issuer's name filter
2. Subject's name filter
3. Subject's and issuer's name filter

### Issuer's name filter

An issuer's name filter contains a full or partial issuer's distinguished name. For an example of using the RACDCERT MAP command to create an issuer's name filter, see [Figure 53 on page 573](#). For an example of the output of a RACDCERT LISTMAP command displaying mapping information for an issuer's name filter, see [Figure 54 on page 574](#).

### Subject's name filter

A subject's name filter can contain a full or partial subject's distinguished name. Using the directory information shown in [Figure 52 on page 572](#), suppose we want to associate all users in the New York office with the user ID NYUSER, and all users in the New York sales department with user ID NYSALES. We will create two subject's name filters, based on the following significant portions of the subject's distinguished names:

```
OU=Sales.OU=New York.OU=US.O=World Sales Corp
```

```
OU=New York.OU=US.O=World Sales Corp
```

### Examples

The RACDCERT MAP commands shown in [Figure 55 on page 574](#) create two subject's name filters based on partial subject's distinguished names.

```

RACDCERT ID(NYSALES) MAP WITHLABEL('NY SALES REPS') TRUST
SDNFILTER('OU=Sales.OU=New York.OU=US.O=World Sales Corp')
RACDCERT ID(NYUSER) MAP WITHLABEL('NY OTHERS') TRUST
SDNFILTER('OU=New York.OU=US.O=World Sales Corp')
SETROPTS RACLIST(DIGTNMAP) REFRESH

```

Figure 55. Sample RACDCERT MAP commands for creating subject's name filters

The filter labeled 'NY SALES REPS' contains the portion of the subject's distinguished name that identifies the user as an employee of the Sales department in the New York office of the US division of the World Sales Corporation. Based on this filter, RACF will associate the user ID NYSALES to any user presenting a certificate containing this significant portion of the subject's distinguished name, who does not have an individual certificate registered with RACF.

The filter labeled 'NY OTHERS' contains the portion of the subject's distinguished name that identifies the user as an employee in the New York office of the US division of the World Sales Corporation. Based on this filter, RACF will associate the user ID NYUSER to any user presenting a certificate containing

this significant portion of the subject's distinguished name, who does not have an individual certificate registered with RACF.

Users that present certificates that contain subject's distinguished names that match *both* filters will be associated with the user ID of the *most specific* filter. In this case, the most specific filter is the filter labeled 'NY SALES REPS'. For example, if the users Agneta and Hiro, whose certificate information is shown in [Table 37 on page 571](#), present certificates while these two subject's name filters are in effect, the following will result:

1. Agneta will be associated with the user ID NYSALES, based on the filter labeled 'NY SALES REPS'.
2. Hiro will be associated with the user ID NYUSER, based on the filter labeled 'NY OTHERS'.

**Note:** If either Agneta or Hiro had individual certificates registered to RACF, they would have been assigned the user ID specified when the certificates were registered.

### ***Details about processing subject's name filters***

Using the previous example, Hiro presents a certificate that is not registered with RACF. The following represents the sequence of processing that RACF, specifically the `initACEE` callable service, will complete to process a subject's name filter.

1. The sequence shown in “[How RACF processes certificate name filters](#)” on [page 577](#) is followed, until the full subject's name is used to check for a matching profile in the DIGTNMAP class, to determine if there is an applicable certificate name filter.

**Result:** No DIGTNMAP profile is found to match:

CN=Hiro Ogura.OU=Admin.OU=New York.OU=US.O=World Sales Corp

2. A partial subject's name is formed by removing the most specific node (the CN node) and used to check for a matching profile in the DIGTNMAP class.

**Result:** No DIGTNMAP profile is found to match:

OU=Admin.OU=New York.OU=US.O=World Sales Corp

3. The next partial subject's name is formed by removing the next most specific node (OU=Admin) and used to check for a matching profile in the DIGTNMAP class.

**Result:** A DIGTNMAP profile is found to match:

OU=New York.OU=US.O=World Sales Corp

4. Processing by `initACEE` continues using the user ID NYUSER for Hiro's certificate.

### **Subject's and issuer's name filter**

A subject's and issuer's name filter contains a combination of a full or partial subject's distinguished name, and the full issuer's distinguished name. These filters can be used when either the subject's name alone or the issuer's name alone does not provide enough information to associate a certificate with a user ID. This happens when two different certificate authorities issue certificates for the same subject name, or, most commonly, when one certificate authority issues certificates for many different subject names.

A subject's and issuer's name filter can contain the full subject's name, including the CN node, and the full issuer's name. In such a case, you can consider registering the certificate that contains these full names using the `RACDCERT ADD` command. However, if you register the certificate, RACF will store the certificate as a DIGTCERT profile and you will need to take action when the certificate expires to remove or replace it.

Using the directory information shown in [Figure 52 on page 572](#), suppose we add another filter to our previously defined name filters. This filter will associate all users in the Administration department of the New York office with the user ID NYADMIN. We will create a subject's and issuer's name filter, based on the following significant portion of the subject name, and the full issuer's name:

OU=Admin.OU=New York.OU=US.O=World Sales Corp



```
OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
```

### Example

The RACDCERT MAP command shown in Figure 56 on page 576 creates a subject's and issuer's name filter based on the partial subject's distinguished name and the full issuer's name.

```
RACDCERT ID(NYADMIN) MAP WITHLABEL('NY ADMIN') TRUST
SDNFILTER('OU=Admin.OU=New York.OU=US.O=World Sales Corp')
IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')
SETROPTS RACLIST(DIGTNMAP) REFRESH
```

Figure 56. Sample RACDCERT MAP command for creating a subject's and issuer's name filter

This filter contains the portion of the subject's distinguished name that identifies the user as an employee of the Administration department in the New York office of the US division of the World Sales Corporation, and the full issuer's distinguished name that identifies the issuer as VeriSign Class 1. Based on this filter, RACF will associate the user ID NYADMIN to any user presenting a certificate issued by VeriSign Class 1 containing this significant portion of the subject's distinguished name, who does not have an individual certificate registered with RACF.

Therefore, if the users Timo and Hiro, whose certificate information is shown in Table 37 on page 571, present certificates while all defined name filters are in effect, the following will result:

1. Hiro will be associated with the user ID NYADMIN, based on the filter labeled 'NY ADMIN'.
2. Timo will be associated with the user ID WEBUSER, based on the filter labeled 'INTERNET OTHERS'.

**Note:** If either Hiro or Timo had individual certificates registered to RACF, they would have been assigned the user ID specified when the certificates were registered.

### Details for processing subject's and issuer's name filters

Timo presents a digital certificate that is not registered with RACF. The following represents the sequence of processing that RACF, specifically the initACEE callable service, will complete in order to process full and partial subject's names.

1. The sequence shown in “How RACF processes certificate name filters” on page 577 is followed, until the full subject's name and issuer's name is used to check for a matching profile in the DIGTNMAP class, to determine if there is an applicable certificate name filter.

**Result:** No DIGTNMAP profile is found to match:

```
CN=Timo Kokkonen.OU=Sales.OU=Los Angeles.OU=US.O=World Sales Corp |
OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
```

2. A partial subject's name is formed by removing the most specific node (the CN node) and used to check for a matching profile in the DIGTNMAP class.

**Result:** No DIGTNMAP profile is found to match:

```
OU=Sales.OU=Los Angeles.OU=US.O=World Sales Corp | OU=VeriSign Class 1
Individual Subscriber.O=VeriSign, Inc.L=Internet
```

3. The next partial subject's name is formed by removing the next most specific node (OU=Sales) and used to check for a matching profile in the DIGTNMAP class.

**Result:** No DIGTNMAP profile is found to match:

```
OU=Los Angeles.OU=US.O=World Sales Corp | OU=VeriSign Class 1 Individual
Subscriber.O=VeriSign, Inc.L=Internet
```

4. The next partial subject's name is formed by removing the next most specific node (OU=Los Angeles) and used to check for a matching profile in the DIGTNMAP class.

**Result:** No DIGTNMAP profile is found to match:

```
OU=US.O=World Sales Corp | OU=VeriSign Class 1 Individual
Subscriber.O=VeriSign, Inc.L=Internet
```



5. The last partial subject's name is formed by removing the next most specific node (OU=US) and used to check for a matching profile in the DIGTNMAP class.

**Result:** No DIGTNMAP profile is found to match:

```
O=World Sales Corp | OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
```

6. The full issuer's name is then used to check for a matching profile in the DIGTNMAP class.

**Result:** A DIGTNMAP profile is found to match:

```
OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
```

7. Processing by initACEE continues using the user ID WEBUSER for the Timo's certificate.

## How RACF processes certificate name filters

When a user presents a digital certificate as identification and the initACEE callable service is called to associate the certificate with a user ID, initACEE first searches the DIGTCERT class using the certificate's serial number and issuer's distinguished name to see if the certificate was previously registered to RACF. If no match is found in the DIGTCERT class, initACEE attempts to locate an appropriate certificate name filter by searching the DIGTNMAP class using a series of full and partial distinguished names until the most specific matching filter is found. If no match is found, and the certificate does not contain a hostIdMappings extension (see [“Using a hostIdMappings extension” on page 605](#)), the certificate cannot be used to identity the user to RACF.

The following values are used in sequence to search for a matching certificate name filter:

1. *subject's-full-name.issuer's-full-name*
2. *subject's-partial-name.issuer's-full-name*
3. *subject's-full-name*
4. *subject's-partial-name*
5. *issuer's-full-name*
6. *issuer's-partial-name*

As soon as a matching certificate name filter is found, the user ID associated with the filter is used to identify the user of the certificate. Note that searching is not done for the following values:

```
subject's-full-name.issuer's-partial-name
subject's-partial-name.issuer's-partial-name
```

Each step of the search using a partial name might actually involve a series of searches for partial name values based on the full name. Each partial name value in the series is determined by removing the next most specific node in the name. For details on searching for a series of partial name values, see the next example using Timo's certificate.

## Using an existing certificate as a model

An existing digital certificate can be used as a model for a certificate name filter, if it is available in a cataloged data set. Using the RACDCERT MAP command with the MAP(*data-set-name*) option, a stored certificate can be used to model the subject's name filter, the issuer's name filter, or both. The subject's distinguished name in the certificate is used beginning with the value specified with the SDNFILTER. The issuer's distinguished name in the certificate is used beginning with the value specified with the IDNFILTER.

For example, let's assume that Ines Soto's certificate is available in data set 'CERTADM.SOTO', and that it contains the following subject's and issuer's names:

```
CN=Ines Soto.OU=Admin.OU=New York.OU=US.O=World Sales Corp
OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet
```

The RACDCERT MAP commands shown in [Figure 57 on page 578](#) can be used to create certificate name filters using Ines Soto's certificate as a model. Note that only the starting point for each filter needs to be specified to indicate where the filter name should begin.

```
RACDCERT ID(WEBUSER) MAP('CERTADM.SOTO') WITHLABEL('INTERNET OTHERS')
IDNFILTER('OU=') TRUST
RACDCERT ID(NYUSER) MAP('CERTADM.SOTO') WITHLABEL('NY OTHERS')
SDNFILTER('OU=N') TRUST
RACDCERT ID(NYADMIN) MAP('CERTADM.SOTO') WITHLABEL('NY SALES REPS')
SDNFILTER('OU=') IDNFILTER('OU=') TRUST
SETOPTS RACLIST(DIGTNMAP) REFRESH
```

*Figure 57. Sample RACDCERT MAP commands using a model certificate*

The RACDCERT MAP commands in [Figure 58 on page 578](#) can be used to create the same certificate name filters as those created by the RACDCERT MAP commands in [Figure 57 on page 578](#). Note that the RACDCERT commands in [Figure 57 on page 578](#) using the model certificate are shorter and might minimize typographic errors when defining long filter names.

```
RACDCERT ID(WEBUSER) MAP WITHLABEL('INTERNET OTHERS') TRUST
IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')
RACDCERT ID(NYUSER) MAP WITHLABEL('NY OTHERS') TRUST
SDNFILTER('OU=New York.OU=US.O=World Sales Corp')
RACDCERT ID(NYADMIN) MAP WITHLABEL('NY ADMIN') TRUST
SDNFILTER('OU=Admin.OU=New York.OU=US.O=World Sales Corp')
IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')
SETOPTS RACLIST(DIGTNMAP) REFRESH
```

*Figure 58. Sample RACDCERT MAP commands not using a model certificate*

## Excluding a certificate by using the NOTRUST option

You can use certificate name filtering to prevent a digital certificate from being associated with a user ID by defining a name filter with the NOTRUST option, as long as it is the most specific filter that matches the certificate you want to exclude. Certificate name filters defined with the NOTRUST option are not used to associate a user ID to a certificate. The NOTRUST option can be used to exclude one or more certificates.

The RACDCERT MAP command in [Figure 59 on page 578](#) defines a fully distinguished subject's and issuer's name filter labeled 'NOT FRANS' with the NOTRUST option to prevent a certificate from being mapped to the NYADMIN user ID.

```
RACDCERT ID(NYADMIN) MAP WITHLABEL('NOT FRANS') NOTRUST
SDNFILTER('CN=Frans De Graaff.OU=Admin.OU=New York.OU=US.O=World Sales Corp')
IDNFILTER('OU=VeriSign Class 1 Individual Subscriber.O=VeriSign, Inc.L=Internet')
SETOPTS RACLIST(DIGTNMAP) REFRESH
```

*Figure 59. Sample RACDCERT MAP command using the NOTRUST option*

## Mapping multiple user IDs using additional criteria

You might need to assign more than one user ID to a certificate, based on the particular circumstances in which the certificate is presented. Such circumstances might include the following:

- The user of the certificate needs access to more than one application, and each application requires a different user ID.
- The same application might run on more than one system, and each system requires a different user ID.

Certificate name filtering allows you to associate more than one user ID to a certificate using additional criteria, such as APPLID and SYSID. Other criteria, such as SSL encryption level, can be used if this information passed with the certificate by the caller of the `initACEE` callable service. For information about passing additional criteria to `initACEE`, see [z/OS Security Server RACF Callable Services](#).

You specify multiple user IDs for a filter using the RACDCERT MAP command with the MULTIID option, and creating one general resource profile in the DIGTCRIT class for each user ID you want to associate with the filter. The name of the DIGTCRIT profile consists of one or more criteria values. The user ID is specified as the APPLDATA value. When you use RACDCERT MAP with the MULTIID option, you do

not specify a user ID. Instead, you use the CRITERIA option of RACDCERT MAP to specify one or more variable names that correspond to values in the DIGTCRIT profile names. Therefore, each MULTIID filter is associated with profiles in the DIGTCRIT class instead of a user ID.

## RACLISTing the DIGTCRIT class

**Guideline:** Activate and RACLIST the DIGTCRIT class for improved performance when you specify additional criteria.

**Example:**

```
SETOPTS RACLIST(DIGTCRIT) CLASSACT(DIGTCRIT)
```

Any RACLISTed criteria that you alter or delete will not reflect your changes until you refresh the DIGTCRIT class. This is because RACF uses RACLISTed profiles before profiles in the RACF database. Therefore, to make your changes effective, refresh the DIGTCRIT class.

**Example:**

```
SETOPTS RACLIST(DIGTCRIT) REFRESH
```

## Using application criteria

When users have only one certificate but need to connect to multiple applications that require different user IDs, you can assign user IDs based on the application identifier (APPLID).

### Example

Michael's Music Company has two Web-based applications: an online royalties application, and an online inventory application. The company has contracted VeriSign to issue certificates to its users, one certificate for each user. When one of the company's users connects to the royalties application, the user's certificate should be assigned the ROYALID user ID. When one of the company's users connects to the inventory application, the user's certificate should be assigned the INVID user ID.

The RACDCERT MAP and RDEFINE commands shown in [Figure 60 on page 579](#) create a full issuer's name filter that maps these two user IDs based on the application being accessed by the user of the certificate. The RACDCERT command uses the MULTIID option to specify additional criteria contained in the DIGTCRIT class using the predefined variable &APPLID. The RDEFINE commands create two profiles in the DIGTCRIT class that associate each APPLID value with the user ID indicated by the APPLDATA value.

```
RACDCERT MULTIID MAP WITHLABEL('All Michael's Music Employees') TRUST
      IDNFILTER('OU=Michael's Music General Subscriber.O=VeriSign,
      Inc.L=Internet')
      CRITERIA(APPLID=&APPLID)
SETOPTS RACLIST(DIGTCRIT) REFRESH

RDEFINE DIGTCRIT APPLID=EROYAL APPLDATA(ROYALID)
RDEFINE DIGTCRIT APPLID=EINV APPLDATA(INVID)
SETOPTS RACLIST(DIGTCRIT) REFRESH
```

*Figure 60. Sample RACDCERT MAP and RDEFINE commands for mapping multiple user IDs*

You can display mapping information for a MULTIID filter using the RACDCERT LISTMAP command with the LABEL option. For example:

```
RACDCERT MULTIID LISTMAP(LABEL('All Michael's Music Employees'))
```

[Figure 61 on page 580](#) shows sample output based on this RACDCERT LISTMAP command.

```
Mapping information for MULTIID:
Label: All Michael's Music Employees
Status: TRUST
Issuer's Name Filter:
    >OU=Michael's Music General Subscriber.O=VeriSign, Inc.L=Internet<
Subject's Name Filter:
    ><
Criteria:
    APPLID=&APPLID
```

Figure 61. Sample output from the LISTMAP command for a MULTIID filter

For details about using the RACDCERT MAP command with the MULTIID option, RACDCERT LISTMAP, and the RDEFINE command, see [z/OS Security Server RACF Command Language Reference](#).

If a user certificate is used for additional applications and should be associated with a user ID for these applications, you can create a generic DIGTCRIT profile named APPLID=\* to cover all other applications. For example, the addition of the following DIGTCRIT profile to the MULTIID filter created in Figure 60 on page 579 specifies that the ALLAPPS user ID should be associated with all certificates used to access all other applications.

```
SETRPTS GENERIC(DIGTCRIT)
RDEFINE DIGTCRIT APPLID=* APPLDATA(ALLAPPS)
SETRPTS RACLIST(DIGTCRIT) REFRESH
```

**Note:** If the caller of the `initACEE` callable service does not specify the `APPLID` variable, only the `APPLID=*` profile in the DIGTCRIT class will be used to determine the RACF user ID.

## Using system criteria

When users have only one certificate but need to connect to multiple systems that require different user IDs, you can assign user IDs based on the system identifier (SYSID). The SYSID is the 4-character SID value specified in the SMFPRMxx member of SYS1.PARMLIB on each system. You specify system criteria using the predefined variable `&SYSID` with the CRITERIA option of the RACDCERT MAP command for MULTIID filters. You must use the RDEFINE command to create profiles in the DIGTCRIT class to associate each SYSID value with the user ID indicated by the APPLDATA value.

## Using multiple criteria

You can use multiple additional criteria with your certificate name filters by specifying multiple values with the CRITERIA option of the RACDCERT MAP command.

### Example

Jamal's Bank has contracted with VeriSign to provide certificates to its customers and its account representatives. Both customers and account representatives access the company's systems through SSL. Customer SSL connections go through system A (SYSID=SYSA) and are only allowed access to general information about the company's offerings. Account representatives connect through system B (SYSID=SYSB) and need access to confidential customer information. Both systems A and B share the RACF database.

The application that serves the company's data invokes `initACEE` and passes user certificates with information about the SSL encryption level used by each user to connect to the system. This information is passed to `initACEE` as a variable called `ENCRLVL`, and the following values are assigned by the application based on the SSL encryption strength of the connection:

#### HIGH

SSL encryption strength using at least 128-bit encryption

#### LOW

SSL encryption strength using 40-bit encryption

The RACDCERT MAP and DIGTCRIT commands shown in Figure 62 on page 581 set up an issuer's name filter that uses multiple user IDs based on SYSID and ENCRLVL. In this example, there is a certificate available for use as a model in data set 'JAMALDC'. The certificate contains the following issuer's name.

```
OU=Jamal's Bank General Subscriber.O=VeriSign, Inc.L=Internet
```

```
RACDCERT MULTIID MAP('JAMALDC') WITHLABEL('All Jamal's Users')
      IDNFILTER('OU') CRITERIA(SYSID=&SYSID.ENCRLVL=&ENCRLVL)
SETROPTS RACLIST(DIGTNMAP) REFRESH

SETROPTS GENERIC(DIGTCRIT)
RDEFINE DIGTCRIT SYSID=SYSB.ENCRLVL=HIGH APPLDATA('ACCTREP')
RDEFINE DIGTCRIT SYSID=SYSB.ENCRLVL=* APPLDATA('GENERAL')
RDEFINE DIGTCRIT SYSID=SYSA.ENCRLVL=* APPLDATA('GENERAL')
SETROPTS RACLIST(DIGTCRIT) REFRESH
```

Figure 62. Sample RACDCERT MAP and RDEFINE commands using multiple criteria

The issuer's name filter created in Figure 62 on page 581 associates the following user IDs:

#### GENERAL

For all customers, and account representatives connecting with low-strength encryption.

#### ACCTREP

For account representatives connecting with high-strength encryption.

### Details for processing an issuer's name filter with multiple criteria

For example, if a customer accesses the Jamal's Bank system using an unregistered user certificate, the following represents the sequence of processing that RACF, specifically the `initACEE` callable service, will complete to process multiple criteria using a DIGTCRIT profile.

1. The sequence shown in “How RACF processes certificate name filters” on page 577 is followed, until the full issuer's name is used to check for a matching profile in the DIGTNMAP class, to determine if there is an applicable certificate name filter.

**Result:** A DIGTNMAP profile is found to match:

```
OU=Jamal's Bank General Subscriber.O=VeriSign, Inc.L=Internet
```

2. The criteria definitions, `SYSID=&SYSID.ENCRLVL=&ENCRLVL` are found in the DIGTNMAP profile, and the supplied values are substituted for each variable: `SYSID=SYSA` and `ENCRLVL=LOW`.

**Result:** A DIGTCRIT profile is found to match:

```
SYSID=SYSA.ENCRLVL=*
```

3. Processing by `initACEE` continues using the user ID GENERAL for the customer's certificate.

**Note:** In this example, if the application calling the `initACEE` callable service does not pass the `ENCRLVL` variable, only the `SYSID=` value is used to determine the user ID. Therefore, the DIGTCRIT profile named `SYSID=SYSA.ENCRLVL=*` is found to match, and the user ID GENERAL is still used for the customer's certificate.

### Activating additional criteria

When you create a certificate name filter using the MULTIID option of the RACDCERT MAP command, you must create corresponding profiles in the DIGTCRIT class to specify the multiple user IDs associated with the filter. The DIGTCRIT class must be active and SETROPTS RACLIST processing must be active for the DIGTCRIT class. Before creating any profiles in the DIGTCRIT class, you must issue the following command:

```
SETROPTS CLASSACT(DIGTCRIT) RACLIST(DIGTCRIT)
```

Once SETROPTS RACLIST processing is active for the DIGTCRIT class, you must refresh the DIGTCRIT class in order for new or changed profiles to take effect. After creating or changing a DIGTCRIT class profile, you must issue the following command:

```
SETROPTS RACLIST(DIGTCRIT) REFRESH
```

## Automatic registration of digital certificates

Your installation can provide a user interface to allow users to register their own digital certificates. You can provide an HTML Web page and CGI program accessed through WebSphere Application Server. (See the sample provided in 'SYS1.SAMPLIB' member RACINSTL.) The registration page can be used to prompt for registration of the user's certificate for his or her RACF user ID. When the user clicks on the registration box, a secure session is set up using SSL and the user's digital certificate. The user is prompted for his or her RACF user ID and password, which is passed from WebSphere Application Server to z/OS UNIX, then to RACF through the `initACEE` callable service (IRRSIA00) for registration. RACF verifies the user ID and password and creates an ACEE. Note that because the validity of the certificate is established when the SSL connection is set up, the DIGTCERT profile for this certificate is marked with the TRUST attribute.

See “[Registering user certificates](#)” on page 604 for details about using the registration function of the `initACEE` callable service.

## ICSF considerations for keys in the PKA key data set (PKDS)

Integrated Cryptographic Service Facility (ICSF) is a software element of z/OS that provides the application programming interfaces to the cryptographic hardware. ICSF provides hardware protection for the storage of the private keys associated with digital certificates and is a more secure solution than non-ICSF private key management. ICSF ensures that the private keys are encrypted under the ICSF master key and stored in the ICSF PKA key data set (PKDS). ICSF controls access to the private keys through the use of RACF general resources in the CSFKEYS and CSFSERV classes. In addition, operational performance is improved because ICSF uses a hardware cryptographic coprocessor.

If ICSF is implemented at your installation, you can use it to store private keys by specifying the PKDS option of the ADD, GENCERT and REKEY functions of the RACDCERT command. You can also use ICSF to generate private keys using the same options. ICSF supports generation and storage of RSA and ECC key types.

You can migrate a non-ICSF private key to the ICSF PKDS by issuing the RACDCERT ADD command function with certain options and specifying the name of the data set that contains the existing certificate. If the certificate data set is no longer available, you can recreate it using the RACDCERT EXPORT command.

For details about using the RACDCERT command, see *[z/OS Security Server RACF Command Language Reference](#)*.

## Using a PCI cryptographic coprocessor to generate private keys

ICSF can use a PCI class cryptographic coprocessor to generate RSA and ECC public/private key pairs. The PCI class of cryptographic coprocessors includes the PCI, PCI X, and the Crypto Express coprocessors. The PKDS option of the RACDCERT command detects when PCI hardware is available, and use it to generate a public/private key pair. A PCI class cryptographic coprocessor is used only when ICSF is active and configured to use it.

## Migrating an ICSF private key in the PKDS from one system to another

Private keys that are stored by RACF in the ICSF PKA key data set (PKDS), and private keys that are generated by ICSF on behalf of RACF, are always encrypted and cannot be recovered in a clear form. Therefore, certificates with such keys cannot be exported from RACF in PKCS #12 format. In general, this restricts your ability to migrate certificates and their private keys from one system to another and share

them among multiple systems. However, you can migrate a certificate and its ICSF private key when both the source and target systems are z/OS systems configured to use ICSF and both share the same ICSF PKA master key. The systems need not share the same RACF database nor share the same ICSF PKDS.

Using the following steps, you can generate a new certificate with a private ICSF key on system A (the source system) and replicate the same certificate and key on system B (the target system). In the RACDCERT command examples shown, the certificate you are migrating is associated with the user ID SYSMAN and the certificate label is 'SECURE.KEY'. The ICSF private key has the PKDS key label 'SECURE.KEY' and is generated by the PCI cryptographic coprocessor. On the target system, the label of the migrated certificate will be 'MIGRATED.KEY' and the label of its PKDS key will also be 'MIGRATED.KEY'.

For details about using the RACDCERT command, see [z/OS Security Server RACF Command Language Reference](#).

## Steps for migrating a certificate and its ICSF private key in the PKDS

### Before you begin:

- Both the source and target system must be configured to use ICSF and share the same ICSF PKA master key. The systems need not share the same RACF database nor share the same ICSF PKDS.
- A PCI-class cryptographic coprocessor must be operational and configured with the ICSF PKDS on each system (both the source and target system) when you specify the PKDS operand of the RACDCERT GENCERT command function. Otherwise, specify the ICSF operand.
- To use your installation's ICSF facilities in Steps “1” on page 583 and “6” on page 583, you might need additional authority to ICSF resources. For information about these resources, see [z/OS Cryptographic Services ICSF Administrator's Guide](#).
- To extract ICSF private keys, you will need a non-RACF utility, such as KEYXFER. To download the KEYXFER utility, go to the [z/OS UNIX System Services Tools and Toys download \(ftp://ftp.software.ibm.com/s390/zos/tools/keyxfer/\)](#).

Perform the following steps to generate a RACF certificate and its ICSF public/private key pair on system A (the source system), and migrate them to system B (the target system).

1. Generate the certificate and its public/private key pair on system A.

```
RACDCERT ID(SYSMAN) GENCERT SUBJECTSDN(CN('Secure Key'))
WITHLABEL('SECURE.KEY') PKDS(*) SIZE(2048)
```

2. Extract the certificate from RACF and store it in an MVS data set called 'MY.CERT'. (The ICSF private key is not extracted in this step.)

```
RACDCERT ID(SYSMAN) EXPORT(LABEL('SECURE.KEY')) DSN(MY.CERT) FORMAT(CERTDER)
```

3. Extract the encrypted private key from ICSF using a non-RACF utility, such as KEYXFER. Use the WRITE\_PKDS function of KEYXFER.

4. Transmit both the key and certificate data sets to system B. This step completes your work on system A.

5. Receive both the key and certificate data sets on system B.

6. Add the encrypted private key to ICSF using a non-RACF utility, such as KEYXFER, specifying the desired PKDS label for the key on system B, 'MIGRATED.KEY'. Use the READ\_PKDS function of KEYXFER.



- 
7. Add the certificate to RACF using the same RACF and PKDS label you used in Step “6” on page 583, 'MIGRATED.KEY'.

```
RACDCERT ID(SYSMAN) ADD(MY.CERT) WITHLABEL('MIGRATED.KEY') PKDS(*)
```

- 
8. List the migrated certificate to verify that RACF found the private key and assigned the private key to the certificate.

```
RACDCERT ID(SYSMAN) LIST(LABEL('MIGRATED.KEY'))
```

**Result:** You should see similar information at the end of the certificate listing:

```
Key Type: RSA
Key Size: 2048
Private Key: YES
PKDS Label: MIGRATED.KEY
Ring Associations:
*** No rings associated ***
```

---

You have now generated a certificate and its ICSF public/private key pair on system A and migrated them to system B. Both system A and system B can now use the same certificate and key pair.

## The *irrcerta*, *irmulti*, and *irrsitec* user IDs

The *irrcerta*, *irmulti*, and *irrsitec* user IDs are defined in USER profiles that are supplied with RACF and cannot be defined by your installation. They are used to anchor certain profiles in the DIGTCERT and DIGTNMAP class that are not associated with individual user IDs, and cannot be used for any other purpose.

- User certificates that you add using the RACDCERT ADD command with the CERTAUTH option are automatically associated with the user ID *irrcerta*.
- User certificates that you add using the RACDCERT ADD command with the SITE option are automatically associated with the RACF user ID *irrsitec*.
- Certificate name filters that you add using the RACDCERT MAP command with the MULTIID option are automatically associated with the RACF user ID *irmulti*.

The use of these user IDs in DIGTCERT and DIGTNMAP profiles is automatic and cannot be changed using RACF commands. These user IDs cannot be administered using the ADDUSER, ALTUSER, DELUSER, LISTUSER and CONNECT commands. Since profiles that are associated with these user IDs contain lowercase characters, the SEARCH FILTER command is not intended for use in locating them and will deliver unpredictable results.

The *irrcerta*, *irmulti*, and *irrsitec* user ID profiles should not be deleted. They must exist at initialization time or RACF initialization will automatically add them. In addition, the remove ID utility (IRRRID00) will not create commands to process these user IDs. For more information about IRRRID00 and the processing of DIGTCERT and DIGTNMAP profiles, see “Finding residual IDs” on page 396.

## Renewing an expiring certificate

When a certificate approaches its expiration date, you can renew the certificate and continue using it. You can choose to renew the certificate using the same private key, thereby extending the life of the private key. Or you can retire the private key and replace it with a new private key (also called certificate *rekeying* or *key rollover*). The following procedures are shown as examples of common scenarios in which you will handle various types of expiring certificates by either renewing private keys or by replacing private keys.



## Renewing a certificate with the same private key

You might receive a notification from your certificate authority that a certificate is nearing its expiration date. When you renew a certificate using the same private key, you extend the life of the private key and all information in the expiring certificate is updated to reflect the renewal, including the key ring connection information. The following procedures outline the steps to renew an expiring personal certificate that was either issued by an external certificate authority, issued by a local certificate authority, or was self-signed within RACF.

### Steps for renewing a certificate issued by an external CA

Perform the following steps to renew an expiring certificate issued by an external certificate authority using the same private key.

1. Optionally, create a certificate request based on the expiring certificate and store it in an MVS data set 'SYSADM.CERT.REQ' by executing the following command:

```
RACDCERT ID(WEBSRV) GENREQ(LABEL('My Web Server Cert'))
DSN('SYSADM.CERT.REQ')
```

If your CA retains your original certificate signing requests (CSR), you might *not* need to create and store a new request based on the expiring certificate. You might be able to request a renewal using the original CSR.

2. Send the certificate request to the CA and receive the newly signed and reissued certificate back from the CA into MVS data set 'SYSADM.CERT.B64'.

**Restriction:** The certificate request data contained in the data set must be sent to, and received from, the external CA using the process defined by the CA. Those steps are not included.

3. Add the newly signed certificate into RACF under the same ID used in step 1 to replace the existing certificate by executing the following command:

```
RACDCERT ID(WEBSRV) ADD('SYSADM.CERT.B64')
```

You have now renewed a certificate that was issued by an external certificate authority using the same private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information.

### Steps for renewing a certificate issued by a local CA

Perform the following steps to renew an expiring certificate using the same private key when the certificate was generated by RACF and issued by a local certificate authority. The expiring certificate was signed by a CERTAUTH certificate labeled 'Local RACF CA'.

1. Create a certificate request based on the expiring certificate and store it in an MVS data set 'SYSADM.CERT.REQ' by executing the following command:

```
RACDCERT ID(WEBSRV) GENREQ(LABEL('My Web Server Cert'))
DSN('SYSADM.CERT.REQ')
```

2. Renew and replace the existing certificate by executing the following command:

```
RACDCERT ID(WEBSRV) GENCERT('SYSADM.CERT.REQ')
SIGNWITH(CERTAUTH LABEL('Local RACF CA'))
```

You have now renewed a certificate that was signed by a local certificate authority and you renewed it using the same private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information.

### Steps for renewing a self-signed certificate in RACF

Perform the following steps to renew an expiring self-signed certificate in RACF using the same private key. This example shows the commands a TSO user YELENA might issue to renew her self-signed certificate.

1. Create a certificate request based on the expiring certificate and store it in an MVS data set 'YELENA.CERT.REQ' by executing the following command:

```
RACDCERT ID(YELENA) GENREQ(LABEL('Yelena's Cert'))  
DSN('YELENA.CERT.REQ')
```

2. Execute the following command to renew and replace the expiring self-signed certificate:

```
RACDCERT ID(YELENA) GENCERT('YELENA.CERT.REQ')  
SIGNWITH(LABEL('Yelena's Cert'))
```

You have now renewed an expiring self-signed certificate using the same private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information.

### Renewing (rekeying) a certificate with a new private key

When you renew a certificate using a new private key, you retire the private key and replace it with a new one. This process is commonly called certificate *rekeying* or *key rollover*. You choose this option to prevent a private key from being overused. (The more a key is used, the more susceptible it is to being broken and recovered by an unintended party.)

All information in the renewed certificate is updated to reflect the renewal, including the key ring connection information. Once you retire and replace the old certificate, you can now begin to use the new certificate and its private key. You can continue to use the old, retired certificate until it expires to verify previously generated signatures. However, you cannot use the retired certificate to sign new certificates. Additionally, do not connect the retired certificate to any key rings as the default certificate.

When you rekey and rollover a private key, you use the REKEY and ROLLOVER operands of the RACDCERT command. The REKEY operand makes a self-signed copy of the original certificate with a new public-private key pair. The ROLLOVER operand finalizes the rekey operation by replacing the use of the original certificate with the new certificate in every key ring to which the original certificate is connected. It also destroys the original private key and copies over the information about its serial number base in case the certificate was being used to sign new certificates.

For more information, see [RACDCERT ROLLOVER \(Rollover certificate\)](#) in *z/OS Security Server RACF Command Language Reference*.

In the following procedures, the expiring certificate was either issued by an external certificate authority, issued by a local certificate authority, or was self-signed within RACF. In each case, you will replace the private key with a new one.

### Steps for rekeying a certificate issued by an external CA

In this procedure, you are renewing a CERTAUTH certificate with label 'Local PKI CA'. It was issued by a commercial CA and is being used by PKI Services for the PKI templates as a certificate authority (CA) certificate, making the PKI Services CA a subordinate CA. The PCI cryptographic coprocessor will be used to generate the new key pair. The size of the new private key will be 2048 bits (RACF default size).

Perform the following steps to rekey a certificate issued by an external certificate authority using a new private key.

1. Initiate® the rekeying by executing the following RACF command:

```
RACDCERT CERTAUTH REKEY(LABEL('Local PKI CA'))
WITHLABEL('Local PKI CA-2') PKDS
```

2. Create a request for an external CA to sign the new public key and reissue the certificate. Create the request for the new key and store it in MVS data set 'SYSADM.CERT.REQ' by executing the following command:

```
RACDCERT CERTAUTH GENREQ(LABEL('Local PKI CA-2')) DSN('SYSADM.CERT.REQ')
```

3. Send the certificate request to the CA and receive the newly signed and reissued certificate back from the CA into MVS data set 'SYSADM.CERT.B64'.

**Restriction:** The certificate request data contained in the data set must be sent to, and received from, the external CA using the process defined by the CA. Those steps are not included.

4. Add the newly signed certificate into RACF under CERTAUTH which is used in step 1 to replace the self-signed rekeyed certificate by executing the following command:

```
RACDCERT CERTAUTH ADD('SYSADM.CERT.B64')
```

5. You are now ready to retire the original certificate and must stop all use of the original private key. If you are rekeying the PKI Services CA certificate, stop the PKI Services daemon.

At this point, the original certificate and its private key exist in RACF with label 'Local PKI CA'. The new certificate and its private key exist in a separate entry in RACF with label 'Local PKI CA-2'. You can proceed to rollover the key.

6. Finalize the rollover by entering the following command:

```
RACDCERT CERTAUTH ROLLOVER(LABEL('Local PKI CA'))
NEWLABEL('Local PKI CA-2')
```

7. (Optional) If you want to keep the original certificate label, you may use the following two commands:

```
RACDCERT CERTAUTH ALTER(LABEL('Local PKI CA')) NEWLABEL('Local PKI CA-1')
RACDCERT CERTAUTH ALTER(LABEL('Local PKI CA-2')) NEWLABEL('Local PKI CA')
```

8. If you rekeyed the PKI Services CA certificate for the PKI templates, restart the PKI Services daemon.

You have now rekeyed a certificate that was issued by an external certificate authority, using a new private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information. You have retired and replaced the old certificate. You can now begin to use the new certificate and its private key. You can continue to use the old certificate for signature verification purposes until it expires. However, you cannot use the old certificate to sign new certificates. Additionally, do not connect the old certificate to any key rings, as the default certificate.

## Steps for rekeying a certificate issued by a local CA

In this procedure, you are rekeying the certificate associated with the user ID FTPSRV with label 'My FTP Server Cert'. The certificate was issued by a CERTAUTH certificate with label 'Local RACF CA' that was generated by RACF.

Perform the following steps to rekey a certificate issued by a local CA and replace the private key.

1. Initiate the rekeying by executing the following RACF command:

```
RACDCERT ID(FTPSRV) REKEY(LABEL('My FTP Server Cert'))
WITHLABEL('My FTP Server Cert-2')
```

2. Create a certificate request based on the new self-signed certificate and store it in an MVS data set 'SYSADM.CERT.REQ' by executing the following command:

```
RACDCERT ID(FTPSRV) GENREQ(LABEL('My FTP Server Cert-2'))
DSN('SYSADM.CERT.REQ')
```

3. Sign the new certificate by executing the following command:

```
RACDCERT ID(FTPSRV) GENCERT('SYSADM.CERT.REQ')
SIGNWITH(CERTAUTH LABEL('Local RACF CA'))
```

4. You are now ready to retire the original certificate and must stop all use of the original private key.

At this point, the original certificate and its private key exist in RACF with label 'My FTP Server Cert'. The new certificate and its private key exist in a separate entry in RACF with label 'My FTP Server Cert-2'. You can now proceed to rollover the key.

5. Finalize the rollover by executing the following command:

```
RACDCERT ID(FTPSRV) ROLLOVER(LABEL('My FTP Server Cert'))
NEWLABEL('My FTP Server Cert-2')
```

6. (Optional) If you want to keep the original certificate label, you may use the following two commands:

```
RACDCERT ID(FTPSRV) ALTER(LABEL('My FTP Server Cert')) NEWLABEL('My FTP Server Cert-1')
RACDCERT ID(FTPSRV) ALTER(LABEL('My FTP Server Cert-2')) NEWLABEL('My FTP Server Cert')
```

You have now renewed a certificate that was signed by a local certificate authority and you renewed it using a new private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information. You have retired and replaced the old certificate. You can now begin to use the new certificate and its private key. You can continue to use the old certificate for signature verification purposes until it expires. However, you cannot use the old certificate to sign new certificates. Additionally, do not connect the old certificate to any key rings, as the default certificate.

## Steps for rekeying a self-signed certificate in RACF

In this procedure, you are rekeying a self-signed certificate in RACF that is associated with the user ID FTPSRV and is labeled 'My FTP Server Cert'.

Perform the following steps to renew an expiring self-signed certificate in RACF by replacing the private key.

1. Initiate the rekeying by executing the following RACF command:

```
RACDCERT ID(FTPSRV) REKEY(LABEL('My FTP Server Cert'))
WITHLABEL('My FTP Server Cert-2')
```

2. You are now ready to retire the original certificate and must stop all use of the original private key.

At this point, the original certificate and its private key exist in RACF with label 'My FTP Server Cert'. The new certificate and its private key exist in a separate entry in RACF with label 'My FTP Server Cert-2'. You can now proceed to rollover the key.

3. Finalize the rollover by entering the following command:

```
RACDCERT ID(FTPSRV) ROLLOVER(LABEL('My FTP Server Cert'))
NEWLABEL('My FTP Server Cert-2')
```

4. (Optional) If you want to keep the original certificate label, you may use the following two commands:

```
RACDCERT ID(FTPSRV) ALTER(LABEL('My FTP Server Cert')) NEWLABEL('My FTP Server Cert-1')
RACDCERT ID(FTPSRV) ALTER(LABEL('My FTP Server Cert-2')) NEWLABEL('My FTP Server Cert')
```

You have now renewed an expiring self-signed certificate using a new private key. All information in the certificate is updated to reflect the renewal, including the key ring connection information. You have retired and replaced the old certificate. You can now begin to use the new certificate and its private key. You can continue to use the old certificate for signature verification purposes until it expires. However, you cannot use the old certificate to sign new certificates. Additionally, do not connect the old certificate to any key rings, as the default certificate.

## Supplied digital certificates

For your convenience, RACF supplies certificates for some certificate authorities so you do not need to define them yourself. These certificates are not connected to any key ring and are defined as untrusted. This means that they are not used for authenticating these certificate authorities until you decide to use them.

**Restriction:** Do not delete the supplied certificates. If they do not exist at IPL time, RACF initialization automatically adds them. Therefore, if you delete them, they are recreated at the next IPL.

Each certificate is identified in RACF by its RACF label (a 32-character name). See Appendix C, “Listings of RACF supplied certificates,” on page 709 for a list of the certificates using their complete names as issued by the certificate authority, and the label for each supplied certificate.

## Steps to begin using a supplied CA certificate

Perform the following steps to begin using a supplied certificate-authority certificate.

For additional steps to begin using the STG Code Signing Certificate Authority (for RACF program signature verification for release z/OS V2R1 and earlier), see “Steps for preparing RACF to verify signed programs (one-time setup)” on page 340.

1. Determine which of the supplied certificates you want to use.

You can issue the following command to view the current certificate information listing for all certificate-authority certificates on your system, or see Appendix C, “Listings of RACF supplied certificates,” on page 709 for a listing of each supplied certificate.

**Example:**

```
RACDCERT CERTAUTH LIST
```

2. Modify each certificate to add the TRUST attribute.

**Example:**

```
RACDCERT CERTAUTH ALTER(LABEL('GeoTrust Global CA')) TRUST
```

---

3. Add a key ring for your server application, such as your Web server.

**Example:**

```
RACDCERT ADDRING(SSLring) ID(WEBSESV)
```

---

4. Add each of your selected certificates to the key ring.

**Example:**

```
RACDCERT ID(WEBSESV) CONNECT(CERTAUTH LABEL('GeoTrust Global CA'))
```

Repeat this step for each certificate you want your server to accept.

---

5. Unless already done, generate or acquire a certificate and private key for your server. Certificates can be generated using a product such as z/OS Security Server PKI Services, or by RACF using the RACDCERT GENCERT command.
- 

## Implementation scenarios

---

### Scenario 1: Secure server with a certificate signed by a certificate authority

Secure servers require the ability to retrieve the certificate that is associated with a particular server, along with the ability to perform operations with the private key of the server, such as establishing an SSL session. Consider a secure server, which has the distinguished name of OU=Inventory, O=XYZZY, C=US and a domain name of xyzzy.com and an alternate domain name of abc123.com. This server runs on z/OS with the user ID INVSESV. The steps to implement a server certificate are:

1. Generate a self-signed certificate for the server. This certificate is associated with the user ID that is associated with the secure server.

```
RACDCERT ID(INVSESV)
          GENCERT
          SUBJECTSDN(CN('xyzzy.com')
                     OU('Inventory')
                     O('XYZZY')
                     C('US'))
          ALTNAME(ADOMAIN('xyzzy.com','abc123.com'))
          WITHLABEL('Inventory Server')
```

**Notes:**

- a. Some SSL applications require that the common name (CN) is equal to the domain name.
- b. It is possible to issue RACF-managed certificates with multiple subject alternate names (SANs) of the same type, such as domain name as shown in this scenario. With this support, you can minimize the number of certificates that are needed to secure access to a server that can be accessed multiple ways. More information is provided in [z/OS Security Server RACF Command Language Reference](#). See the description of the RACDCERT GENCERT command, for the ALTNAME parameters AIP, ADOMAIN, AEMAIL, and AURI.

2. Create a certificate request to send to your chosen certificate authority. The certificate request is based on the certificate that was created in the previous step. Place this certificate into the data set 'MARKN.INVSERV.GENREQ'.

```
RACDCERT ID(INVSERV)
        GENREQ(LABEL('Inventory Server'))
        DSN('MARKN.INVSERV.GENREQ')
```

3. Send the certificate request to the certificate authority. The certificate request is in base64-encoded text. Typically, the request is sent to the certificate authority by using "copy and paste" to place the certificate request into an email that is sent to the certificate authority.

**Note:** RACF is not involved with this step.

4. The certificate authority validates the certificate. If the certificate is approved by the certificate authority, it is signed by the certificate authority, and returned to the requestor.

**Note:** RACF is not involved with this step.

5. Receive the returned certificate into a data set (for example, 'MARKN.INVSERV.CERT'). The returned certificate is in base64-encoded text. This can be done with "cut and paste", FTP, or other technique.

**Note:** RACF is not involved with this step.

6. Replace the self-signed certificate with the certificate signed by the certificate authority. The certificate is only replaced if the user ID that is specified as the ID value on the RACDCERT ADD command is the same user ID that was specified when the certificate was created. If the ID is not the same, then the certificate is added anew.

```
RACDCERT ID(INVSERV)
        ADD('MARKN.INVSERV.CERT')
        WITHLABEL('Inventory Server')
```

7. Connect the certificate to INVSERV's existing key ring and mark it as the default certificate.

```
RACDCERT ID(INVSERV)
        CONNECT(ID(INVSERV) LABEL('Inventory Server'))
        RING(RING01)
        DEFAULT)
```

8. Assuming that the chosen certificate authority certificate has already been added to RACF under CERTAUTH with the label of 'External Inventory CA', connect it to the key ring as well. This completes the certificate hierarchy from root to inventory server.

```
RACDCERT ID(INVSERV)
        CONNECT(CERTAUTH LABEL('External Inventory CA'))
        RING(RING01))
```

9. Give user INVSERV permission to read its own key ring by administering a profile in either the FACILITY or the RDATA LIB class.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(INVSERV) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB INV SERV.RING01.LST UACC(NONE)
PERMIT INV SERV.RING01.LST CLASS(RDATA LIB) ID(INV SERV) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

10. Configure the INV SERV software to use RING01 for SSL. For example, for z/OS HTTP Server, set the keyFile directive to KeyFile RING01 SAF.

**Note:** RACF is not involved with this step.

## Scenario 2: Secure server with a locally signed certificate

This is similar to “Scenario 1: Secure server with a certificate signed by a certificate authority” on page 590 with the exception that the certificate assigned to the secure server is a locally signed certificate rather than one signed by a certificate authority. Assume that the local certificate authority has the distinguished name of OU= 'Local Certificate Authority', O=XYZZY, C=US. The steps to implement a locally signed server certificate are:

1. Generate a self-signed certificate to represent the local certificate authority. This certificate is used as the certificate-authority certificate.

```
RACDCERT CERTAUTH
GENCERT
SUBJECTSDN(OU('Local Certificate Authority')
           O('XYZZY')
           C('US'))
KEYUSAGE(CERTSIGN)
WITHLABEL('XYZZY Local Certificate Authority')
```

2. Export the certificate to a data set, in this case 'MARKN.LOCCERTA.CERT'.

```
RACDCERT CERTAUTH
EXPORT(LABEL('XYZZY Local Certificate Authority'))
DSN('MARKN.LOCCERTA.CERT')
```

3. Place the certificate into the z/OS UNIX file system.

```
OPUT 'MARKN.LOCCERTA.CERT' '/u/loccerta/certauth.cacert'
```

**Note:** RACF is not involved with this step.

4. Configure WebSphere Application Server to recognize the file /u/loccerta/certauth.cacert as a certificate-authority MIME type.

**Note:** RACF is not involved with this step.

5. Each end user must point their browser to the z/OS UNIX file containing the certificate and run an acceptance dialog to allow the browser to accept the self-signed certificate. Each browser has its own mechanism for performing this step.

**Note:** RACF is not involved with this step.

6. Logon to the server user ID INV SERV and create a certificate for the server, signed with the certificate-authority certificate that was created in Step “1” on page 592.

```
RACDCERT ID(INV SERV)
GENCERT
SUBJECTSDN(CN('xyzzzy.com')
           OU('Inventory')
           O('XYZZY')
           C('US'))
WITHLABEL('Inventory Server')
```



```
SIGNWITH(CERTAUTH
        LABEL('XYZZY Local Certificate Authority'))
```

7. Connect the certificate to INVSERV's existing key ring and mark it as the default certificate.

```
RACDCERT ID(INVSERV)
        CONNECT(ID(INVSERV) LABEL('Inventory Server'))
        RING(RING01)
        DEFAULT)
```

8. Connect the local certificate authority certificate to the key ring as well. This completes the certificate hierarchy from root to inventory server.

```
RACDCERT ID(INVSERV)
        CONNECT(CERTAUTH LABEL('XYZZY Local Certificate Authority'))
        RING(RING01))
```

9. Give user INVSERV permission to read its own key ring by administering a profile in either the FACILITY or the RDATALIB class.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(INVSERV) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETOPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATALIB class:

```
RDEFINE RDATALIB INVSERV.RING01.LST UACC(NONE)
PERMIT INVSERV.RING01.LST CLASS(RDATALIB) ID(INVSERV) ACCESS(READ)
```

- If the RDATALIB class is not already active, activate and RACLIST it.

```
SETOPTS CLASSACT(RDATALIB) RACLIST(RDATALIB)
```

- If the RDATALIB class is already active and RACLISTed, refresh it.

```
SETOPTS RACLIST(RDATALIB) REFRESH
```

10. Configure INVSERV's software to use RING01 for SSL. For example, for z/OS HTTP Server, set the keyFile directive to KeyFile RING01 SAF.

**Note:** RACF is not involved with this step.

## Scenario 3: Migrating an ikeyman or gskkyman certificate

The installation needs to migrate their existing certificates on z/OS. These certificates were created with the ikeyman or gskkyman utility and reside in the z/OS UNIX file system. The steps to migrate these certificates are:

1. Using ikeyman or gskkyman, export the certificate from the ikeyman or gskkyman key database file as a PKCS #12 export file and place it into the z/OS UNIX file system.

**Note:** RACF is not involved with this step.

2. Export the file to an MVS data set, in this case MARKN.IMPORTED.CERT.

```
OGGET '/u/markn/cert.usercert' 'MARKN.IMPORTED.CERT' BINARY
```

3. Add the certificate to the RACF database and assign it a user. Assume that ikeyman or gskkyman encrypted the certificate with the password xyz.

```
RACDCERT ID(MARKN)
        ADD('MARKN.IMPORTED.CERT')
        WITHLABEL('Mark's Personal Certificate')
        TRUST
        PASSWORD('xyz')
```

Now you can use the certificate on z/OS as desired.

**Important:** Delete the ikeyman or gskkyman copy of the certificate so that the private key cannot be used inadvertently.

## Scenario 4: Secure server-to-server session enablement

A company wants to use two different secure servers for two different applications. The first application is for its internal employee data, which allows employees to read their own information, and allows designated employees to modify information. This server is called `internal_ss`. The company also has an external secure server, which is used by client applications running on the customer's systems, to order materials and check the status of orders. This server is called `external_ss`. The `internal_ss` server executes with a user ID of `INSS`, and accepts certificates only from the company's internal certificate authority, whose name is `ACME Local Certificate Authority`. The `external_ss` server executes with a user ID of `EXSS`, and accepts certificates from either the internal certificate authority called `ACME Local Certificate Authority` or the external certificate authority called `Really Big Certificate Authority`. Really Big Certificate Authority's certificate is in the data set `'REALBIG.CERTIF'`.

The commands to accomplish this are shown below. The authority checks that are shown assume that the person who issues these commands does not have `SPECIAL` authority, and is neither the user ID `INSS` or the user ID `EXSS`.

1. Create the user IDs for the secure servers.

```
ADDUSER INSS
ADDUSER EXSS
```

**Authority required:** `CLAUTH` for the `USER` class and `JOIN` in the default group to which they are connected.

2. Create the internal certificate authority.

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('ACME CA') O('ACME') C('US'))
        WITHLABEL('ACME')
```

**Authority required:**

`CONTROL` to the `IRR.DIGTCERT.GENCERT` and `IRR.DIGTCERT.ADD` resources in the `FACILITY` class,

or

`READ` to `IRR.DIGTCERT.CERTIFAUTH.ACME.UPD.GENCERT` in the `RDATA LIB` class.

3. Add the external certificate authority certificate.

```
RACDCERT CERTAUTH ADD('REALBIG.CERTIF') TRUST WITHLABEL('Really Big')
```

**Note:** Make sure the external certificate authority certificate is what you intend to add. Check its certificate fingerprint first by issuing `RACDCERT CHECKCERT('REALBIG.CERTIF')`.

**Authority required:**

`CONTROL` to the `IRR.DIGTCERT.ADD` resource in the `FACILITY` class,

or  
 READ to IRR.DIGTCERT.CERTIFAUTH.REALLY\_BIG.UPD.ADD  
 in the RDATA LIB class.

4. Generate the server certificates and the associated private keys. On platforms other than z/OS, this is performed using a facility such as `mkkf`, `ikeyman`, or equivalent. On z/OS, this is performed using the `RACDCERT GENCERT` command. The internal server certificate was signed by the internal certificate authority. To generate the certificates for the internal server:

```
RACDCERT ID(INSS)
         GENCERT
         SUBJECTSDN(CN('Internal Secure Server') C('US'))
         WITHLABEL('INSS-001')
         SIGNWITH(CERTAUTH LABEL('ACME'))
```

#### Authority required:

CONTROL to the resource IRR.DIGTCERT.GENCERT  
 in the FACILITY class,  
 or  
 READ to IRR.DIGTCERT.INSS.INSS-001.UPD.GENCERT and  
 IRR.DIGTCERT.CERTIFAUTH.ACME.UPD.GENCERT  
 in the RDATA LIB class.

The external server certificate must be signed by the "Really Big" external certificate authority. To do that, follow Steps 2 - 6 in Scenario 1 replacing the ID and LABEL values specified with ID(EXSS) and LABEL('EXSS-001').

5. Create key rings for the secure servers.

```
RACDCERT ID(EXSS) ADDRING(RING01)
RACDCERT ID(INSS) ADDRING(RING01)
```

#### Authority required:

UPDATE to the resource IRR.DIGTCERT.ADDRING  
 in the FACILITY class,  
 or  
 READ to EXSS.RING01.UPD.ADDRING, and  
 INSS.RING01.UPD.ADDRING  
 in the RDATA LIB class.

6. Connect the certificates to INSS's key ring.

```
RACDCERT ID(INSS) CONNECT(ID(INSS) LABEL('INSS-001') RING(RING01) DEFAULT)
RACDCERT ID(INSS) CONNECT(CERTAUTH LABEL('ACME') RING(RING01))
```

#### Authority required:

CONTROL to the resource IRR.DIGTCERT.CONNECT  
 in the FACILITY class,  
 or  
 READ to INSS.RING01.UPD.CONNECT, and  
 IRR.DIGTCERT.INSS.INSS\_001.LST.CONNECT  
 in the RDATA LIB class.

7. Connect the certificates to EXSS's key ring.

```
RACDCERT ID(EXSS) CONNECT(ID(EXSS) LABEL('EXSS-001') RING(RING01) DEFAULT)
RACDCERT ID(EXSS) CONNECT(CERTAUTH LABEL('ACME') RING(RING01))
RACDCERT ID(EXSS) CONNECT(CERTAUTH LABEL('Really Big') RING(RING01))
```

#### Authority required:

CONTROL to the resource IRR.DIGTCERT.CONNECT  
in the FACILITY class,  
or  
READ to EXSS.RING01.UPD.CONNECT,  
IRR.DIGTCERT.EXSS.EXSS-001.LST.CONNECT,  
IRR.DIGTCERT.CERTIFAUTH.ACME.LST.CONNECT, and  
IRR.DIGTCERT.CERTIFAUTH.REALLY\_BIG.LST.CONNECT  
in the RDATA LIB class.

8. Give user INSS permission to read its own key ring by administering a profile in either the FACILITY or the RDATA LIB class.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(INSS) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB INSS.RING01.LST UACC(NONE)
PERMIT INSS.RING01.LST CLASS(RDATA LIB) ID(INSS) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

9. Configure INSS's software to use RING01 for SSL. For example, for z/OS HTTP Server, set the keyFile directive to KeyFile RING01 SAF.

**Note:** RACF is not involved with this step.

10. Give user EXSS permission to read its own key ring by administering a profile in either the FACILITY or the RDATA LIB class.

- When using the FACILITY class:

```
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(EXSS) ACCESS(READ)
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
PERMIT EXSS.RING01.LST CLASS(RDATA LIB) ID(EXSS) ACCESS(READ)
SETROPTS RACLIST(RDATA LIB) REFRESH
```

11. Configure EXSS's software to use RING01 for SSL. For example, for z/OS HTTP Server, set the keyFile directive to KeyFile RING01 SAF.

**Note:** RACF is not involved with this step.

## Scenario 5: Creating client browser certificates with a locally signed certificate

The installation wants to locally issue client browser certificates. This is similar to “Scenario 2: Secure server with a locally signed certificate” on [page 592](#) in that a local certificate-authority certificate must

first be created. In this case, a client certificate is created, locally signed, exported from RACF in PKCS #12 format, and imported into the user's browser.

1. Follow Steps 1 through 6 as described in [“Scenario 2: Secure server with a locally signed certificate” on page 592](#) to create a local certificate-authority certificate to use for signing client browser certificates.
2. User MARKN can obtain a local browser certificate for himself using the following command:

```
RACDCERT ID(MARKN)
GENCERT
SUBJECTSDN(CN('Mark Napolitano')
            OU('Local Certificate Authority')
            O('XYZZY')
            C('US'))
WITHLABEL('My Browser Cert')
KEYUSAGE(HANDSHAKE)
SIGNWITH(CERTAUTH LABEL('XYZZY Local Certificate Authority'))
```

3. Export the certificate and private key to an MVS data set in PKCS #12 binary form where the password is 'The circus is coming':

```
RACDCERT ID(MARKN)
EXPORT
(LABEL('My Browser Cert'))
DSN('MARKN.BROWSEC.P12BIN')
PASSWORD('The circus is coming')
FORMAT(PKCS12DER)
```

4. Use FTP to send the exported certificate data set in binary format to the target workstation. Use the appropriate browser-specific procedure to import the PKCS #12 package.

**Note:** RACF is not involved with this step.

5. Optionally, the certificate labeled 'My Browser Cert' can be deleted from the RACF database if an appropriate certificate name filter is available to provide a user ID association, and the specific association between this certificate and the user ID MARKN is not required.

## Scenario 6a: Enabling secure outbound FTP using a shared virtual key ring

A company wants to allow its employees to make FTP requests from z/OS to three FTP servers out on the Internet. The clients (z/OS users) will authenticate to the FTP servers with preestablished user IDs and passwords. Therefore, FTP will be used without client authentication. For privacy protection, the company will use secure FTP to encrypt the information being transferred. To use the FTP client with SSL, a key ring containing the certificate authority certificates must be specified for the target FTP servers. Because a client certificate is not required, one key ring will suffice for all users. You can use a virtual key ring or a real key ring. This scenario uses a virtual key ring. (For instructions using a real key ring, see Scenario 6b.) In this scenario, the CA certificates for the three FTP servers were already obtained and reside in the following three data sets: 'FTPD.CACERT1', 'FTPD.CACERT2', and 'FTPD.CACERT3'.

1. Add the three certificate authority certificates to RACF:

```
RACDCERT CERTAUTH ADD('FTPD.CACERT1') WITHLABEL('CA for FTP Server 1')
RACDCERT CERTAUTH ADD('FTPD.CACERT2') WITHLABEL('CA for FTP Server 2')
RACDCERT CERTAUTH ADD('FTPD.CACERT3') WITHLABEL('CA for FTP Server 3')
```

2. Authorize access to the virtual key ring under CERTAUTH for the z/OS users (USER01, USER02) who need to communicate with the external FTP servers. Do this by administering a profile in either the FACILITY or the RDATA LIB class. Using the FACILITY class provides global control of all rings, whereas using the RDATA LIB class provides granular control of a specific ring.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(USER01 USER02) ACCESS(READ)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB CERTIFAUTH.IRR_VIRTUAL_KEYRING.LST UACC(NONE)
PERMIT CERTIFAUTH.IRR_VIRTUAL_KEYRING.LST CLASS(RDATA LIB)
ID(USER01 USER02) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

3. Configure the FTP client to use the virtual key ring under CERTAUTH by specifying the following KEYRING directive:

```
KEYRING *AUTH*/*
```

**Note:** RACF is not involved with this step.

## Scenario 6b: Enabling secure outbound FTP using a shared real key ring

A company wants to allow its employees to make FTP requests from z/OS to three FTP servers out on the Internet. The clients (z/OS users) will authenticate to the FTP servers with preestablished user IDs and passwords. Therefore, FTP will be used without client authentication. For privacy protection, the company will use secure FTP to encrypt the information being transferred. To use the FTP client with SSL, a key ring containing the certificate authority certificates must be specified for the target FTP servers. Because a client certificate is not required, one key ring will suffice for all users. You can use a virtual key ring or a real key ring. This scenario uses a real key ring. (For instructions using a virtual key ring, see Scenario 6a.) In this scenario, the CA certificates for the three FTP servers were already obtained and reside in the following three data sets: 'FTPD.CACERT1', 'FTPD.CACERT2', and 'FTPD.CACERT3'.

1. Add the three certificate authority certificates to RACF:

```
RACDCERT CERTAUTH ADD('FTPD.CACERT1') WITHLABEL('CA for FTP Server 1')
RACDCERT CERTAUTH ADD('FTPD.CACERT2') WITHLABEL('CA for FTP Server 2')
RACDCERT CERTAUTH ADD('FTPD.CACERT3') WITHLABEL('CA for FTP Server 3')
```

2. Create a key ring to hold the three CA certificates. (This key ring represents the FTP trust policy for this company.) The key ring must be associated with a single user ID (for example, SHAREID) even though it will be shared by multiple users.

```
RACDCERT ID(SHAREID) ADDRING(RING01)
```

3. Connect the three certificate authority (CA) certificates to the shared key ring. Because this key ring will contain only CA certificates, it will not have a default certificate. Therefore, the DEFAULT keyword is not specified.

```
RACDCERT ID(SHAREID) CONNECT(CERTAUTH LABEL('CA for FTP Server 1') RING(RING01))
RACDCERT ID(SHAREID) CONNECT(CERTAUTH LABEL('CA for FTP Server 2') RING(RING01))
RACDCERT ID(SHAREID) CONNECT(CERTAUTH LABEL('CA for FTP Server 3') RING(RING01))
```

4. Authorize access to the shared key ring for the ring owner (SHAREID) and for the z/OS users (USER01 and USER02) who need to communicate with the external FTP servers. Do this by administering a profile in either the FACILITY or the RDATA LIB class.

- When using the FACILITY class:

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(USER01 USER02) ACCESS(UPDATE)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB SHAREID.RING01.LST UACC(NONE)
PERMIT SHAREID.RING01.LST CLASS(RDATA LIB) ID(USER01 USER02) ACCESS(READ)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

5. Configure the FTP client to use the shared key ring by specifying its fully qualified name for the KEYRING directive:

```
KEYRING SHAREID/RING01
```

**Note:** RACF is not involved with this step.

## Scenario 7: Sharing one certificate among multiple servers

This is similar to “[Scenario 1: Secure server with a certificate signed by a certificate authority](#)” on page 590 with the exception that this scenario involves sharing a single certificate (and its private key) between two servers, the inventory server and the FTP server. The inventory server is associated with the user ID INVSERV, while the FTP server is associated with user ID FTPD.

1. Generate a self-signed placeholder certificate for the two servers. Because this certificate will be shared, it should be associated with SITE rather than the INVSERV user ID.

```
RACDCERT SITE GENCERT
SUBJECTSDN(CN('xyzy.com') OU('Shared') O('XYZZY') C('US'))
WITHLABEL('Shared Server')
```

2. Create a certificate request to send to your chosen certificate authority. The certificate request that we are creating is based on the certificate that we created in the previous step. Place this certificate into the data set 'MARKN.SHRSERV.GENREQ'.

```
RACDCERT GENREQ(LABEL('Shared Server'))
SITE DSN('MARKN.SHRSERV.GENREQ')
```

3. Send the certificate request to the CA and receive the returned certificate from the CA into data set 'MARKN.SHRSERV.CERT'. The **ADD (data-set-name) parameter** specifies the data set containing one digital certificate or certificate package in binary or Base64 format. You must specify a cataloged sequential data set. The record format (RECFM) expected by RACDCERT is variable blocked (VB). When you specify the ADD function, RACDCERT opens the specified data set, and reads in the content to add the certificate(s).

**Restriction:** The certificate request data contained in the data set must be sent to, and received from, the external CA using the process defined by the CA. Those steps are not included.

4. Add the certificate signed by the certificate authority. This will automatically replace the original self-signed certificate with the actual certificate.

**Attention:** Do not delete the self-signed certificate before performing this step.

The example command uses the SITE keyword because the self-signed placeholder certificate was created under SITE and will only be replaced if the SITE keyword is specified on the RACDCERT ADD command. If SITE is not specified, then the certificate is added as a new certificate.

```
RACDCERT SITE ADD('MARKN.SHRSERV.CERT') WITHLABEL('Shared Server')
```

5. Create a key ring to be shared between the two user IDs. (The key ring must be associated with a single user ID even though it will be shared by the two servers. Therefore, associate the key ring with the user ID of one of the two servers.) Then, connect the shared certificate to INVSERV's key ring and mark it as the default certificate.

```
RACDCERT ID(INVSERV) ADDRING(RING01)
RACDCERT ID(INVSERV) CONNECT(SITE LABEL('Shared Server')
RING(RING01) USAGE(PERSONAL) DEFAULT))
```

6. Protect the shared key ring and the private key of the certificate by administering a profile in the FACILITY or RDATA LIB class and permit the user IDs of each server access to access the certificate and private key.

- When using the FACILITY class:

- a. Permit the user ID of each server to access the key ring. Because the key ring is associated with INVSERV, the user ID for the inventory server needs only READ access.

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(INVSERV) ACCESS(READ)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(FTPD) ACCESS(UPDATE)
```

- If the FACILITY class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(FACILITY) RACLIST(FACILITY)
```

- If the FACILITY class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

- b. Permit the user ID of each server to access the private key. They each need CONTROL access to IRR.DIGTCERT.GENCERT.

```
RDEFINE FACILITY IRR.DIGTCERT.GENCERT UACC(NONE)
PERMIT IRR.DIGTCERT.GENCERT CLASS(FACILITY) ID(INVSERV FTPD)
ACCESS(CONTROL)
```

- When using the RDATA LIB class:

```
RDEFINE RDATA LIB INVSERV.RING01.LST UACC(NONE)
PERMIT INVSERV.RING01.LST CLASS(RDATA LIB) ID(INVSERV) ACCESS(READ)
PERMIT INVSERV.RING01.LST CLASS(RDATA LIB) ID(FTPD) ACCESS(UPDATE)
```

- If the RDATA LIB class is not already active, activate and RACLIST it.

```
SETROPTS CLASSACT(RDATA LIB) RACLIST(RDATA LIB)
```

- If the RDATA LIB class is already active and RACLISTed, refresh it.

```
SETROPTS RACLIST(RDATA LIB) REFRESH
```

If the caller's user ID has CONTROL authority to the <ringOwner>.<ringName>.LST resource in the RDATA LIB class, it can extract the private key from a CERTAUTH or SITE certificate.

7. Configure each server to use the shared key ring. Because the key ring is associated with INVSERV, the KEYRING directive need not be fully qualified for when you configure the INVSERV server. For the FTP server, specify the fully qualified name.



Server user ID	Key ring directive
INVSERV	KEYRING RING01
FTP	KEYRING INVSERV/RING01

If INVSERV is a z/OS HTTP Server, set the keyFile directive to:

```
KeyFile RING01 SAF
```

**Note:** RACF is not involved in this step.

## Scenario 8: Using the IBM Encryption Facility for z/OS

Wen Ting is preparing to use the IBM Encryption Facility for z/OS to encrypt some of her own data sets and to recover encrypted data sent to her by Yun, a business partner. She wants to create a new certificate with a 2048-bit RSA public/private key pair called Wen Ting's certificate. IBM Encryption Facility requires a PKDS label so she allows RACF to create a default PKDS label.

1. Wen Ting creates her certificate using the RACDCERT GENCERT command:

```
RACDCERT GENCERT SUBJECTSDN(CN('Wen Ting's certificate'))
WITHLABEL('Wen Ting's certificate') SIZE(2048) PKDS
NOTAFTER(DATE(2020/08/10))
```

2. She lists the certificate to see the PKDS label generated by RACF. Here is her RACDCERT LIST command and her output:

```
RACDCERT LIST(LABEL('Wen Ting's certificate'))

Digital certificate information for user WENTING:
  Label: Wen Ting's certificate
  Certificate ID: 2QfHxdbZx8XUaqweQMOFmaNA46iXhUBgQOKFmUB7QPDw
  Status: TRUST
  Start Date: 2005/08/11 00:00:00
  End Date: 2028/08/10 23:59:59
  Serial Number:
    >00<
  Issuer's Name:
    >CN=Wen Ting's certificate<
  Subject's Name:
    >CN=Wen Ting's certificate<
  Signing Algorithm: sha256RSA
  Key Type: RSA
  Key Size: 2048
  Private Key: YES
  PKDS Label: IRR.
  PKDS Label: IRR.DIGTCERT.WENTING.SY1.BD7103108611F42F
  Certificate Fingerprint (SHA256):
    4A:23:79:FA:B4:81:DF:D3:32:E3:18:9B:85:42:E9:36:
    13:D8:93:D7:EE:84:4E:10:A7:93:4E:F1:6D:A2:54:25
  Ring Associations:
    *** No rings associated ***
```

3. Wen Ting needs to send her new certificate to Yun so that he can send her his encrypted data. Before doing this, she exports the certificate using this RACDCERT EXPORT command:

```
RACDCERT EXPORT(LABEL('Wen Ting's certificate')) DSN(FOR.YUN.CRT)
```

4. She sends the exported certificate to Yun using email or FTP. (RACF is not involved with this step.) The exported certificate does not contain the private key so the data set that she transmits need not be protected in any way.
5. When Yun receives the file, he adds it to his company's RACF database as a site certificate using the RACDCERT ADD command and calls it WenTing. To use the IBM Encryption Facility, he also needs the public key for this certificate stored in the ICSF PKDS.

Yun chooses to add the certificate from Wen Ting as a site certificate on his system (using the SITE operand). The SITE designation is most appropriate for this certificate because it will not be used as a

CA certificate (so CERTAUTH is not correct) and because Wen Ting is not a user on his system (so USER is not required).

Here is Yun's RACDCERT ADD command:

```
RACDCERT SITE ADD(WENTING.CRT) WITHLABEL('WenTing') ICSF(*)
```

6. Yun lists the new certificate to see the PKDS label. Here is his RACDCERT LIST command and his output:

```
RACDCERT SITE LIST(LABEL('WenTing'))  
  
Digital certificate information for SITE:  
Label: WenTing  
Certificate ID: egljcv8XUaqweQMOFmaNA46iXhUBgQOKFmUB7QPDw  
Status: TRUST  
Start Date: 2005/08/11 00:00:00  
End Date: 2028/08/10 23:59:59  
Serial Number:  
    >00<  
Issuer's Name:  
    >CN=Wen Ting's certificate<  
Subject's Name:  
    >CN=Wen Ting's certificate<  
Signing Algorithm: sha256RSA  
Key Type: RSA  
Key Size: 2048  
Private Key: No  
PKDS Label: WENTING  
Certificate Fingerprint (SHA256):  
    4A:23:79:FA:B4:81:DF:D3:32:E3:18:9B:85:42:E9:36:  
    13:D8:93:D7:EE:84:4E:10:A7:93:4E:F1:6D:A2:54:25
```

Compare Wen Ting and Yun's output listings. They now share the same certificate and can begin exchanging encrypted information.

## Chapter 23. Controlling applications that invoke callable services

This topic provides information about using RACF to authorize applications that invoke certain SAF callable services. It also provides additional usage information specific to the RACF implementation of SAF callable services.

SAF callable services are supported by RACF and might be supported by other security products. See *z/OS Security Server RACF Callable Services* for detailed information about invoking these services.

### Authorizing applications

In general, applications that run in system key or in supervisor state do *not* require RACF authorization to use callable services. Applications that do not run in system key or in supervisor state require RACF authorization. Certain callable services, such as R\_admin (IRRSEQ00) for envelope retrieval functions, are exceptions.

Before you can use RACF to authorize applications that do not run in system key or supervisor state, you must define RACF user IDs for them. Then, permit at least READ access to the appropriate general resource, usually in the FACILITY class, as described in following topics for each callable service.

### Defining applications as RACF users

Each application, such as a server, must be defined as a RACF user, if not already defined. The application might run as a job or a started procedure. If the application executes as a batch job, define the RACF user ID associated with the batch job. If the application executes as a started procedure, assign a RACF user ID using one of the following methods:

- Add the procedure name as an entry in the STARTED class. (This is the preferred method.)
- Add the procedure name in the RACF started procedure table (ICHRIN03), unless this table has already been modified by your installation to contain a generic entry.

**Guideline:** Assign the PROTECTED attribute to the user IDs that you associate with applications. For more information, see [“Assigning RACF user IDs to started procedures” on page 141](#).

### Defining resources that control callable services

In general, define resources in the FACILITY class to authorize applications that do not run in system key or supervisor state. Certain callable services, such as initACEE (IRRSIA00) use general resources in classes other than the FACILITY class, such as the SERVAUTH class.

#### Guidelines:

- Ensure that an existing generic profile in the general facility class does not inadvertently grant authority to the resources you use to control callable services.
- Define profiles to protect these resources using UACC(NONE), at least until you determine which applications to authorize.

### Activating your authorizations

Activate the general resource class, usually the FACILITY class, to activate your authorizations. If it is not already active at your installation, activate the class using the SETROPTS command.

#### Example:

```
SETROPTS CLASSACT(FACILITY)
```

If your installation maintains profiles for the general resource class in storage using SETROPTS RACLIST processing, you must issue the following command to refresh the class after you define or alter any profiles in the class.

**Example:**

```
SETROPTS RACLIST(FACILITY) REFRESH
```

In general, if the resource class is not active or the appropriate resource is not defined when an application invokes the callable service, only applications running in system key or supervisor state can use the callable service.

## initACEE (IRRSIA00) callable service

---

Authorized applications, such as servers, that invoke the `initACEE` callable service (IRRSIA00) can administer certificates associated with users and certificates associated with certificate authorities. You authorize these applications by administering the same FACILITY class resources checked by the RACDCERT command. See [“Controlling the use of the RACDCERT command”](#) on page 549.

Authorized applications, such as Web servers, can also present a client's certificate containing a `hostIdMappings` extension and invoke the `initACEE` callable service to request to have a security context (ACEE) created or have the client's user ID queried and returned. You authorize these applications by administering profiles in the SERVAUTH class.

## Registering user certificates

Applications can invoke the `initACEE` callable service (IRRSIA00) and pass a user certificate requesting registration. If the caller's user ID has at least READ authority to the IRR.DIGTCERT.ADD resource, the certificate is associated with that user ID. This `initACEE` register function performs the same function as the RACDCERT ADD command. Therefore, a profile is created in the DIGTCERT class with the user ID in the APPLDATA field, and the user's profile is updated with the name of the DIGTCERT profile. Once the certificate is registered, it can be used in the same manner as a certificate that was registered using the RACDCERT command.

**Note:** RACF generates a label name for user certificates registered through the `initACEE` callable service. The generated label name is of the form `LABELnnnnnnnn` where `nnnnnnnn` is the first integer value (starting at 00000001) that generates a unique label name.

## Deregistering user certificates

Applications can invoke the `initACEE` callable service (IRRSIA00) requesting deregistration of a user certificate. This deregistration function performs the same function as the RACDCERT DELETE command, and disassociates the certificate from the current user ID. If the caller's user ID has at least READ authority to the IRR.DIGTCERT.DELETE resource, the profile in the DIGTCERT class associating the certificate with this user ID is deleted, and the name of the DIGTCERT profile is removed from the user's profile.

## Replacing certificate-authority certificates

Applications can invoke the `initACEE` callable service (IRRSIA00) and pass a certificate-authority certificate, requesting replacement of a previously registered certificate-authority certificate. If the caller's user ID has at least CONTROL authority to the IRR.DIGTCERT.ADD resource and the previously registered certificate-authority certificate is eligible for replacement, the certificate will be replaced and the new certificate will be associated with the `irrcerta` user ID.

A previously registered certificate-authority certificate is eligible for replacement when:

1. Its public key matches that of the input certificate-authority certificate.
2. Its subject's distinguished name matches that of the input certificate-authority certificate.
3. It has a private key.

If the caller has CONTROL authority to the IRR.DIGTCERT.ADD resource but the previously registered certificate-authority certificate is not eligible for replacement, it will not be replaced. The input certificate will be added as a user certificate and will be associated with the user ID of the caller. See [“Registering user certificates”](#) on page 604.

## Using a hostIdMappings extension

Authorized applications, such as Web servers, can present a client's certificate containing a hostIdMappings extension and invoke the initACEE callable service (IRRSIA00) to request to have a security context (ACEE) created or have the client's user ID queried and returned if the attributes value INTA\_X500\_RET is set and the X500name parameter is passed. For the format of the hostIdMappings extension, see [z/OS Security Server RACF Callable Services](#).

In these cases, the application is seeking to complete a login for a client whose certificate includes a hostIdMappings extension that might specify the user ID to be used on a particular server (host). Controlling an identity used for login purposes is a very important security objective. Therefore, you should exercise administrative control in the following areas by authorizing:

1. Which certificates with a hostIdMappings extension will be honored
2. Which servers will be authorized to accept logins using certificates that contain explicit user IDs and host names

When an application calls the initACEE callable service for this purpose and passes a certificate that has a hostIdMappings extension, the caller must have at least READ authority for the IRR.HOST.*host-name* resource defined in the SERVAUTH class, and the certificate must have been issued by a certificate authority that is defined with the HIGHTRUST option.

The initACEE callable service builds a security context (ACEE) for the user ID contained in hostIdMappings extension only if the certificate presented is not registered in the RACF database, and there is no matching certificate name filter.

## Administering profiles in the SERVAUTH class

You authorize servers to accept logins for clients whose certificates contain a hostIdMappings extension by administering profiles in the SERVAUTH class. Be sure that each server you want to authorize is defined as a RACF user, if not already defined. Servers might run as jobs or started procedures. For example:

```
ADDGROUP WEBSRVGP
ADDUSER WEBSRV1 GROUP(WEBSRVGP) NOPASSWORD
ADDUSER WEBSRV2 GROUP(WEBSRVGP) NOPASSWORD
```

**Note:** You should assign protected user IDs for servers using the NOPASSWORD option. See [“Assigning RACF user IDs to started procedures”](#) on page 141.

Define resources in the SERVAUTH class using the following format:

```
IRR.HOST.host-name
```

Permit servers to access this resource with at least READ authority. This will allow them to accept logins for the host name specified in the resource name. For example, to allow the servers in the WEBSRVGP to accept logins for the host system called MVSDSN1, execute the following commands:

```
RDEFINE SERVAUTH IRR.HOST.MVSDSN1 UACC(NONE)
PERMIT IRR.HOST.MVSDSN1 CLASS(SERVAUTH) ID(WEBSRVGP) ACCESS(READ)
SETROPTS CLASSACT(SERVAUTH)
```

In this example, if a server running under the authority of user ID WEBSRV1 presents a client certificate issued by a certificate authority with HIGHTRUST status and the certificate contains a hostIdMappings extension that includes a user ID mapping for host name MVSDSN1, a security context (ACEE) will be created for the user ID mapped to MVSDSN1, as indicated in the hostIdMappings extension.

## Using the HIGHTRUST option

You can control which certificates with a `hostIdMappings` extension will be honored by authorized servers. You do this by defining the certificate authority that issues these certificates as highly trusted on your system. For example:

```
RACDCERT CERTAUTH ALTER(LABEL('Local Certificate Authority')) HIGHTRUST
```

See [z/OS Security Server RACF Command Language Reference](#) for details about syntax and authorization required for using the RACDCERT command.

## R\_admin (IRRSEQ00) callable service

---

Applications, such as servers, can invoke the R\_admin callable service (IRRSEQ00) to manage and retrieve RACF profile data and SETROPTS data.

For information about IRRSEQ00 functions and authorization required, see [R\\_admin \(IRRSEQ00\): RACF administration API](#) in [z/OS Security Server RACF Callable Services](#).

## R\_auditx (IRRSAX00) callable service

---

Authorized applications, such as servers, can invoke the R\_auditx callable service (IRRSAX00) to generate an SMF type 83 audit record that records a security-related event and optionally to issue a message to the console indicating an authentication or authorization failure. For detailed information about invoking the R\_auditx callable service, see [z/OS Security Server RACF Callable Services](#).

To authorize applications that do not run in system key or supervisor state, you must define RACF user IDs for them (see “[Defining applications as RACF users](#)” on page 603) and authorize their user IDs or groups for at least READ access to the IRR.RAUDITX resource in the FACILITY class.

## R\_cacheserv (IRRSCH00) callable service

---

Authorized applications, such as servers, can invoke the R\_cacheserv callable service (IRRSCH00) to request the storage or retrieval of security-relevant information from a cache.

To authorize applications that do not run in system key or supervisor state, you must define RACF user IDs for them (see “[Defining applications as RACF users](#)” on page 603) and authorize their user IDs or groups for at least READ access to the resource called IRR.RCACHESERV.cachename in the FACILITY class.

With READ authority, the server is authorized to use only the FETCH function of IRRSCH00. With UPDATE authority or higher, the server can use all functions of the service.

Authorization changes that you make for a server currently invoking IRRSCH00 will not take effect until the job step task invoking the service ends and a new one is started.

## R\_data1ib (IRRSDL00 or IRRSDL64) callable service

---

Authorized applications, such as servers, that invoke the R\_data1ib callable service (IRRSDL00 or IRRSDL64) can extract private keys and manage certificate serial numbers. You authorize these applications for these functions by administering the same FACILITY class resources checked by the RACDCERT command. See “[Controlling the use of the RACDCERT command](#)” on page 549.

Authorized applications can also use R\_data1ib callable service to create key rings (except virtual key rings) and populate them by connecting certificates. For authorization details about all functions of R\_data1ib, see [RACF Authorization for R\\_data1ib \(IRRSDL00 or IRRSDL64\)](#) in [z/OS Security Server RACF Callable Services](#).

## Extracting private keys

Applications can invoke the `R_data1ib` callable service (IRRSDL00 or IRRSDL64) to extract private keys associated with certain certificates in key rings, virtual key rings, and z/OS PKCS #11 tokens, under certain conditions.

For details about when private keys can be extracted, see [Usage Notes for R\\_data1ib \(IRRSDL00 or IRRSDL64\)](#) in *z/OS Security Server RACF Callable Services*.

## Managing certificate serial numbers

Applications can invoke the `R_data1ib` callable service (IRRSDL00 or IRRSDL64) to manage serial numbers for certain certificates.

### User certificates

An application can increment the "last serial number issued" (CERTLSER) for a user certificate if the following conditions are met:

1. The caller's user ID is the user ID associated with the certificate.
2. The caller's user ID has at least READ authority to the resource IRR.DIGTCERT.GENCERT in the FACILITY class.

### CERTAUTH and SITE certificates

An application can increment the "last serial number issued" (CERTLSER) for a CERTAUTH or SITE certificate, if the caller's user ID has at least CONTROL authority to the resource IRR.DIGTCERT.GENCERT in the FACILITY class.

## R\_dcekey (IRRSDK00) callable service

Authorized applications, such as servers, can invoke the `R_dcekey` callable service (IRRSDK00) to enable z/OS DCE to retrieve or set a DCE password (a key), or to retrieve an LDAP bind password. The following functions are supported:

- Retrieve a DCE password from a user profile's DCE segment. (The password is decrypted using the key that was stored in the user's DCE segment when the password was encrypted.)
- Set the DCE password in a user profile's DCE segment. (The password is encrypted using the key stored in the DCE.PASSWORD.KEY profile in the KEYSMSTR class.)
- Retrieve the LDAP bind password from the PROXY segment of a general resource profile in the LDAPBIND class or from the IRR.PROXY.DEFAULTS profile in the FACILITY class. (The password is decrypted using the key that was stored in the profile's PROXY segment when the password was encrypted, for example when the RDEFINE or RALTER PROXY command was issued.)

For detailed information about invoking the `R_dcekey` callable service, see [z/OS Security Server RACF Callable Services](#).

For callers not running in system key or supervisor state, all of the following conditions must be met:

- The caller must be running in a *clean* environment. (For more information, see [“Maintaining a clean environment in BASIC or ENHANCED mode”](#) on page 306.)
- The caller's user ID or group must be authorized for at least READ authority to either one of the following FACILITY class profiles:
  - BPX.SERVER
  - IRR.RDCEKEY

When authorizing applications using the BPX.SERVER resource, the caller is defined as the user ID associated with the ACEE of the address space. When authorizing applications using the IRR.RDCEKEY resource, the caller is defined as the user ID associated with the ACEE of the current TCB or, if no ACEE

is associated with the current TCB, then the ACEE associated with the address space is used to locate the user ID.

## **R\_GetInfo (IRRSGL00) callable service**

---

Authorized applications, such as servers, can invoke the R\_GetInfo callable service (IRRSGL00) to retrieve a subset of Security Server information. Invokers provide a function code to identify which subset of information is requested. For more information about invoking the R\_GetInfo callable service, see [z/OS Security Server RACF Callable Services](#).

For callers not running in system key or supervisor state, the caller's user ID or group must be authorized for at least READ authority to one of the following FACILITY class profiles:

- BPX.SERVER
- IRR.RGETINFO.EIM
- IRR.RGETINFO.REALM

When authorizing applications through the BPX.SERVER resource, the caller is defined as the user ID associated with the ACEE of the address space.

When authorizing applications through the IRR.RGETINFO.EIM or IRR.RGETINFO.REALM resource, the caller is defined as the user ID associated with the ACEE of the current TCB or, if no ACEE is associated with the current TCB, then the ACEE associated with the address space is used to locate the user ID.

If user-related custom fields (extracted from the CSDATA segment of the USER profile) are available in the user ACEE, problem-state applications can retrieve this information by using the R\_GetInfo callable service. For problem state callers, the user ID that is represented by the environmental ACEE (task or address space) must be authorized in the FIELD class to see the CSDATA segment fields being requested.

## **R\_dceruid (IRRSUD00) callable service**

---

You need to define the IRR.RDCERUID profile in the FACILITY class to control the use of the SAF R\_dceruid callable service. This maps the DCE UUID to the RACF user ID. Applications that use this service must have at least READ access to the resource IRR.RDCERUID.

## **R\_PKIServ (IRRSPX00) callable service**

---

Authorized applications, such as servers, that invoke the R\_PKIServ callable service (IRRSPX00) can request the generation, retrieval, and administration of PKIX-compliant X.509 Version 3 certificates and certificate requests. Applications can request end-user functions or administrative functions related to these requests. You authorize these applications by administering RACF resources in the FACILITY class, based on whether the application requests end-user functions or administrative functions.

See [z/OS Security Server RACF Callable Services](#) for the details of invoking IRRSPX00.

## **Authorizing end-user functions**

The end-user functions are:

### **EXPORT**

Retrieves (exports) a previously requested certificate, or retrieves (exports) the PKI Services registration authority (RA) certificate or the certificate authority (CA) certificate.

### **GENCERT**

Generates an auto-approved certificate.

### **GENRENEW**

Generates an auto-approved renewal certificate. (The request submitted is automatically approved.)

### **QRECOVER**

Lists certificates whose key pairs were generated by PKI Services under a requestor's e-mail address and passphrase.



**REQCERT**

Requests a certificate that an administrator must approve before it is created.

**REQRENEW**

Requests certificate renewal. The administrator needs to approve the request before the certificate is renewed.

**RESPOND**

Invokes the PKI OCSP responder.

**REVOKE**

Revokes a certificate that was previously issued.

**SCEPREQ**

Generates a certificate request using Simple Certificate Enrollment Protocol (SCEP).

**VERIFY**

Confirms that a given user certificate was issued by this certificate authority and, if so, returns the certificate fields.

For end-user functions, FACILITY class resources protect this interface. Access authority is based on the user ID for the application (the user ID from the ACEE associated with the address space). To determine the user ID for the application, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.

The form for the FACILITY class resources is:

```
IRR.RPKISERV.function[.ca_domain]
```

***function***

Specifies one of the end-user function names in the preceding list.

***ca\_domain***

Optionally specifies the PKI Services certificate authority (CA) domain name. Use this when your installation has established multiple PKI Services CAs and the CA\_domain parameter is provided with IRRSPX00.

**Restriction:** If the name of your initial CA domain is longer than 8 characters, you must truncate it to exactly 8 characters when you define the resource name in the FACILITY class.

**Example:** For the GENCERT function, when the *ca\_domain* is named Customers and the CA\_domain parameter is provided with IRRSPX00, then the FACILITY class resource controlling the function is IRR.RPKISERV.GENCERT.CUSTOMER. (The name Customers was truncated to CUSTOMER. See the restriction for the *ca\_domain* parameter.) When the CA\_domain parameter is not provided with IRRSPX00, the FACILITY class resource is IRR.RPKISERV.GENCERT.

The access authorities you can assign for these FACILITY class resources have the following effects:

**NONE**

Access is denied.

**READ**

Access is permitted based on subsequent access checks against the caller's user ID.

**UPDATE**

Access is permitted based on subsequent access checks against the application's user ID.

**CONTROL (or user ID has RACF SPECIAL)**

Access is permitted, and no subsequent access checks are made.

**Example:** If you defined the FACILITY class profile IRR.RPKISERV.GENCERT.CUSTOMER to control access to the GENCERT function on the CA domain named Customers, you can prevent the user ID MYAPP from using the GENCERT function on that CA domain by issuing the command:

```
PERMIT IRR.RPKISERV.GENCERT.CUSTOMER CLASS(FACILITY) ID(MYAPP) ACCESS(NONE)
```

For SAF GENCERT and EXPORT requests where the application has READ and UPDATE access, subsequent access checks are performed against the IRR.DIGTCERT.*function* FACILITY resources. These are identical to the checks the RACDCERT TSO command makes. See *z/OS Security Server RACF Command Language Reference* for more information.

For PKI Services EXPORT, GENCERT, GENRENEW, QRECOVER, REQCERT, REQRENEW, RESPOND, REVOKE, SCEPREQ, and VERIFY requests in which the application has READ and UPDATE access, subsequent access checks are performed against the IRR.DIGTCERT.*function* FACILITY resources.

The following table summarizes the access requirements for the user ID whose access is checked.

*Table 38. Summary of access authorities required for PKI Services requests*

<b>Request</b>	<b>Access</b>
EXPORT	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.EXPORT <ul style="list-style-type: none"> <li>– READ access if PassPhrase is specified or if CertID is specified as PKICACERT.</li> <li>– UPDATE access if the PassPhrase parameter is not specified with IRRSPX00.</li> <li>– CONTROL access if you want to export a PKCS #7 certificate.</li> </ul> </li> </ul>
GENCERT	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.GENCERT — CONTROL access</li> <li>• IRR.DIGTCERT.ADD <ul style="list-style-type: none"> <li>– UPDATE access if any hostIdMappings information is specified in the certificate request parameter list or the UserId field in the certificate request parameter list indicates the certificate is being requested for another user other than the caller</li> <li>– READ access otherwise</li> </ul> </li> </ul>
GENRENEW	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.GENRENEW — READ access</li> <li>• IRR.DIGTCERT.GENCERT — CONTROL access</li> </ul> <p><b>Note:</b> It is assumed that the calling application has already verified the input certificate using the VERIFY function.</p>
QRECOVER	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.QRECOVER — READ access</li> </ul>
REQCERT	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.REQCERT — READ access</li> </ul>
REQRENEW	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.REQRENEW — READ access</li> </ul> <p><b>Note:</b> It is assumed that the calling application has already verified the input certificate using the VERIFY function.</p>
RESPOND	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.RESPOND — READ access</li> </ul>
REVOKE	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.REVOKE — READ access</li> </ul> <p><b>Note:</b> It is assumed that the calling application has already verified the target certificate using the VERIFY function.</p>
SCEPREQ	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.SCEPREQ — READ access</li> </ul>
VERIFY	<ul style="list-style-type: none"> <li>• IRR.DIGTCERT.VERIFY — READ access</li> </ul> <p><b>Note:</b> It is assumed that the calling application has already verified that the end user possesses the private key that correlates to the input certificate.</p>

## Authorizing administrative functions

The administrative functions are:

### **CERTDETAILS**

Get detailed information about one PKI Services issued certificate.

### **MODIFYCERTS**

Change PKI Services issued certificates.

### **MODIFYREQS**

Change PKI Services certificate requests.

### **QUERYCERTS**

Query PKI Services issued certificates.

### **QUERYREQS**

Query PKI Services about certificate requests.

### **PREREGISTER**

Preregister clients who use Simple Certificate Enrollment Protocol (SCEP).

### **REQDETAILS**

Get detailed information about one PKI Services certificate request.

There are two ways that you can control access to these functions:

- Use resources in the RACF FACILITY class. This class allows you to control access based on the CA domain.
- Use resources in the RACF PKISERV class. This class allows you to control access on a more granular level than the FACILITY class, based on the CA domain, the administrative function, and the template.

## Using the FACILITY class to control access to administrative functions

For the all administrative functions, the following single FACILITY class resource protects this interface.

```
IRR.RPKISERV.PKIADMIN[.ca_domain]
```

### **ca\_domain**

Optionally specifies the PKI Services certificate authority (CA) domain name. Use this when your installation has established multiple PKI Services CAs and the CA\_domain parameter is provided with IRRSPX00.

**Restriction:** If the name of your initial CA domain is longer than 8 characters, you must truncate it to exactly 8 characters when you define the resource name in the FACILITY class.

- If the caller is RACF SPECIAL, no further access is necessary.
- Otherwise, the caller needs:
  - READ access to perform read operations (QUERYREQS, QUERYCERTS, REQDETAILS, and CERTDETAILS)
  - UPDATE access for the action operations (PREREGISTER, MODIFYREQS and MODIFYCERTS).

**Example:** For administrative functions, when the *ca\_domain* is named Customers and the CA\_domain parameter is provided with IRRSPX00, the FACILITY class resource controlling this interface is IRR.RPKISERV.PKIADMIN.CUSTOMER. (The name Customers was truncated to CUSTOMER. See the restriction for the *ca\_domain* value.) When the CA\_domain parameter is not provided with IRRSPX00, IRR.RPKISERV.PKIADMIN is the name of the FACILITY class resource.

To determine the appropriate access level of the caller, the current TCB is checked for an ACEE. If one is found, the authority of that user is checked. If there is no ACEE associated with the current TCB, the ACEE associated with the address space is used to locate the user ID.



**Attention:** UPDATE access to the IRR.RPKISERV.PKIADMIN[*ca\_domain*] resource also controls who can act as PKI Services administrators. PKI Services administrators play a very powerful role in your organization. The decisions they make when managing certificates and certificate requests determine who will access your computer systems and what privileges they will have when doing so.

**Guideline:** Give UPDATE authority to only highly trusted individuals, but avoid allowing these same individuals to have direct access to the end-user functions of the R\_PKIServ callable service described in [“Authorizing end-user functions” on page 608](#). This helps to maintain a secure separation of duties.

## Using the PKISERV class to control access to administrative functions

You can use profiles in the PKISERV class to control access to R\_PKIServ administrative functions on a more granular level than you can with profiles in the FACILITY class. If the `AdminGranularControl` switch in the `pkiserv.conf` configuration file is set to T, profiles in the PKISERV class are checked in addition to profiles in the FACILITY class to determine authorization to these functions. If no profile is found protecting a function, authorization to the function fails.

In order to use the PKISERV class, you need to take the following steps:

1. Activate generic profile checking for the class:

```
SETOPTS GENERIC(PKISERV)
```

2. Define profiles for the PKISERV class resources and authorize users to use the resources:

```
RDEFINE PKISERV profile_name UACC(NONE)
PERMIT profile_name CLASS(PKISERV) ID(user_ID or group) ACCESS(access_level)
```

3. Activate and RACLIST the class:

```
SETOPTS CLASSACT(PKISERV) RACLIST(PKISERV)
```

Any time that you update the profiles in the class, refresh the in-storage profiles:

```
SETOPTS RACLIST(PKISERV) REFRESH
```

For the query functions (QUERYREQS, QUERYCERTS, REQDETAILS, and CERTDETAILS), the resources in the PKISERV class are of the form:

```
ca_domain.action.template_nickname
```

where

### ***ca\_domain***

Specifies the PKI Services certificate authority (CA) domain name.

#### **Rules:**

- The domain name is at most 8 characters long.
- The domain name can contain only alphanumeric characters and the national characters @, #, and \$.
- If there is no domain name, the qualifier must be NOCADOMAIN.

### ***action***

Specifies the function. It has one of the following values:

- QUERYREQS
- QUERYCERTS
- QUERYREQDETAILS
- QUERYCERTDETAILS

**Rules:**

- For the REQDETAILS function, if the administrator has READ access to the QUERYREQDETAILS profile, the password value is replaced by blanks before it is returned. If the administrator has UPDATE access, the password value is returned.
- For the CERTDETAILS function, if the administrator has READ access to the QUERYCERTDETAILS profile, the password value is replaced by blanks before it is returned. If the administrator has UPDATE access, the password value is returned.
- For all other functions, READ access is sufficient.

***template\_nickname***

Specifies the nickname of the certificate template.

**Rules:**

- The template nickname is at most 8 characters long.
- The template nickname can contain only alphanumeric characters and the national characters @, #, and \$.
- If there is no template nickname, the qualifier must be NONICKNAME.

**Example:** An administrator has either READ or UPDATE access to the FACILITY class profile IRR.RPKISERV.PKIADMIN.MYDOMAIN and also has READ access to the PKISERV class profiles MYDOMAIN.QUERYREQS.1YBSSL and MYDOMAIN.QUERYCERTS.1YBSSL. That administrator can perform QUERYREQS and QUERYCERTS functions on the requests and certificates created with the template '1-Year PKI SSL Browser Certificate' in the domain MYDOMAIN. If that same administrator does not have READ or UPDATE access to the PKISERV class profile MYDOMAIN.QUERYREQS.5YSSSL, that administrator would not be able to perform QUERYREQS functions on requests created with the template '5-Year PKI SSL Server Certificate' in the same domain.

For the update functions (MODIFYREQS, MODIFYCERTS, and PREREGISTER), the resources in the PKISERV class are of the form:

```
ca_domain.action.template_nickname
```

where

***ca\_domain***

Specifies the PKI Services certificate authority (CA) domain name.

**Rules:**

- The domain name is at most 8 characters long.
- The domain name can contain only alphanumeric characters and the national characters @, #, and \$.
- If there is no domain name, the qualifier must be NOCADOMAIN.

***action***

Specifies the function. It has one of the following values:

- PREGISTER
- APPROVE (for MODIFYREQS)
- APPROVEWITHMODS (for MODIFYREQS)
- REJECT (for MODIFYREQS)
- DELETEREQS (for MODIFYREQS)
- REVOKE (for MODIFYCERTS)
- DELETECERTS (for MODIFYCERTS)
- RESUME (for MODIFYCERTS)
- AUTORENEWENABLE (for MODIFYCERTS)
- AUTORENEWDISABLE (for MODIFYCERTS)

- CHANGEMAIL (for MODIFYCERTS)
- CREATECRL (for MODIFYCERTS)
- POSTCERT (for MODIFYCERTS)

### **template\_nickname**

Specifies the nickname of the certificate template.

#### **Rules:**

- The template nickname is at most 8 characters long.
- The template nickname can contain only alphanumeric characters and the national characters @, #, and \$.
- The template is irrelevant for CREATECRL and POSTCERT and the template nickname is not included in the resource name for these actions.
- You must specify a template nickname for the PREREGISTER action.
- For all actions other than CREATECRL, POSTCERT, and PREREGISTER, If there is no template nickname, the qualifier must be NONICKNAME.

Examples:

- READ access to the profile MYDOMAIN.APPROVE.1YBSSL allows the administrator to approve requests under the template '1-Year PKI SSL Browser Certificate' in the domain MYDOMAIN.
- READ access to the profile MYDOMAIN.APPROVEWITHMODS.1YBSSL allows the administrator to modify the content of requests and then approve them under the '1-Year PKI SSL Browser Certificate' template in the domain MYDOMAIN.
- READ access to the profile MYDOMAIN.REVOKE.1YBSSL allows the administrator to revoke or suspend certificates under the '1-Year PKI SSL Browser Certificate' template in the domain MYDOMAIN.
- READ access to the profile MYDOMAIN.PREREGISTER.5YSCEPP allows the administrator to preregister requests under the '5-Year SCEP Certificate - Preregistration' template in the domain MYDOMAIN.

## **R\_proxyserv (IRRSPY00) callable service**

---

Authorized applications, such as servers, can invoke the R\_proxyserv callable service (IRRSPY00) to request the services of the z/OS LDAP server to retrieve information from a directory information tree (DIT) on an LDAP directory. (For an example of an X.500 DIT, see [Figure 52 on page 572](#).) IRRSPY00 can be successfully invoked by applications that execute in Language Environment® and by those that do not.

You authorize these servers by administering a FACILITY class resource called IRR.RPROXYSERV. The server must be associated with a RACF user ID or group that has at least READ authority to this resource to successfully invoke IRRSPY00.

Use of IRRSPY00 requires the installation of the z/OS LDAP server and the LDAP server must operate in program call (PC) support mode and support the extended operations (EXOP) backend. For information about configuring this support, see [z/OS IBM Tivoli Directory Server Administration and Use for z/OS](#).

## **R\_ticketerv (IRRSPK00) callable service**

---

Authorized applications, such as servers, can invoke the R\_ticketerv callable service (IRRSPK00) to extract principal names from a GSS-API context token. This enables an application server to determine the client principal who originated an application-specific request, when the request includes a GSS-API context token and the intended recipient is the application server. For detailed information about invoking the R\_ticketerv callable service, see [z/OS Security Server RACF Callable Services](#).

To authorize applications that do not run in system key or supervisor state, you must define RACF user IDs for them. (See [“Defining applications as RACF users” on page 603](#).) These RACF user IDs must be given at least READ access to a RACF general resource called IRR.RTICKETSERV in the FACILITY class.

For all callers, the use of `R_ticketerv` requires access to `IRRPTAUTH` resources in the `PTKTDATA` class. For details, see [R\\_ticketerv \(IRRSPK00\): Parse or extract](#) in *z/OS Security Server RACF Callable Services*.

## Permitting access to the `IRR.RTICKETSERV` resource

Authorization to use the `R_ticketerv` callable service (`IRRSPK00`) is controlled through a RACF general resource called `IRR.RTICKETSERV` in the `FACILITY` class. You must define a profile to protect this resource, and then permit application user IDs to access the resource with at least `READ` authority.

The following example protects the `IRR.RTICKETSERV` resource in the `FACILITY` class with `UACC(NONE)` and authorizes an application server called `SERVER9` to use `R_ticketerv`.

```
RDEFINE FACILITY IRR.RTICKETSERV UACC(NONE)
PERMIT IRR.RTICKETSERV CLASS(FACILITY) ID(SERVER9) ACCESS(READ)
```

```
SETOPTS CLASSACT(FACILITY)
SETOPTS RACLIST(FACILITY) REFRESH
```





## Chapter 24. RACF and the z/OS LDAP server

This topic provides information on using RACF with the z/OS LDAP server. It covers two topics:

- Defining proxy information about the z/OS LDAP server so that other products can communicate with an LDAP directory. (Using a PROXY segment in the LDAPBIND profile is required for this communication.)
- Setting up LDAP event notification for changes to RACF users, groups, and general resources. (Using a PROXY segment in the LDAPBIND profile is *not* required for LDAP event notification.)

### Defining an LDAPBIND class profile

A profile that is defined to the LDAPBIND class that contains a PROXY segment holds information that is needed by products to communicate with an LDAP directory. That information includes:

- The LDAP server URL and port (LDAPHOST)
- The distinguished name (DN) to use when authenticating to the LDAP server.
- The password to use when authenticating to the LDAP server.

In the following example:

- The profile name is MY.LDAP.SERVER1
- The LDAP server URL is ldap://some.ldap.host:389
- The bind DN is cn=Joe User, ou=Poughkeepsie,o=IBM,c=US
- The bind password is MYPASS1 (which is case-sensitive)

```
RDEFINE LDAPBIND MY.LDAP.SERVER1
  PROXY(LDAPHOST(ldap://some.ldap.host:389)
  BINDDN('cn=Joe User,ou=Poughkeepsie,o=IBM,c=US') BINDPW('MYPASS1'))
```

You can list the PROXY segment with the RLIST command. The bind password is not displayed and only an indication of whether it is present (YES or NO).

```
RLIST LDAPBIND MY.LDAP.SERVER PROXY NORACF
CLASS          NAME
-----
LDAPBIND      MY.LDAP.SERVER1

PROXY INFORMATION
-----
LDAPHOST=LDAP://SOME.LDAP.HOST:389
BINDDN=cn=Joe User,ou=Poughkeepsie,o=IBM,c=US
BINDPW=YES
```

To get PKI Services to use the preceding information, you must update the PKI Services configuration to specify the LDAPBIND class profile.

```
[LDAP]
NumServers=1
BindProfile1=MY.LDAP.SERVER1
```

Optionally, default LDAP binding information can be defined in the PROXY segment of the IRR.PROXY.DEFAULTS profile in the FACILITY class.

```
RDEFINE FACILITY IRR.PROXY.DEFAULTS
  PROXY(LDAPHOST(ldap://some.ldap.host:389)
  BINDDN('cn=Joe User,ou=Poughkeepsie,o=IBM,c=US') BINDPW('MYPASS1'))
```

In this case, no BindProfile statement should appear in the PKI Services configuration file for that server. For more information, see [z/OS Cryptographic Services PKI Services Guide and Reference](#).

For information about storing keys that encrypt LDAP bind passwords, see [“Storing encryption keys using the KEYSMSTR class” on page 255.](#)

## LDAP event notification

LDAP event notification is used by IBM Tivoli Directory Integrator, in conjunction with password and password phrase envelopes (see Chapter 25, “Password and password phrase enveloping,” on page 623), to enable a heterogeneous password synchronization solution.

You can customize RACF to create LDAP change log entries in response to changes in user, group, and general resource profiles. This provides an open, remote method of change notification. An LDAP client can read the LDAP change log, detect updates to RACF users, groups, group membership, and general resources, and then retrieve RACF entries using only LDAP interfaces. To use this function, the LDAP server must be configured to support the SDBM backend. For details, see [Change logging in z/OS IBM Tivoli Directory Server Administration and Use for z/OS.](#)

Event notifications, through the creation of LDAP change log entries, are controlled by RACF resources in the RACFEVNT class. If RACFEVNT is active, and the appropriate resource is protected by either a discrete or generic profile, LDAP change log entries are created for the corresponding event types on a system-wide basis.

Table 39 on page 618 shows the name of the RACF resource in the RACFEVNT class used to control notifications for each type of supported change event.

<i>Table 39. LDAP event notification of RACF profile changes</i>	
<b>Resource in the RACFEVNT class</b>	<b>Change event type</b>
NOTIFY.LDAP.USER	Password and password phrase changes, regardless of the command or method used
	Updates to a user's revoke status (that is, changes to the FLAG4 field in the USER profile), regardless of the command or method used
	Users added using the ADDUSER command
	User modifications made using the ALTUSER or PASSWORD command
	Users deleted using the DELUSER command
NOTIFY.LDAP.GROUP	Groups added using the ADDGROUP command
	Group modifications made using ALTGROUP command
	Groups deleted using the DELGROUP command
NOTIFY.LDAP.CONNECT	Group membership changes made using any of the following commands: <ul style="list-style-type: none"> <li>• ALTUSER command, only when issued with GROUP, UACC, or AUTHORITY operand</li> <li>• CONNECT command</li> <li>• REMOVE command</li> </ul>
	Users established in their default groups using the ADDUSER command
NOTIFY.LDAP.class-name	General resources added using the RDEFINE command
	General resource modifications made using the RALTER command
	Changes made using the PERMIT command to the standard or conditional access list of a general resource
	General resource deletions made using the RDELETE command

Table 39. LDAP event notification of RACF profile changes (continued)

Resource in the RACFEVNT class	Change event type
<b>Notes:</b> <ul style="list-style-type: none"> <li>The RACF panels and the R_admin callable service (IRRSEQ00) internally issue TSO commands, so these interfaces are supported.</li> <li>The RACF panels can generate multiple commands while processing a user profile, and this might result in multiple change log entries.</li> <li>An application that updates supported profiles, using methods other than TSO commands, can create its own change log entry using the R_proxyserv callable service (IRRSPY00), documented in <a href="#">z/OS Security Server RACF Callable Services</a>.</li> </ul>	
<b>Restrictions:</b> <ul style="list-style-type: none"> <li>Other RACF commands that update user and general resource profiles, such as RACDCERT, RACLINK, and RACMAP are not supported.</li> <li>Commands issued from the RACF parameter library during RACF subsystem initialization are <i>not</i> logged because logging occurs only when the subsystem address space is fully functional. By contrast, parameter library commands issued as a result of a SET INCLUDE command <i>are</i> logged because the subsystem address space is initialized when it processes the SET command.</li> </ul>	

## LDAP change log entries

The LDAP change log entry contains information such as the change initiator, the affected user, group, or general resource, the type of update (add, modify, or delete), and the time and date of the change. It does *not* contain a list of the RACF profile fields that were changed nor does it contain the new values for these fields.

In the case of a change to the standard or conditional access list of a general resource, the changes attribute of the change log entry indicates that a general resource profile was added, modified or deleted. The changes attribute does *not* identify the user or group permission that was added, modified, or removed.

In the case of a password or password phrase change, the changes attribute of the change log entry identifies the password or password phrase field as the changed field. The changes attribute does *not* contain the actual password or password phrase value but contains one of the following values:

### **\*ComeAndGetIt\***

This indicates there is an encrypted password envelope or password phrase envelope that can be subsequently retrieved. (See [Chapter 25, “Password and password phrase enveloping,”](#) on page 623 for details about envelopes.)

### **\*NoEnvelope\***

This indicates there is no password envelope or password phrase envelope.

When other fields in a user's profile are changed in the same request that updates the values of the password and password phrase, three LDAP change log entries are created: one entry to log the password update, one to log the password phrase update, and another entry to log the information about the other changed fields. Removal of a user's password phrase does *not* create a separate log entry. (See Example 2.)

**Example 1:** An administrator issues the following command for a revoked user who is eligible for both password and password phrase enveloping.

```
ALTUSER userid PASSWORD(newpass) PHRASE(newphrase) RESUME OWNER(group-name)
```

If successful, this command causes *three* entries to be created to log the user profile changes.

- One entry identifies the password field as changed and contains the \*ComeAndGetIt\* value.

- A second entry identifies the password phrase field as changed and contains the \*ComeAndGetIt\* value.
- A third entry identifies changes in the user's revoke status and owning group.

**Example 2:** An administrator issues the following command to update a user's password, remove the password phrase, and change the user's name.

```
ALTUSER userid PASSWORD(newpass) NOPHRASE NAME(new-user-name)
```

If successful, this command causes *two* entries to be created to log the user profile changes.

- One entry identifies the password field as changed and contains the \*ComeAndGetIt\* value.
- A second entry contains information about the change in the user's name and removal of the password phrase.

**Example 3:** An administrator issues the following command to remove a user's password phrase and change the user's name.

```
ALTUSER userid NOPHRASE NAME(new-user-name)
```

If successful, this command causes *one* entry to be created to log the user profile changes. This entry contains information about the change in the user's name and removal of the password phrase.

For more information about the LDAP change log, see [Change logging in z/OS IBM Tivoli Directory Server Administration and Use for z/OS](#).

## LDAP notification occurs in real-time only

If the LDAP server is unavailable at the time the RACF change occurs, then that change log entry is lost and message IRRCL131I is issued to the console. There is currently no queuing mechanism whereby a change notification can be retried at a later point in time. This does not affect the RACF database itself; LDAP notification is attempted only after the RACF database has been updated.

To temporarily disable change notifications and suppress the informational messages while the LDAP server is unavailable, see “Disabling LDAP change notification” on page 621. If you have implemented password enveloping, also see “Disabling enveloping” on page 633.

## RRSF considerations for applications that exploit enveloping

Applications that exploit LDAP change log entries for registry synchronization should take network topology into account when propagating locally initiated RACF changes to other z/OS RACF systems in the network. In particular, if RACF is configured in an RRSF network and user profile, password, or password phrase updates are synchronized across RRSF nodes, then application deployment must include consideration of which propagation mechanism is used for specific types of changes to specific systems. Neglecting the interaction of the various propagation mechanisms could result in an unending cascade of updates for the same RACF change. For example, for an RRSF network that fully mirrors updates to user profiles, passwords, and password phrases, an LDAP based propagation mechanism should only communicate with a single RRSF node, and let that node propagate the change to other RACF nodes. Further, this RACF node should be the only node configured to perform LDAP event notification for user updates.

## Activating LDAP change notification

To activate LDAP change notification, define RACFEVNT class resources for the notifications you want and then activate the RACFEVNT class. (You need not define resources in the LDAPBIND class to enable LDAP change notifications.)

1. Define the RACFEVNT class resources for the LDAP notifications you want by creating one or more discrete profiles or by creating a generic profile.

There are two ways you might activate *all* supported LDAP notification types: defining multiple discrete profiles or defining one generic profile. Otherwise, define a subset of resources based on the LDAP notifications you want.

**Example showing multiple discrete profiles:**

```
RDEFINE RACFEVNT NOTIFY.LDAP.USER
RDEFINE RACFEVNT NOTIFY.LDAP.GROUP
RDEFINE RACFEVNT NOTIFY.LDAP.CONNECT
RDEFINE RACFEVNT NOTIFY.LDAP.FACILITY
```

**Example showing a generic profile:**

```
SETOPTS GENERIC(RACFEVNT)
RDEFINE RACFEVNT NOTIFY.LDAP.*
```

You might also define a generic profile to activate multiple general resource classes. The following example activates the JES-related classes called JESINPUT, JESJOBS, and JESSPOOL.

**Example:**

```
SETOPTS GENERIC(RACFEVNT)
RDEFINE RACFEVNT NOTIFY.LDAP.JES*
```

2. Activate the RACFEVNT class and optionally RACLIST it to improve performance.

```
SETOPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT)
```

## Disabling LDAP change notification

To temporarily disable LDAP change notification, issue the following command. Note that this command also disables password enveloping.

```
SETOPTS NOCLASSACT(RACFEVNT) NORACLIST(RACFEVNT)
```

To resume LDAP change notification and password enveloping, issue the following command.

```
SETOPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT)
```

To suppress LDAP change notifications *without* disabling enveloping, do not deactivate the RACFEVNT class. Instead, delete only the RACFEVNT profiles you defined in Step [“1”](#) on page 620 of [“Activating LDAP change notification”](#) on page 620.

**Example:**

```
RDELETE RACFEVNT NOTIFY.LDAP.*
SETOPTS RACLIST(RACFEVNT) REFRESH
```

To resume LDAP change notifications, follow the steps in [“Activating LDAP change notification”](#) on page 620.



## Chapter 25. Password and password phrase enveloping

This topic provides information about creating *password envelopes* and *password phrase envelopes* to enable authorized applications to recover user passwords and password phrases. Envelopes are used by IBM Tivoli Directory Integrator, in conjunction with LDAP notification (see [“LDAP event notification”](#) on page 618), to enable a heterogeneous password synchronization solution.

### Overview of enveloping

RACF can be configured to save user passwords and password phrases so that an authorized application can recover them in clear text. This ability can be restricted to a subset of your users and can be further limited to only passwords or password phrases.

When an eligible user's password or password phrase is changed, the new value is encrypted under a public key within a key ring associated with the user ID of the RACF subsystem address space. The encrypted value is then stored in the user's profile. When an application requests the password or password phrase, RACF decrypts the value, and then encrypts it in PKCS #7 format for recipients whose digital certificates have been placed on the same RACF key ring. An authorized application can then decrypt the *password envelope* or *password phrase envelope* using the recipient's private key.

The R\_Admin callable service (IRRSEQ00) provides the interface by which an application can retrieve an envelope. See [z/OS Security Server RACF Callable Services](#) for interface documentation, including a description of the envelope structure.

For the most part, new passwords and new password phrases are enveloped for an eligible user, with the following exceptions:

- Initial ADDUSER passwords and password phrases.
- When the new value of the password or password phrase is the same as the current value.
- When the ALTUSER or PASSWORD command is used to change the password, and the new password is equal to the user's default group name.
- When an application uses RACROUTE or ICHEINTY, rather than a RACF command, to set the password, and the password contains characters that are not accepted by the RACF commands.
- When an application uses RACROUTE or ICHEINTY to set the password and specifies ENCRYPT=NO.
- When an application uses ICHEINTY to set the password phrase but the password phrase does not have a valid (9 - 100 character) length.

### Resources that control enveloping

The PASSWORD.ENVELOPE and PASSPHRASE.ENVELOPE resources in the RACFEVNT class control whether new passwords and password phrases are enveloped for a given user. You can optionally control both password and password phrase enveloping using a single generic profile, such as PASS\*.ENVELOPE. If the user whose password or password phrase is changed has at least READ access to the appropriate resource, then the new password or password phrase is enveloped. Thus, you can use these resources to selectively authorize users whose passwords or password phrases will be enveloped. For example, you can exclude sensitive user IDs from both password and password phrase enveloping by authorizing those IDs to the PASS\*.ENVELOPE resource with access level NONE.

#### Restrictions:

- An enveloped password or password phrase is not displayed in the user's LISTUSER output. (The lines PASSWORD ENVELOPED=YES and PHRASE ENVELOPED=YES in the LISTUSER output indicates when a password or password phrase envelope is present. See [z/OS Security Server RACF Command Language Reference](#) for LISTUSER details.)

- An enveloped password or password phrase is not unloaded by the database unload (IRRDBU00) utility. (Certain fields in the output indicate that a password or password phrase envelope is present. See [z/OS Security Server RACF Macros and Interfaces](#) for details about IRRDBU00 output records.)
- No SMF records are created as a result of failed access checks to resources in the RACFEVNT class. You can set audit options in the resource profiles to log successes, and thus maintain a history of whose passwords and password phrases are enveloped.
- If the user fails verification (when the RACROUTE REQUEST=VERIFY is executed during envelope processing), the user's new password or password phrase is *not* enveloped, even when the password or password phrase change is successful. One possible reason for a verification failure (during envelope processing) is that the user is *revoked* at the time that envelope processing occurs.

For example, if an administrator uses the ALTUSER command to change the password of a revoked user who is eligible for password enveloping, the user's password is changed but the user's password is *not* enveloped. Even when the administrator subsequently resumes the revoked user, the password is *not* enveloped.

To envelope the password or password phrase of an eligible user who is revoked, you must resume the user before the change, or resume the user with the same ALTUSER command that changes the password or password phrase.

### Example (correct):

```
ALTUSER userid PASSWORD(new-password) PHRASE(new-password-phrase) RESUME
```

### Example (correct):

```
ALTUSER userid RESUME
ALTUSER userid PASSWORD(new-password) PHRASE(new-password-phrase)
```

### Example (incorrect):

```
ALTUSER userid PASSWORD(new-password) PHRASE(new-password-phrase)
ALTUSER userid RESUME
```

When you use the correct examples, the revoked user's new password and password phrase are enveloped.

## Signing hash algorithm and encryption strength used to create the envelope

Both the signing hash algorithm and encryption strength are configurable attributes. Use application data (APPLDATA) in the RACFEVNT resource profiles to specify the signing hash algorithm that signs the PKCS #7 envelope, and the encryption strength used when encrypting the envelope. The syntax of the APPLDATA string consists of a character string indicating the signing hash algorithm, followed by a forward slash (/), followed by a string indicating the encryption strength.

### Examples:

```
RDEFINE RACFEVNT PASSPHRASE.ENVELOPE UACC(NONE) APPLDATA('MD5/STRONG')
RALTER RACFEVNT PASSWORD.ENVELOPE APPLDATA('MD5/STRONG')
```

```
RDEFINE RACFEVNT PASSPHRASE.ENVELOPE UACC(NONE)
  APPLDATA('x509_alg_sha512Digest/x509_alg_aesCbc256')
RALTER RACFEVNT PASSWORD.ENVELOPE
  APPLDATA('x509_alg_sha512Digest/x509_alg_aesCbc256')
```

Allowable values for the signing hash algorithm:

- x509\_alg\_sha256Digest
- x509\_alg\_sha384Digest
- x509\_alg\_sha512Digest
- MD5 (default)
- SHA1



Allowable values for the encryption strength:

- x509\_alg\_aesCbc256
- STRONG (default)
- MEDIUM
- WEAK

**Guideline:** Use the strongest encryption possible. If you must use weaker encryption, for example, due to export regulations, then protect yourself against offline attacks by carefully controlling access to the RACF database and any other repository where envelopes might be stored after being retrieved from RACF.

Encryption strength value	Data encryption method
x509_alg_aesCbc256	AES (a 256-bit encryption key)
STRONG	Triple DES (a 168-bit encryption key)
MEDIUM	DES (a 56-bit encryption key)
WEAK	RC2 (a 40-bit encryption key)

**Note:** Strong Some encryption methods may not be available at all installations based on government export regulations. See [z/OS Cryptographic Services System SSL Programming](#) for more information.

If the APPLDATA is not specified in the profile, the defaults are taken as noted above. If an empty qualifier exists in the APPLDATA, then the default value is used for that qualifier. For example, if the APPLDATA is specified as SHA1, then SHA1 is used as the signing hash algorithm, and triple DES is used as the encryption algorithm. If the APPLDATA is specified as /MEDIUM, then MD5 is used as the signing hash algorithm, and DES is used as the encryption algorithm.

If the APPLDATA is specified incorrectly, an error message is issued to the console. Thereafter, the default values are used whenever users who are eligible for enveloping change their passwords or password phrases, or whenever an application requests the retrieval of an envelope.

The APPLDATA can be changed at any time.

## The IRR.PWENV.KEYRING key ring

IRR.PWENV.KEYRING is the name of a key ring associated with the identity of the RACF subsystem address space (RASP). It contains a certificate with private key for the RASP itself. This certificate is used to encrypt new passwords and password phrases for eligible users. It is also used to decrypt the stored passwords and password phrases when a PKCS #7 envelope is retrieved by an authorized application, and to sign the contents of the returned envelope.

IRR.PWENV.KEYRING also contains certificates of all the principals who are intended to retrieve a user's changed password or password phrase from RACF. Changed passwords and password phrases are encrypted using the public keys contained within these certificates. RACF encrypts passwords and password phrases for up to 20 certificates on this key ring.

## Controlling envelope retrieval

Only applications running in system key or supervisor state can use the R\_admin callable service (IRRSEQ00) to retrieve envelopes. In addition, applications must have access to the appropriate resource in the FACILITY class.

The following resources in the FACILITY class control the retrieval of envelopes from RACF by applications invoking the R\_admin callable service (IRRSEQ00).

Resource name	Controls retrieval of ...
IRR.RADMIN.EXTRACT.PWENV	Only password envelopes
IRR.RADMIN.EXTRACT.PPENV	Only password phrase envelopes

Resource name	Controls retrieval of ...
IRR.RADMIN.EXTRACT.*	Both password and password phrase envelopes

You can set the audit options for these resources to log successes, and thus maintain a history of whose passwords and password phrases are retrieved, and by whom. Failures can also be logged. (The log string identifies the user whose password or password phrase was retrieved.)

## The NOTIFY.LDAP.USER resource

When a resource is defined in the RACFEVNT class called NOTIFY.LDAP.USER, an LDAP change log entry is created when a user's password or password phrase is changed. See [“LDAP event notification” on page 618](#) for details.

## Setting up enveloping

There are several RACF setup steps to perform in order for recipients to be able to retrieve user password and password phrase changes. See the setup steps in the following topics:

1. [“Preparing the address space of the RACF subsystem” on page 626](#)
2. [“Generating a local CA certificate using RACF as the CA” on page 627](#)
3. [“Generating an X.509 V3 certificate for the RACF address space” on page 627](#)
4. [“Generating an X.509 V3 certificate for the envelope recipient” on page 628](#)
5. [“Copying the certificates to the host system \(if generated elsewhere\)” on page 630](#)
6. [“Exporting RACF's certificate to the recipient key database” on page 631](#)
7. [“Authorizing the envelope recipient” on page 631](#)
8. [“Activating enveloping” on page 632.](#)

**Tip:** While you are initially configuring and testing this function, check the console for error messages. As RACF detects errors during the enveloping process, they will be reported to the console, not to the end user who initiates a password or password phrase change.

## Preparing the address space of the RACF subsystem

- Add an OMVS segment to the user ID and an OMVS segment to the default group of the RACF subsystem address space. Use the output of the SET LIST command to identify the user ID of the RACF subsystem.

You can specify the UID and GID values of your choice by explicitly assigning a unique UID with the UID operand of the ALTUSER command, and by explicitly assigning a GID using the GID operand of the ALTGROUP command.

Alternatively, use the AUTOUID and AUTOGID keywords to automatically assign a unique UID and GID. (For setup instructions, see [“Enabling automatic assignment of unique UNIX identities” on page 518.](#)) For example, if the RACF subsystem runs under the user ID RACFSUB whose default group is STCGRP, execute the following commands to add OMVS segments:

### Example:

```
ALTUSER RACFSUB OMVS(AUTOUID HOME(/) PROGRAM(/bin/sh))
ALTGROUP STCGRP OMVS(AUTOGID)
```

- If the RACF subsystem identity does not have the TRUSTED or PRIVILEGED attribute, it will require the necessary FACILITY class authorization in order to extract certificates from a key ring. (The certificate setup is described in [“Generating an X.509 V3 certificate for the RACF address space” on page 627.](#))

```
RDEFINE FACILITY IRR.DIGTCERT.LISTRING UACC(NONE)
PERMIT IRR.DIGTCERT.LISTRING CLASS(FACILITY) ID(RACFSUB) ACCESS(READ)
```

You might already be protecting this resource, perhaps with a generic profile. Modify this step as needed for your environment.

**Guideline:** If your installation uses RACF remote sharing facility (RRSF), assign the TRUSTED attribute to the RACF address space user ID.

## Generating a local CA certificate using RACF as the CA

If you want to use RACF as your certificate authority (CA), create a CA certificate, if you have not already done so. The following command creates a certificate authority (or signing) certificate identified by the label RACFCA that will be used to create subsequent certificates, such as the certificate that RACF will use during the enveloping process. Command values, such as `subjectsdn`, `organization`, and `country`, can be modified to reflect the naming structure and conventions used at your installation.

```
RACDCERT CERTAUTH GENCERT SUBJECTSDN(CN('RACFCA') O('ibm') C('us'))
WITHLABEL('RACFCA') NOTAFTER(DATE(2020-12-31)) ICSF
```

**Guideline:** Use ICSF to store private keys, as shown in the RACDCERT CERTAUTH example. If you do not use ICSF, then omit this operand from the command.

**Note:** Certificates signed with this local CA certificate show an issuer of `cn=RACFCA,o=ibm,c=us` when listed with the RACDCERT LIST command.

RACF signs the envelope so that a recipient can verify that the envelope signature was created using RACF's certificate (created in “Generating an X.509 V3 certificate for the RACF address space” on page 627). If the recipient also wants to check the veracity of RACF's certificate, this CA certificate is needed.

## Generating an X.509 V3 certificate for the RACF address space

You must generate an X.509 V3 certificate and associated private key for the RACF address space. You must also add the certificate to RACF's key ring as the default certificate so that RACF uses it during the enveloping process.

### Steps for generating a certificate and private key for the RACF address space

#### Before you begin:

- The RACDCERT commands shown in the following steps are examples. Your values might be different.
  - In this example, the user ID of the RACF address space is RACFSUB. The actual user ID for your RACF address space is defined by setting up a started task identity using either the started procedures table (ICHRIN03), or by defining a profile in the STARTED class.
 

See [z/OS Security Server RACF System Programmer's Guide](#) for information about setting up the RACF subsystem.
  - The label of the new certificate generated in these steps is RASP1. You can specify a label of your own choice.
  - The new certificate in these steps is signed by RACF, which is the certificate authority used in the context of this example. The local CA certificate for RACF is identified by the label RACFCA.
- After completing these steps, avoid changing RACF's private key. If you change it, RACF will not be able to build PKCS #7 envelopes for existing passwords or password phrases. (Because the passwords and password phrases were encrypted under the old public key, they cannot be decrypted under the new private key.) Normal operation will resume as users subsequently change their passwords and password phrases.

**Guidelines:** If available, use ICSF to store the private key created in Step “1” on page 628. Unless you use ICSF to store private keys, any user with READ access to the RACF database, or a user authorized to invoke a RACROUTE REQUEST=EXTRACT request, can obtain the default certificate's private key and any user's encrypted password or password phrase. As always, protect the RACF database and its copies against inappropriate access. Further, verify that applications retrieving envelopes do so using only the R\_admin interface.

Perform the following steps to generate an X.509 V3 certificate and associated private key, and prepare them for RACF use during the enveloping process.

1. Generate a digital certificate containing a private key for the RACF address space.

**Example:**

```
RACDCERT ID(RACFSUB) GENCERT
  SUBJECTSDN(CN('RACF AddrSpace System 1')O('ibm')C('us'))
  WITHLABEL('RASP1')
  SIGNWITH(CERTAUTH LABEL('RACFCA'))
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
  NOTAFTER(DATE(2020-12-31))
  ICSF
```

If you do not use ICSF, then omit this operand from the command.

- 
2. Create a RACF key ring named IRR.PWENV.KEYRING. Note that the name of the key ring is case-sensitive.

**Example:**

```
RACDCERT ID(RACFSUB) ADDRING(IRR.PWENV.KEYRING)
```

- 
3. Connect the certificate you created for the RACF address space in Step “1” on page 628 to the key ring. You must connect it as the default certificate as shown in the following example.

**Example:**

```
RACDCERT ID(RACFSUB) CONNECT(LABEL('RASP1')
  RING(IRR.PWENV.KEYRING)
  DEFAULT
  USAGE(PERSONAL))
```

- 
4. Verify that your new certificate is marked trusted by listing it using the RACDCERT LIST command. (This also applies to the other certificates you create during setup for enveloping.) If the certificate is not trusted, use the following command to mark it trusted.

**Example:**

```
RACDCERT ID(RACFSUB) ALTER (LABEL('RASP1')) TRUST
```

---

You have now created an X.509 V3 certificate and associated private key for the RACF address space, and connected them to RACF's key ring.

Once you complete these steps, if you change the user ID under which the RACF subsystem runs, you will need to repeat these steps using the new RACF user ID.

## Generating an X.509 V3 certificate for the envelope recipient

During the enveloping process, RACF encrypts data that can be recovered only by the intended recipient of that data. An intended recipient, such as the identity of the IBM Tivoli Directory Integrator process running on a non-z/OS platform, is identified by an X.509 V3 certificate. Certificates that identify intended recipients of the envelopes must be connected to the key ring IRR.PWENV.KEYRING associated with the user ID of the RACF subsystem address space. Note that these certificates are used only to encrypt password and password phrase information; they are not used to bind to LDAP or to authenticate to RACF.

**Guideline:** In general, authorize only trusted *applications*, not users, to extract envelopes.

Certificates for intended recipients might be created by RACF, and exported to non-z/OS processes, for instance. The creation of the certificates might be accomplished using the following sample RACDCERT commands that generate certificates for IDI1 and APP2 (the identities of processes that are authorized to retrieve envelopes). These certificates are signed with the local CA (RACF) certificate that is identified by the label RACFCA.

**Example:**

```
RACDCERT ID(IDI1) GENCERT
  SUBJECTSDN(CN('IBM Tivoli Directory Integrator Server 1') O('ibm') C('us'))
  WITHLABEL('IDI1') SIGNWITH(CERTAUTH LABEL('RACFCA'))
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
```

```
RACDCERT ID(APP2) GENCERT
  SUBJECTSDN(CN('Application Server 2') O('ibm') C('us'))
  WITHLABEL('APP2') SIGNWITH(CERTAUTH LABEL('RACFCA'))
  KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
```

**Rule:** The KEYUSAGE attributes HANDSHAKE, DATAENCRYPT and DOCSIGN must be specified for the certificates.

You might also create certificates directly on the recipient platform and import them into RACF. Any key management system can be used to create the recipient key pair and certificate, as long as it can export certificates in an industry standard format understood by the RACDCERT command.

The following example uses **keytool**, the key management tool available with many Java virtual machines (JVMs), including the JVM shipped with IBM Tivoli Directory Integrator.

You might not need to perform all of the steps below if you already have a key management infrastructure. The examples below assume you are starting new.

Create certificates:

Use of RSA as a key algorithm is required. If **keytool** uses the default value DSA, you must change it to RSA.

**Example:**

```
keytool -genkey -alias IDI1 -keypass xxxxxx
  -storepass xxxxxx -keystore recipient.jks
  -dname "CN=(IBM Directory Tivoli Integrator Server 1),O=(ibm),C=(us)"
  -keyalg RSA
```

```
keytool -genkey -alias APP2 -keypass xxxxxx
  -storepass xxxxxx -keystore recipient.jks
  -dname "CN=(Application Server 2),O=(ibm),C=(us)"
  -keyalg RSA
```

The **keytool** commands above create two self-signed certificates. For example purposes, this is sufficient. In a production environment, you might want to use something other than a self-signed certificate.

Export certificates:

The **rfc** keyword specifies base64-encoded output.

**Example:**

```
keytool -export -alias IDI1 -file IDI1.b64 -keystore recipient.jks -storepass
  xxxxxx -rfc
keytool -export -alias APP2 -file APP2.b64 -keystore recipient.jks -storepass
  xxxxxx -rfc
```

The following example uses IBM Key Management, running on Windows 2000. The commands are slightly different from those in the previous example, but the procedure is the same.

Create key database:

**Example:**

```
gsk5cmd.exe -keydb -create -db recipient.kdb -pw xxxx
```

Create certificates:

**Example:**

```
gsk5cmd.exe -cert -create -db recipient.kdb -pw xxxx -label IDI1 -dn  
"CN=(IBM Tivoli Directory Integrator Server 1),O=(ibm),C=(us)"
```

```
gsk5cmd.exe -cert -create -db recipient.kdb -pw xxxx -label APP2 -dn  
"CN=(Application Server 2),O=(ibm),C=(us)"
```

The **gsk5cmd** commands above create two self-signed certificates. For example purposes, this is sufficient. In a production environment, you might want to use something other than a self-signed certificate.

Export certificates:

**Example:**

```
gsk5cmd.exe -cert -extract -db recipient.kdb -pw xxxx -label  
"IDI1" -target IDI1.b64 -format ascii  
  
gsk5cmd.exe -cert -extract -db recipient.kdb -pw xxxx -label  
"APP2" -target APP2.b64 -format ascii
```

At this point, using either example above, the contents of the certificates are contained in the files IDI1.b64 and APP2.b64.

## Copying the certificates to the host system (if generated elsewhere)

Copy the certificates to the host system. Because the certificates were exported in base64, they can be cut and pasted into a host file, using a text editor. If you use FTP, transfer them using ASCII (not binary) mode. The files should start with -----BEGIN CERTIFICATE----- and end with -----END CERTIFICATE----- when viewed on the host. For this example, the files are copied to CERT.IDI1.TEXT and CERT.APP2.TEXT.

Import the certificates in RACF using the RACDCERT command:

**Example:**

```
RACDCERT ID(IDI1) ADD(CERT.IDI1.TEXT) TRUST WITHLABEL('IDI1')  
RACDCERT ID(APP2) ADD(CERT.APP2.TEXT) TRUST WITHLABEL('APP2')
```

If you created self-signed certificates, RACF will warn that the certificate authority is not defined to RACF, but will properly import the certificates.

Connect the recipient certificates to the key ring. RACF will encrypt the password or password phrase for up to 20 recipient certificates:

**Example:**

```
RACDCERT ID(RACFSUB) CONNECT(ID(IDI1) LABEL('IDI1')  
RING(IRR.PWENV.KEYRING) USAGE(PERSONAL))  
RACDCERT ID(RACFSUB) CONNECT(ID(APP2) LABEL('APP2')  
RING(IRR.PWENV.KEYRING) USAGE(PERSONAL))
```

RACF constructs the PKCS #7 envelope at the time the envelope is requested. So, if you add a certificate for a principal, that principal will be able to decrypt envelopes for any eligible user whose current password or password phrase has already been changed (assuming the principal has the authorization described in the next step). Likewise, if a certificate is removed from the key ring, the principal will not be able to decrypt envelopes, even if the password or password phrase was changed while the certificate was on the ring, and even if the authorization described in the next step is not removed for the principal.

## Exporting RACF's certificate to the recipient key database

If you have implemented IBM Tivoli Directory Integrator, or the recipient intends to verify the signature of envelopes as they are decrypted to ensure they are from RACF, then both the CA certificate and the RACF address space certificate must be available to the recipient in the recipient key database.

Export both the RACF CA certificate and the RACF address space certificate. (These certificates were created in [“Generating a local CA certificate using RACF as the CA”](#) on page 627 and [“Generating an X.509 V3 certificate for the RACF address space”](#) on page 627.)

### Example:

```
RACDCERT CERTAUTH EXPORT(LABEL('RACFCA')) DSN(CERT.RACFCA.TEXT) FORMAT(CERTB64)
RACDCERT ID(RACFSUB) EXPORT(LABEL('RASP1')) DSN(CERT.RASP.TEXT) FORMAT(CERTB64)
```

These files must be transferred to the recipient system. Use FTP (ASCII mode) or simply cut and paste them to create the `racfca.cert` and `rasp.cert` files. Then, import the files:

Using the **keytool** command:

### Example:

```
keytool -import -alias RACFCA -file racfca.cert -keystore
recipient.jks -storepass xxxxxx
keytool -import -alias RASP1 -file rasp.cert -keystore
recipient.jks -storepass xxxxxx
```

Using the **gsk5cmd** command:

### Example:

```
gsk5cmd.exe -cert -add -db recipient.kdb -pw xxxx -label
"RACFCA" -file racfca.cert -format ascii

gsk5cmd.exe -cert -add -db recipient.kdb -pw xxxx -label
"RASP1" -file rasp.cert -format ascii
```

## Authorizing the envelope recipient

Authorize these same principals to the `R_admin` function (to retrieve envelopes from RACF) using one of the following examples. Example 1 allows you to separately control retrieval of password envelopes and password phrase envelopes. Example 2 allows you to control retrieval of both password envelopes and password phrase envelopes using the same resource.

The FACILITY resource names shown in these examples are described in [“Controlling envelope retrieval”](#) on page 625.

### Example 1:

```
RDEFINE FACILITY IRR.RADMIN.EXTRACT.PWENV UACC(NONE)
PERMIT IRR.RADMIN.EXTRACT.PWENV CLASS(FACILITY) ID(IDI1 APP2) ACCESS(READ)

RDEFINE FACILITY IRR.RADMIN.EXTRACT.PPENV UACC(NONE)
PERMIT IRR.RADMIN.EXTRACT.PPENV CLASS(FACILITY) ID(IDI1 APP2) ACCESS(READ)
```

### Example 2:

```
RDEFINE FACILITY IRR.RADMIN.EXTRACT.* UACC(NONE)
PERMIT IRR.RADMIN.EXTRACT.* CLASS(FACILITY) ID(IDI1 APP2) ACCESS(READ)
```

**Guideline:** In general, authorize only trusted *applications*, not *users*, to extract envelopes.

Failed access attempts to these resources are logged by default. The log string of the audit record contains the user ID whose envelope is being retrieved. If you use a generic profile (shown in Example 2), look for the resource name in the audit record and you can distinguish whether a password envelope or password phrase envelope was retrieved.

**Guideline:** Given the sensitive nature of this function, you should log successful accesses as well. For example, a user with the RACF AUDITOR attribute can execute the following command:

```
RALTER FACILITY IRR.RADMIN.EXTRACT.* GLOBALAUDIT(ALL(READ))
```

If the FACILITY class is already ACTIVE and RACLISTed, refresh the class.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

## Activating enveloping

This procedure involves defining resources in the RACFEVNT class, and activating this class.

1. Define the RACFEVNT resources to control enveloping. You can define each resource separately (PASSWORD.ENVELOPE and PASSPHRASE.ENVELOPE) or define a generic profile (shown in Example 2) to control both. The APPLDATA field of this profile specifies the hash algorithm to use when signing the PKCS #7 envelope, and the encryption strength to use when encrypting the envelope. The following examples specify the strongest signing and encryption:

### Example 1:

```
RDEFINE RACFEVNT PASSWORD.ENVELOPE UACC(NONE) APPLDATA('MD5/STRONG')  
RDEFINE RACFEVNT PASSPHRASE.ENVELOPE UACC(NONE) APPLDATA('MD5/STRONG')
```

### Example 2:

```
SETROPTS GENERIC(RACFEVNT)  
RDEFINE RACFEVNT PASS*.ENVELOPE UACC(NONE) APPLDATA('MD5/STRONG')
```

**Example 3:** If you previously implemented password enveloping by defining the PASSWORD.ENVELOPE resource and want to add support for password phrases using a generic profile (shown in Example 2), you can model the new generic profile on your previously defined resource and then delete the old profile. For example:

```
SETROPTS GENERIC(RACFEVNT)  
RDEFINE RACFEVNT PASS*.ENVELOPE FROM(PASSWORD.ENVELOPE)  
RDELETE RACFEVNT PASSWORD.ENVELOPE
```

### Guidelines:

- Define separate discrete profiles (as shown in Example 1) if you need the flexibility to envelope only passwords or only password phrases. Using separate profiles also allows you to authorize a set of users eligible for password enveloping and a different set eligible for password phrase enveloping. (See Step “2” on page 632 for information about authorizing users.)
  - In general, avoid assigning UACC(READ) to the resources that control enveloping because UACC(READ) enables enveloping for all users (except RESTRICTED users), including highly privileged users such as system-SPECIAL users and user IDs that you use for emergency recovery purposes. Consider assigning UACC(READ) *only* if you specifically authorize sensitive user IDs with ACCESS(NONE). This prevents their passwords and password phrases from being enveloped.
2. Enable enveloping for users whose passwords or password phrases are to be encrypted for the intended recipients (whose digital certificates are connected to the IRR.PWENV.KEYRING key ring). This is done by permitting a given user or group to the appropriate envelope resources in the RACFEVNT class.

### Example 1:

```
PERMIT PASSWORD.ENVELOPE CLASS(RACFEVNT) ID(USER1 USER2 GROUPA)  
ACCESS(READ)  
PERMIT PASSPHRASE.ENVELOPE CLASS(RACFEVNT) ID(USER1 USER2 GROUPA GROUPB)  
ACCESS(READ)
```



**Example 2:**

```
PERMIT PASS*.ENVELOPE CLASS(RACFEVNT) ID(USER1 USER2 GROUPA GROUPB)
ACCESS(READ)
```

**Restrictions:**

- If you specified UACC(READ) in Step “1” on page 632, you must specifically permit restricted users with ACCESS(READ) by user ID or group if you want their passwords or password phrases to be enveloped. (Restricted user IDs are not authorized by the UACC in profiles.)
- Protected user IDs are not eligible for enveloping because they have no passwords or password phrases.

If you specified UACC(NONE) in Step “1” on page 632, you must include newly added users and groups into the access scheme so that passwords and password phrases are properly enveloped or not, over time. In particular, you might want to have intended group connections in place for all new users before their initial logons when they must change their passwords and password phrases. Also, keep in mind that if a user is connected to several groups, his effective authority is the highest allowed by any of his groups (assuming he isn't specifically permitted by user ID, in which case this authority overrides that granted by any of his groups).

For example, if list-of-groups processing (SETOPTS GRPLIST option) is active, and user BOB is connected to groups GROUP1 and GROUP2, and GROUP1 is permitted to PASSWORD.ENVELOPE with ACCESS(NONE), GROUP2 is permitted with ACCESS(READ), and BOB is not explicitly permitted, then BOB's effective access to PASSWORD.ENVELOPE is READ, and BOB's password will be enveloped.

3. Optionally, activate LDAP change log notification for USER profile changes so an LDAP change log entry is created whenever a user's new password or password phrase is enveloped. This step is required if you use an application like IBM Tivoli Directory Integrator to implement a heterogeneous password synchronization solution.

**Example:**

```
RDEFINE RACFEVNT NOTIFY.LDAP.USER UACC(READ)
```

Optionally, you can use a generic profile to activate LDAP change log notification. See “[LDAP event notification](#)” on page 618 for details.

4. Enable enveloping by activating the RACFEVNT class. It can be RACLISTed to improve performance but this is not required.

```
SETOPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT)
```

5. Stop and restart the RACF subsystem address space after defining the needed enveloping resources in the RACFEVNT class and activating the RACFEVNT class. If the RACF subsystem address space is already up and running when you configure enveloping and you do not stop and restart the address space, it will not have the proper environment set up to perform the function and will fail any request to envelope passwords or password phrases.

## Disabling enveloping

If you delete the generic profile or the resources that control enveloping in the RACFEVNT class (defined in Step “1” on page 632), you disable enveloping. However, when you delete these resources, RACF does *not* automatically delete your existing envelopes, nor can you directly delete them using the RACF commands. If you want to remove existing envelopes from user profiles to minimize the size of your RACF database, perform the following steps.

To temporarily disable enveloping, do not perform these steps. Instead, issue SETOPTS NOCLASSACT(RACFEVNT) NORACLIST(RACFEVNT). This command disables LDAP change notification and enveloping without removing existing envelopes. To resume enveloping, issue SETOPTS CLASSACT(RACFEVNT) RACLIST(RACFEVNT).

To temporarily disable enveloping without removing existing envelopes and without disabling LDAP change notification, follow only Step “2” on page 634 of the following steps. However, before doing so, make note of the access list entries or consider copying the access list to a new profile for later use. For example, issue: RDEFINE RACFEVNT STASHPROF FROM(PASS\*.ENVELOPE)

Perform the following steps to prepare RACF to systematically delete (during an interim time period that you determine) each existing envelope when the user's password or password phrase is changed. These steps show command examples using generic profiles. If you defined individual resource names when you implemented enveloping, modify the commands shown.

## Steps for disabling enveloping and deleting existing envelopes

Perform the following steps to remove existing password envelopes and password phrase envelopes from user profiles, and disable enveloping.

1. Delete the IRR.RADMIN.EXTRACT.\* resource in the FACILITY class and refresh the FACILITY class. This prevents applications that use the R\_admin callable service (IRRSEQ00) from retrieving envelopes.

### Example:

```
RDELETE FACILITY IRR.RADMIN.EXTRACT.*
SETROPTS RACLIST(FACILITY) REFRESH
```

- 
2. Alter the PASS\*.ENVELOPE profile to update the UACC to NONE (if not already specified) and remove all access authorizations. Then, refresh the RACFEVNT class. This step disallows password and password phrase enveloping for all users.

### Examples:

```
RALTER RACFEVNT PASS*.ENVELOPE UACC(NONE)
PERMIT PASS*.ENVELOPE CLASS(RACFEVNT) RESET

SETROPTS RACLIST(RACFEVNT) REFRESH
```

- 
3. Determine the current change interval for passwords and password phrases by inspecting the password processing options listed in the output of the SETROPTS LIST command.

### Sample output:

```
PASSWORD PROCESSING OPTIONS:
PASSWORD CHANGE INTERVAL IS 180 DAYS.
```

- 
4. Choose the length of your interim period (for instance, 240 days) to allow the change interval to elapse while the RACFEVNT class continues to remain active. Consider a time period long enough to maximize the number of users who will be active.

During this period, user passwords and password phrases will expire and the users who log on will be forced to change them to new values. Because you disallowed enveloping for all users in Step 2, RACF will not envelope the new values and will systematically delete existing envelopes.

- 
5. (Optional) Before the end of your interim period, gauge your progress by running the IRRDBU00 utility (see “Using the RACF database unload utility (IRRDBU00)” on page 371) to report on users who still have envelopes. (Envelopes will remain for users who are inactive during your interim period.) Revise the length of your interim period if needed.
-

6. At the end of your interim period, disable enveloping for passwords and password phrases by deleting the PASS\*. ENVELOPE profile and refreshing the RACFEVNT class.

```
RDELETE RACFEVNT PASS*.ENVELOPE
SETROPTS RACLIST(RACFEVNT) REFRESH
```

7. If you do not need the RACFEVNT class for LDAP event notification (see “LDAP event notification” on page 618), deactivate the RACFEVNT class and remove any RACLISTed profiles for the RACFEVNT class.

```
SETROPTS NOCLASSACT(RACFEVNT) NORACLIST(RACFEVNT)
```

8. Delete the RACF key ring you created for enveloping. Also, delete any unneeded certificates you created for enveloping.

**Example:**

```
RACDCERT ID(RACFSUB) DELRING(IRR.PWENV.KEYRING)
```

When you are finished, you have disabled enveloping for both passwords and password phrases in a manner that allowed RACF to systematically delete existing envelopes. If some users were inactive during your interim period, their envelopes still remain.

## Planning considerations for heterogeneous password synchronization

Before implementing enveloping, carefully weigh the risks against the benefits. RACF has always implemented one-way encryption when storing new user passwords and password phrases. This implementation makes it impossible for even a system administrator to obtain a user's password or password phrase once that user has changed the initial logon value. This implementation protects users against unauthorized use of their passwords and password phrases, and increases system accountability. Implementing the enveloping function makes it possible for an authorized user or application to retrieve a user's current password and password phrase. Subsequent use of this password or password phrase will result in a loss of accountability, resulting in the question: *Who actually entered the user ID and password or password phrase and is now working under the user's identity?* A RACF administrator currently has the capability of simply changing the user's password or password phrase, and then logging on as that user. However, when this occurs, the user will become aware at next logon time that his password or password phrase was changed.

Looking at a wider view, IBM Tivoli Directory Integrator uses enveloping to implement a heterogeneous password synchronization solution. While password synchronization is a topic somewhat outside the scope of RACF, it can be considered a security exposure that reduces the security of your enterprise. z/OS has traditionally been viewed as a highly secure platform, in much part due to the security of user passwords. In an heterogeneous environment, a password synchronization application can open up a z/OS system to a successful hack from any other platform, such as Windows or UNIX.

On the other hand, password synchronization can be viewed as a usability enhancement. Users might have many accounts on many different systems. Therefore, managing the different passwords that have different syntax requirements, as well as different expiration dates, can have a significant impact to user productivity. This complexity can itself lead to a loss of overall security because users might be tempted to write down their passwords, or select *easy-to-guess* passwords, in an attempt to manage the complexity.

There are other solutions that perform password synchronization in which z/OS is a participating platform. Such applications use RACF exits to intercept password changes. The PKCS #7 enveloping function, in conjunction with LDAP event notification, provides a simpler way for such applications to subscribe

to password and password phrase changes, but does not by itself provide a higher or lower degree of security than is already put in place by such applications. Ultimately, you must rely on the application to maintain the security of RACF passwords and password phrases from the point those values are intercepted. For example, the application should not send a password or password phrase across a network in clear text and should protect any repository that might contain these values in clear text.

It is the installation's choice to evaluate password synchronization software, and enable PKCS #7 enveloping in support of such software. Part of deploying such software is ensuring proper user education and network security. Several RACF implementation features help to minimize the risks:

- As with all new RACF enhancements, enveloping is not enabled by default. You must enable it before any passwords or password phrases are enveloped and retrievable.
- Public key cryptography protects the envelopes as they are sent across the network, and makes them available only to authorized principals whose certificates you connect to a RACF key ring. Using RACF key rings to authorize principals keeps the trust policy within your scope, as the RACF security administrator.
- The encryption and retrieval of envelopes is fully dynamic with respect to key ring changes. (The envelope is created on demand when it is requested, using the current contents of the key ring.) Therefore, if the private key of a recipient is compromised, you can remove the recipient's certificate from the RACF key ring to dynamically render envelopes indecipherable to the thief of the recipient's private key. Even if the thief is able to masquerade as the intended recipient, bind to LDAP, and retrieve an envelope under the recipient's identity, he will not be able to decipher it using the stolen private key.
- Profiles in the RACFEVNT class establish your enveloping policy. This policy allows you to scope password and password phrase enveloping to a subset of RACF users, allowing you to exclude sensitive user IDs, and control enveloping for passwords and password phrases independently, at your discretion.
- Policy changes are logged to SMF for each privileged operation associated with enveloping, such as changes to the key ring, changes to the enveloping policy, and actual retrievals of envelopes from RACF.

## Chapter 26. Defining and using custom fields

This topic describes how to define and use custom fields. It contains a task roadmap ([“Task roadmap for defining and using custom fields”](#) on page 638) and detailed implementation instructions.

### Overview of custom fields

Custom fields are fields within the RACF database that you customize to store security information about the RACF profiles at your installation. You can tailor the names and attributes of custom fields. Once you define custom fields, use RACF commands, such as the ALTUSER and ALTGROUP to add data to custom fields.

For each custom field, you can customize the following attributes:

- The name of the custom field, which is used as the RACF command operand for TSO/E commands.
- The data type for the custom field. Choose character, numeric, hexadecimal, or flag (YES or NO) fields.
- The help text for each custom field.
- The output heading for the listing commands:
  - LISTUSER
  - LISTGRP
  - RLIST
  - LISTDSD
- The acceptable values for the data in each field based on data type. You can customize several options, including the following:
  - For character fields, you can customize maximum length, restrict the character contents, and allow mixed-case characters.
  - For numeric fields, you can customize maximum value and minimum value.
  - For hexadecimal fields, you can customize the maximum length.

Your installation can provide additional customization by tailoring exit routines to validate custom field data. For details, see [Custom field validation exit \(IRRVAF01\)](#) in *z/OS Security Server RACF System Programmer's Guide*. Alternatively, you can implement your validation in a System REXX exec. The name of the exec is specified in the custom field definition. For more information on writing the REXX exec, see [VALREXX](#) in *z/OS Security Server RACF System Programmer's Guide*.

Define custom fields and their attributes for profiles using the RDEFINE command. Each custom field and its attributes is stored in the CFDEF segment of a general resource profile in the CFIELD class. (For details about naming the CFIELD profiles that define your custom fields, see [“Profiles in the CFIELD class”](#) on page 638.)

Add custom field data to RACF profiles using:

- The ADDUSER or ALTUSER command for user profiles.
- The ADDGROUP or ALTGROUP command for group profiles.
- The ADDSD or ALTDSD command for data set profiles.
- The RALTER or RDEFINE command for general resource profiles.

For example, if a custom field named DIVISION is defined at your installation, you might add a division name for a user by issuing the following command:

**Example:**

```
ALTUSER ROBIN CSDATA(DIVISION(SALES))
```

Custom field data in profiles is stored in the CSDATA segment of these profiles. You can list custom field data using the CSDATA keyword of the listing commands.

## Task roadmap for defining and using custom fields

### About this task

Perform the following subtasks to define a custom field and begin using it.

Subtask	Associated instructions (see ... )
Define a custom field and its field attributes.	<a href="#">“Steps for defining a custom field and its attributes” on page 640.</a>
Activate a custom field.	<a href="#">“Steps for activating a custom field” on page 642.</a>
Use and administer a custom field.	<a href="#">“Steps for adding data to a custom field” on page 643.</a>
Optionally, delegate authority to define custom fields.	<a href="#">“Steps for authorizing users to define custom fields” on page 645.</a>
Optionally, delegate authority to add and update data in a custom field.	<a href="#">“Steps for authorizing users to update data in a custom field” on page 646.</a>

## Defining a custom field and its field attributes

Define a custom field and its attributes by issuing the RDEFINE command with the CFDEF operand. This creates a profile in the CFIELD class with a CFDEF segment that contains the definition of your custom field and its attributes. You can modify certain attributes of a custom field using the RALTER command with the CFDEF operand.

Based on the data type you choose for the custom field, you can specify several additional options or accept the defaults for RDEFINE command processing. See detailed command syntax for the RDEFINE and RALTER commands in [z/OS Security Server RACF Command Language Reference](#).

To prevent errors and simplify the task of defining custom fields, the RDEFINE command provides extensive default processing to set appropriate initial values based on the data type of your custom field. You can specify the data type or accept CHAR as the default. You cannot change the data type of a custom field using the RALTER command.

### Guidelines:

- Take advantage of the extensive default processing with the CFDEF operand of the RDEFINE command to ensure that your custom field is defined with appropriate values for its data type. The CFDEF operand of the RALTER command does not provide default processing. When you want to make changes to an existing custom field, see [“Changing attributes of an existing custom field” on page 647.](#)
- Consider using the RACF ISPF panels to add CFIELD class profiles with CFDEF segment values. The ISPF panels display appropriate keyword choices based on the data type of the custom field you are creating.

## Profiles in the CFIELD class

Each profile in the CFIELD class contains a CFDEF segment that defines a custom field and its attributes. Profiles in the CFIELD class are defined and modified using the RDEFINE and RALTER commands. Generic profiles in the CFIELD class are not allowed. Profile names in the CFIELD class must be fully qualified.

In the following examples of profile names, DATASET.CSDATA.MYFIELD defines the MYFIELD custom field for data sets, GENERAL.CSDATA.COMMENT defines the COMMENT custom field for general resources, USER.CSDATA.EMPSEER defines the EMPSEER custom field for user profiles, and GROUP.CSDATA.COMPADDR defines the COMPADDR custom field for group profiles.

**Examples:**

```

DATASET.CSDATA.MYFIELD
GENERAL.CSDATA.COMMENT

USER.CSDATA.EMPSEER
GROUP.CSDATA.COMPADDR

```

**CFIELD profile names**

The format for a profile name in the CFIELD class is as follows:

*profile-type.segment-name.custom-field-name*

The variables of a profile name in the CFIELD class are defined as follows:

***profile-type***

Specify USER, GROUP, GENERAL, or DATASET to indicate whether this custom field is defined for a user profile or a group profile.

***segment-name***

Specify CSDATA.

***custom-field-name***

Specify a name of up to 8 characters for this custom field. The *custom-field-name* value you choose will be used as the keyword for the following commands, based on the type of profile you specify in the CFIELD profile name.

Users can use the *custom-field-name* as a keyword to do the following:

- Add and change data values for this custom field in the CSDATA segment.

**Examples:**

```

ALTUSER user-ID CSDATA(EMPSEER(value))
ALTGROUP group CSDATA(COMPADDR(value))
RALTER class-name profile-name CSDATA(COMMENT(value))
ALTDSD profile-name CSDATA(MYFIELD(value))

```

- Remove the data from this custom field in the CSDATA segment, when prefixed with the *NO* characters.

**Examples:**

```

ALTUSER user-ID CSDATA(NOEMPSEER)
ALTGROUP group CSDATA(NOCOMPADDR)

```

**Syntax rules for custom field names:**

- 1 - 8 characters in length.
- Valid characters include 0 - 9, A - Z, # (X'7B'), \$ (X'5B'), @ (X'7C'), and several special characters. TSO/E syntax requirements apply. For details about TSO/E syntax requirements, see [Syntax requirements for command and subcommand names in z/OS TSO/E Programming Services](#).

**Restriction:** If TSO/E disallows the command keywords associated with your custom field name, the custom field is not usable.

**Guidelines:**

- Avoid defining custom field names that begin with the characters *NO* because they might conflict with another custom field and cause unpredictable results.

For example, if you define two custom fields called *THING* and *NOTHING* for user profiles, when a user issues the *ALTUSER* command with the *NOTHING* keyword, it is unclear whether the user intends to remove *THING* data from the user profile or add *NOTHING* data.

- You can use a custom field whose name is a subset of another custom field name. For example, if you have defined the custom fields *HOME* and *HOMEADDR*, you can add data to the custom

field HOME, if you fully specify the keyword HOME. However, in this example H, HO, and HOM are ambiguous.

## Steps for defining a custom field and its attributes

### Before you begin:

- Before you use the ISPF panels to define custom fields, your system programmer must add the IRRXUT12 program to the list of TSO/E authorized programs in the AUTHPGM NAMES data set in the IKJTSoxx member of parmlib. If this program is not added, you cannot use the ISPF panels to define custom fields.
- At your option, you can delegate the authority to selected users or groups for defining custom fields. To do this, see [“Authorizing users to define custom fields”](#) on page 644.
- The following steps include some command examples that specify all keywords for CFIELD profiles, rather than allowing RDEFINE processing to supply default values based on data type. You need not specify all keywords as shown.

Perform the following steps to define a new custom field.

1. Issue the RDEFINE command to define a new custom field and its attributes.

**Example 1 - Defining a numeric custom field:** Suppose you want to define a numeric field called EMPSER as an employee serial number in a user profile. Employee serial numbers in your company are 6 - 8 numeric digits.

```
RDEFINE CFIELD USER.CSDATA.EMPSER UACC(NONE)
  CFDEF(TYPE(NUM)
    FIRST(NUMERIC) OTHER(NUMERIC)
    MAXLENGTH(8)
    MINVALUE(100000)
    MAXVALUE(99999999)
    HELP('EMPLOYEE SERIAL NUMBER, 6 - 8 DIGITS')
    LISTHEAD('EMPLOYEE SERIAL='))
```

### Note:

- Because the EMPSER field is defined as TYPE (NUM), values for serial numbers must be specified as numbers when they are added to user profiles.
- Because NUM is the data type, FIRST (NUMERIC) and OTHER (NUMERIC) are the only valid options. They are the default values for TYPE (NUM) and need not be specified with the RDEFINE command.
- For information about listing numeric fields, see the note in Step [“2”](#) on page 644 of [“Steps for adding data to a custom field”](#) on page 643.

**Example 2 - Defining a character custom field:** Suppose you want to define a character field called ADDRESS as an employee's home address in a user profile. You want to define a second character field called PHONE as the employee's home telephone number. In addition, you want to validate the format of the telephone number using a System REXX exec named PHONEVAL.

```
RDEFINE CFIELD USER.CSDATA.ADDRESS UACC(NONE)
  CFDEF(TYPE(CHAR)
    MAXLENGTH(100)
    FIRST(ANY) OTHER(ANY)
    HELP('HOME ADDRESS, UP TO 100 CHARACTERS')
    MIXED(YES)
    LISTHEAD('HOME ADDRESS ='))

RDEFINE CFIELD USER.CSDATA.PHONE UACC(NONE)
  CFDEF(TYPE(CHAR)
    MAXLENGTH(20)
    FIRST(ANY) OTHER(ANY)
    HELP('HOME PHONE, UP TO 20 CHARACTERS')
    VALREXX('PHONEVAL')
    MIXED(NO)
    LISTHEAD('HOME PHONE ='))
```

### Notes:



- TYPE (CHAR) is the default data type and needs not be specified with the RDEFINE command.
- Because FIRST (ANY) and OTHER (ANY) are specified, the values for ADDRESS and PHONE can be added as quoted strings.
- The PHONEVAL System REXX exec must be in the System REXX concatenation at the time a value is assigned to the phone custom field.
- Because MIXED (YES) is specified, the ADDRESS value can contain upper and lower case characters.

**Example 3 - Defining a flag custom field:** Suppose you want to define a flag field in a user profile that indicates whether or not an employee is active.

```
RDEFINE CFIELD USER.CSDATA.ACTIVE UACC(NONE)
CFDEF (TYPE (FLAG)
FIRST (NONATABC) OTHER (NONATABC)
MAXLENGTH (3)
HELP ('CURRENTLY ACTIVE?'))
```

**Note:**

- Because the ACTIVE field is defined as TYPE (FLAG), values must be either YES or NO when adding the flags to user profiles.
- Because FLAG is the data type, the following options are the only valid options. They are the default values for TYPE (FLAG) and need not be specified with the RDEFINE command.
  - FIRST (NONATABC)
  - OTHER (NONATABC)
  - MAXLENGTH (3)

**Example 4 - Defining a hexadecimal custom field:** Suppose you want to define an employee code field in a user profile that can be extracted and processed by the payroll program. The employee code is a string of 8 hexadecimal characters.

```
RDEFINE CFIELD USER.CSDATA.CODE UACC(NONE)
CFDEF (TYPE (HEX)
MAXLENGTH (8)
FIRST (NONATNUM) OTHER (NONATNUM)
HELP ('EMPLOYEE CODE, ENTER 8 HEX CHARACTERS')
LISTHEAD ('EMPLOYEE CODE ='))
```

**Note:**

- Because the CODE field is defined as TYPE (HEX), the data must be specified as characters A - F and 0 - 9.
- Because the data type is HEX, FIRST (NONATNUM) and OTHER (NONATNUM) are the only valid options. They are the default values for TYPE (HEX) and need not be specified with the RDEFINE command.
- For the MAXLENGTH of a TYPE (HEX) custom field, specify an even number because hexadecimal data is stored and displayed as an even number of characters.

**Example 5 - Defining a maximum-length character field:** Suppose you want to define a maximum-length character field called COMPADDR to store the company address in a group profile. Because the address value will include blank characters, COMPADDR will be a quoted string. The address value will also contain mixed-case characters. Default values for all other attributes, including TYPE, are provided by RDEFINE default processing.

```
RDEFINE CFIELD GROUP.CSDATA.COMPADDR UACC(NONE)
CFDEF (FIRST (ANY) OTHER (ANY)
MIXED (YES))
```

**Note:**

- Because FIRST (ANY) and OTHER (ANY) are specified, the value for the COMPADDR field can be added as a quoted string.

- Because HELP and LISTHEAD are not specified, they default as shown in the example of the RLIST command output in Step “2” on page 642.

- 
2. Issue the RLIST command to list the new custom field. Review the results of default RDEFINE processing.

**Example:**

```
RLIST CFIELD GROUP.CSDATA.COMPADDR CFDEF NORACF

CLASS      NAME
-----
CFIELD     GROUP.CSDATA.COMPADDR

CFDEF INFORMATION
-----
TYPE = CHAR
MAXLENGTH = 00001100
MAXVALUE = NONE
MINVALUE = NONE
FIRST = ANY
OTHER = ANY
MIXED = YES
HELP = COMPADDR
LISTHEAD = COMPADDR=
```

---

You have now defined a new custom field.

If you encountered errors, see the appropriate messages documentation and check [“Common errors when defining and using custom fields”](#) on page 651.

## Activating a custom field

Each time you define a new custom field, or you change or delete the definition of custom field, you must notify your system programmer to rebuild the RACF dynamic parse table and restart dynamic parse using the IRRDPI00 command. The CFIELD class must be active when the dynamic parse table is rebuilt.

### Steps for activating a custom field

**Before you begin:**

- System programming skills are required to complete some of the following steps. To activate a new or changed custom field, your system programmer must issue the IRRDPI00 command to rebuild the RACF dynamic parse table and restart dynamic parse. No IPL is required. The IRRDPI00 command is described in [Dynamic parse and IRRDPI00](#) in *z/OS Security Server RACF System Programmer's Guide*.
- If your RACF installation is enabled for sysplex communications or uses RRSF, be aware that the IRRDPI00 command is not automatically propagated. You cannot use new or changed custom fields until your system programmer executes IRRDPI00 UPDATE on each target system.

**Guideline:** Before activating custom fields on multiple systems, test your first system by executing [“Steps for adding data to a custom field”](#) on page 643.

Perform the following steps to activate a new or changed custom field.

1. Activate the CFIELD class if not already active.

```
SETROPTS CLASSACT(CFIELD)
```

- 
2. Notify your system programmer to issue the IRRDPI00 command using the command options as follows.

- a. Optionally, execute the IRRDPI00 CHECK command. When the CFIELD class is active, the output of the IRRDPI00 CHECK command indicates errors in your custom field definitions in the CFIELD class.

---

- b. Execute the IRRDPI00 UPDATE command. When the CFIELD class is active, the IRRDPI00 UPDATE command activates new and changed custom fields.

---

- c. Optionally, execute the IRRDPI00 LIST command. When the CFIELD class is active, the IRRDPI00 LIST command lists the custom fields in effect.

**Examples:**

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

- d. If RACF is enabled for sysplex communication, execute IRRDPI00 UPDATE on each system in the sysplex.

---

- e. If you have a RACF remote sharing facility (RRSF) network and you propagate commands that define fields in the CFIELD class, execute IRRDPI00 UPDATE on each target system.

---

3. Confirm with your system programmer that the IRRDPI00 UPDATE command was issued on the required systems. If you proceed to the next task, [“Adding data to a custom field” on page 643](#), and are unable to add data to the new custom field, it might be because IRRDPI00 UPDATE was not executed.

You have now activated your new or changed custom field. You can begin to use the custom field.

## Adding data to a custom field

You can now add data to the CSDATA segment of profiles by using the ISPF panels or by specifying a custom field name as the command operand for the following commands:

- For user profiles, issue the ADDUSER and ALTUSER commands.
- For group profiles, issue the ADDGROUP and ALTGROUP commands.
- For data set profiles, issue the ADDSD and ALTDSD commands.
- For general resource profiles, issue the RDEFINE and RALTER commands.

**Guidelines:**

- Use the ISPF panels to enter data for custom fields, rather than issuing the RACF commands. When you use the ISPF panels, the custom fields are listed and you can simply select them to add, update or delete data for them.
- If you delegate authority for adding data to custom fields (using [“Steps for authorizing users to update data in a custom field” on page 646](#)), provide customized instructions to delegated users about the data requirements for your installation's custom fields.

**Restriction:** The amount of data that can be stored in the CSDATA segment of a user or group profile is limited to 65535 bytes.

## Steps for adding data to a custom field

**Before you begin:**

- Review the CFIELD profile that defines the custom field by issuing the RLIST command. For an example, see Step [“2” on page 642](#) of [“Steps for defining a custom field and its attributes” on page 640](#).

For a list of *all* custom field names (CFIELD profile names) in your RACF database, issue the SEARCH command for the CFIELD class.

**Example:** SEARCH CLASS(CFIELD) NOMASK

- If you delegate authority to view custom field definitions (using Step “1” on page 645 of “Steps for authorizing users to define custom fields” on page 645), authorized users can issue the RLIST command to review CFIELD profiles.

Perform the following steps to add data to a custom field.

1. Add custom field data to the CSDATA segment of a user or group profile.

**Example 1:** To add data to multiple custom fields in the user profile of ANDREW, issue the following:

```
ALTUSER ANDREW CSDATA(EMPSER(256400)
ADDRESS('14 Main Street, Anywhere, IL 01234')
PHONE(555-555-5555)
CODE(FC01B2D8)
ACTIVE(NO))
```

**Example 2:** To add data to the COMPADDR field in the profile of the ABCSUPPLY group, issue the following:

```
ALTGROUP ABCSUPPLY CSDATA(COMPADDR('75 Industrial Way, Someplace, NC 55555'))
```

- 
2. Issue the LISTUSER or LISTGRP command to review the contents of the CSDATA segment for the changed user or group profile.

**Example 1:**

```
LISTUSER ANDREW CSDATA NORACF
USER=ANDREW
CSDATA INFORMATION
-----
ACTIVE= NO
HOME ADDRESS= 14 Main Street, Anywhere, IL 01234
EMPLOYEE CODE= FC01B2D8
EMPLOYEE SERIAL= 0000256400
HOME PHONE= 555-555-5555
```

**Example 2:**

```
LISTGRP ABCSUPPLY CSDATA NORACF
INFORMATION FOR GROUP ABCSUPPLY
CSDATA INFORMATION
-----
COMPADDR= 75 Industrial Way, Someplace, NC 55555
```

**Note:** When listing numeric custom fields, the value is always displayed using 10 integers, with leading zeros if necessary. Therefore, the length of a displayed numeric field might not match the length as defined by the MAXLENGTH attribute for the custom field. For example, the listed value for EMPLOYEE SERIAL (as shown in the LISTUSER example) is correctly listed as 0000256400 while the MAXLENGTH attribute for EMPSER is 8 (as defined in Step “1” on page 640 **Example 1** of “Steps for defining a custom field and its attributes” on page 640).

You have now added data to a custom field of a RACF profile.

If you encountered errors, see the appropriate messages documentation and check “Common errors when defining and using custom fields” on page 651.

## Authorizing users to define custom fields

Optionally, you can delegate the authority for defining custom fields to selected users and groups. Doing so, enables others to view and create profiles in the CFIELD class. It does not allow them to add or

view data in the CSDATA segments of RACF profiles. (To authorize users to add or view data in CSDATA segments, perform the steps in [“Authorizing users to update data in a custom field”](#) on page 645.)

To authorize users to view and define custom fields, define FIELD profiles that enable field-level access checking for the CFDEF segment of CFIELD profiles. The access list and the UACC value of the FIELD profile determine which users can view or define a custom field.

## Steps for authorizing users to define custom fields

### Before you begin:

- In these steps, you will define FIELD profiles to authorize users to access some or all fields in the CFDEF segment of CFIELD profiles. For a complete listing of the profile name qualifiers you can use to control each field, see details about the CFDEF segment in [“Field-level access checking”](#) on page 200.
- You can define generic profiles in the FIELD class if you enable generics in the FIELD class:

```
SETOPTS GENERIC(FIELD)
```

Optionally, perform the following steps to delegate the authority to view and define custom fields.

1. Authorize all users to use the RLIST command to view all fields in the CFDEF segment of CFIELD profiles.

When you authorize UACC(READ) for the appropriate FIELD profiles, users can use the RLIST command to display custom field names and attributes for those fields. This information is useful to users who add CSDATA for those fields.

### Example:

```
SETOPTS GENERIC(FIELD)
RDEFINE FIELD CFIELD.CFDEF.* UACC(READ)

SETOPTS CLASSACT(FIELD) RACLIST(FIELD)
or, if the FIELD class is already in use:
SETOPTS RACLIST(FIELD) REFRESH
```

2. Authorize selected users and groups to use the RDEFINE and RALTER commands to define and modify all fields in the CFDEF segment of CFIELD profiles.

When you authorize UPDATE access to the appropriate FIELD profiles, you delegate authority to define custom fields.

### Example:

```
ALTUSER USERADM CLAUTH(CFIELD)
PERMIT CFIELD.CFDEF.* CLASS(FIELD) ID(USERADM) ACCESS(UPDATE)
SETOPTS RACLIST(FIELD) REFRESH
```

You have now authorized users and groups to update fields in the CFDEF segment of CFIELD profiles. This allows them to view and define custom fields. It does not allow them to add or view data in the CSDATA of user and group profiles. To authorize users to add or view data in CSDATA segments, perform the steps in [“Authorizing users to update data in a custom field”](#) on page 645.

## Authorizing users to update data in a custom field

Optionally, you can delegate the authority to use RACF commands for viewing and adding custom field information in profiles. To authorize selected users and groups, define FIELD profiles that enable field-level access checking for the CSDATA segment of RACF profiles. The access list and the UACC value of the FIELD profile determine which users can view, update, and delete a custom field.

**Rule:** When you define the FIELD profile, use the CFIELD profile name as the FIELD profile name, or define a generic profile.

For a list of custom field names (CFIELD profile names) that are defined at your installation, issue the RLIST or SEARCH command for the CFIELD class.

### Examples:

```
SEARCH CLASS(CFIELD) NOMASK
RLIST CFIELD * CFDEF NORACF
```

## Authorizing users for the ISPF panels to update custom field data

To enable users to use the ISPF panels to update data in custom fields, you must first create FACILITY class profiles that define the IRR.RADMIN.LISTUSER, IRR.RADMIN.LISTGRP, IRR.RADMIN.RLIST, and IRR.RADMIN.LISTDSD resources. The panels that support custom fields invoke the R\_admin callable service on behalf of the ISPF user to extract existing values of custom fields so they can be displayed in the panels. Therefore, users who update data in custom fields must be authorized with READ access to the appropriate IRR.RADMIN resource. Absent this authorization, the user can still use the panels, but they will not be primed with existing field values.

**Tip:** Use the UACC(READ) option rather than authorizing each user or group to the appropriate IRR.RADMIN resource. Otherwise, if you use UACC(NONE), you must individually authorize each user or group to use the enhanced ISPF panels.

## Steps for authorizing users to update data in a custom field

**Before you begin:** To allow users to use the ISPF panels to update data in custom fields, see [“Authorizing users for the ISPF panels to update custom field data” on page 646](#).

Optionally, perform the following steps to authorize selected users and groups to view, add, and update data in a custom field. Steps [“1” on page 646](#) through [“4” on page 647](#) show various methods you can use to authorize users. If you perform any of these steps, you must perform Step [“5” on page 647](#) to activate your changes.

1. Use the &RACUID variable to authorize users to view and update their own user information in a custom field.

**Example:** Suppose you want to authorize all users to update their own home addresses and telephone numbers in the ADDRESS and PHONE fields.

```
RDEFINE FIELD USER.CSDATA.ADDRESS UACC(NONE)
RDEFINE FIELD USER.CSDATA.PHONE UACC(NONE)

PERMIT USER.CSDATA.ADDRESS CLASS(FIELD) ID(&RACUID) ACCESS(UPDATE)
PERMIT USER.CSDATA.PHONE CLASS(FIELD) ID(&RACUID) ACCESS(UPDATE)
```

- 
2. Authorize selected users and groups to add and update data in the custom fields of user profiles.

**Example:** Suppose you want to authorize the HR group to view and update each user's ACTIVE and EMPSER fields.

```
RDEFINE FIELD USER.CSDATA.ACTIVE UACC(NONE)
RDEFINE FIELD USER.CSDATA.EMPSER UACC(NONE)

PERMIT USER.CSDATA.ACTIVE CLASS(FIELD) ID(HR) ACCESS(UPDATE)
PERMIT USER.CSDATA.EMPSER CLASS(FIELD) ID(HR) ACCESS(UPDATE)
```

- 
3. Authorize selected users and groups to view data in the custom fields of user profiles.

**Example:** Suppose you want to authorize the HELPDESK group to view each user's CODE field.

```
RDEFINE FIELD USER.CSDATA.CODE UACC(NONE)
PERMIT USER.CSDATA.CODE CLASS(FIELD) ID(HELPDESK) ACCESS(READ)
```

4. Authorize selected users and groups to update data in the custom fields of group profiles.

**Example:** Suppose you want to authorize the procurement department to update each group's COMPADDR field.

```
RDEFINE FIELD GROUP.CSDATA.COMPADDR UACC(NONE)
PERMIT GROUP.CSDATA.COMPADDR CLASS(FIELD) ID(PROCGRP) ACCESS(UPDATE)
```

5. Activate your authorizations in the FIELD class:

**Example:**

```
SETROPTS CLASSACT(FIELD) RACLIST(FIELD)
or, if the FIELD class is already in use:
SETROPTS RACLIST(FIELD) REFRESH
```

You have now authorized selected users to view, add, and update custom field information for users and groups at your installation.

## Changing attributes of an existing custom field

**Rule:** Avoid using the RALTER command with the NOCFDEF option to change an existing custom field.

When you define a custom field with the RDEFINE CFIELD command, some custom field attributes are assigned default values. You can change most attributes in the definition of a custom field using the RALTER CFIELD command with the CFDEF operand. However, if you use the RALTER command, default attributes are not assigned or changed. Therefore, you might change an attribute to a value that is incompatible with the data type. Certain attributes are interrelated so if you use the RALTER command, make changes carefully.

When you make a change to a custom field definition (whether you use RALTER or you delete it using RDELETE and redefine it using RDEFINE), any CSDATA values that you have already added for the custom field are *not* changed. For example, if you use the HOMEADDR keyword of the ALTUSER command to add a 50-character HOMEADDR value to the profiles of five users, and you subsequently reduce the maximum length of the HOMEADDR custom field to 20 characters, the HOMEADDR values for those five users are not changed. In this case, those five users will have 50-character HOMEADDR values even though the maximum length for the custom field is now defined as 20 characters.

**Guideline:** Consider using the RACF ISPF panels to modify CFDEF segment values in CFIELD class profiles. The ISPF panels will display the current values in a CFDEF segment and allow you to update them using a simple user interface.

After you change a custom field definition, you must activate your change by having your system programmer execute the IRRDP100 UPDATE command to rebuild the dynamic parse tables on all systems that will use the changed custom field.

**Restrictions:** You cannot change certain attributes of a custom field.

- You cannot change the TYPE attribute using the RALTER command. If you need to change the TYPE, see [“When you need to change the data type” on page 648](#) for instructions about redefining a custom field.
- You can update but you cannot remove the MAXLENGTH value. (See [“When you need to change the MAXLENGTH of a numeric field” on page 649](#).)
- You can update but you cannot remove the LISTHEAD value.
- You can update but you cannot remove the HELP value.
- You can update but you cannot remove the FIRST value.
- You can update but you cannot remove the OTHER value.
- You can update but you cannot remove the MIXED value.

## When you need to change the data type

You cannot use the RALTER command to change the data type of a custom field. Instead, you must delete the CFIELD profile using the RDELETE command and then redefine it with the RDEFINE command. If you have already assigned field values to any users or groups, the data type for those custom fields will not change because the TYPE is stored in the RACF database with the custom field data. The new TYPE for the custom field will be reflected when subsequent CSDATA values are assigned using the new custom field with the updated TYPE.

For example, you might have a user custom field called PHONE defined with TYPE(NUM) that you want to change to a TYPE(Char) field. If you already used the PHONE keyword to define phone numbers for three user profiles, those PHONE values are stored in the RACF database as integer values. After you change the definition of the PHONE custom field to TYPE(Char), the three integer values remain, while subsequent phone numbers will be stored in the RACF database as character fields. This might confuse users who list these values because numeric fields and character fields are displayed differently.

### Steps for changing the data type

Perform the following steps to change the TYPE attribute of an existing custom field.

1. Optionally, delete all CSDATA values that were previously defined for the custom field you want to change. This step is not required.

For example, to delete all occurrences of the PHONE field in all user profiles, execute the database unload utility (IRRDBU00) to locate all PHONE keywords in the CSDATA segments of user profiles. Then, issue the following command for each user that has a CSDATA PHONE field.

**Example:**

```
ALTUSER user-ID CSDATA(NOPHONE)
```

(See [“Using the RACF database unload utility \(IRRDBU00\)”](#) on page 371 for details about using IRRDBU00.)

2. Issue the RLIST command to review attributes of the custom field. Record the attributes you want to keep unchanged.

**Example:**

```
RLIST CFIELD USER.CSDATA.PHONE NORACF CFDEF
```

3. Issue the RDELETE command to remove the custom field definition.

**Example:**

```
RDELETE CFIELD USER.CSDATA.PHONE
```

4. Redefine the custom field, specifying your new data type. For instructions, see [“Steps for defining a custom field and its attributes”](#) on page 640.

**Example:**

```
RDEFINE CFIELD USER.CSDATA.PHONE
CFDEF(TYPE(Char) MAXLENGTH(20) FIRST(Any) OTHER(Any))
```

5. Activate the new custom field using the IRRDPI00 UPDATE command to refresh the dynamic parse definitions on each system that will use the custom field. For instructions, see [“Steps for activating a custom field”](#) on page 642.



---

You have now changed the data type of custom field by deleting and redefining the custom field. You have also optionally removed any CSDATA values that were previously defined for the custom field. You can now begin to add data for the new custom field. New CSDATA values that you define for this field will now be stored using the new data type.

## When you need to change the MAXLENGTH of a numeric field

When you change the MAXLENGTH attribute of a custom field with the TYPE(NUM) attribute, you might also need to adjust the maximum value (MAXVALUE) and minimum value (MINVALUE) attributes accordingly. This is because the maximum value and minimum value might have been assigned with default values when you defined the numeric custom field using the RDEFINE CFIELD command.

### Steps for changing the MAXLENGTH of a numeric field

Perform the following steps to change the MAXLENGTH attribute of an existing custom field with the TYPE(NUM) attribute.

1. Issue the RLIST command to review the attributes of the custom field. Record the values for the MAXLENGTH, MAXVALUE, and MINVALUE attributes.

#### Example:

```
RLIST CFIELD USER.CSDATA.SALRYCAP NORACF CFDEF
```

```
CLASS    NAME
-----
CFIELD   USER.CSDATA.SALRYCAP

CFDEF INFORMATION
-----
TYPE= NUM
MAXLENGTH= 00000006
MAXVALUE= 0000999999
MINVALUE= 0000000100
FIRST= NUMERIC
OTHER= NUMERIC
MIXED= NO
HELP= SALARY MAXIMUM, 3-6 DIGITS
LISTHEAD= SALARY CAP =
```

- 
2. Issue the RALTER command to update the values for MAXLENGTH and other related attributes, as needed.

#### Example:

```
RALTER CFIELD USER.CSDATA.SALRYCAP
CFDEF(MAXLENGTH(8) MAXVALUE(99999999) MINVALUE(1000)
HELP('SALARY MAXIMUM, 4-8 DIGITS'))
```

- 
3. Reissue the RLIST command to review attributes of the updated custom field.

#### Example:

```
RLIST CFIELD USER.CSDATA.SALRYCAP NORACF CFDEF
```

```
CLASS    NAME
-----
CFIELD   USER.CSDATA.SALRYCAP

CFDEF INFORMATION
-----
TYPE= NUM
MAXLENGTH= 00000008
MAXVALUE= 0099999999
MINVALUE= 0000001000
FIRST= NUMERIC
```

```
OTHER= NUMERIC  
MIXED= NO  
HELP= SALARY MAXIMUM, 4-8 DIGITS  
LISTHEAD= SALARY CAP =
```

- 
4. Activate the new custom field using the IRRDPI00 UPDATE command to refresh the dynamic parse definitions on each system that will use the custom field. For instructions, see [“Steps for activating a custom field”](#) on page 642.
- 

You have now changed the MAXLENGTH and other related attributes for a numeric field using the RALTER command. You can now begin to add data for the changed custom field.

## Removing a custom field

---

If you no longer need the definition of a custom field, you can delete the CFIELD profile that defines it. However, before you remove it, you should delete any occurrences of that keyword in a CSDATA segment from profiles by using the custom field keyword with the NO prefix. Once you delete the CFIELD profile, you cannot use the custom field keyword with the NO prefix for that custom field. Instead, you must delete the entire CSDATA segment using the NOCSDATA keyword when you want to remove field data.

### Steps for removing a custom field

**Before you begin:** Update or remove applications that use the custom field definition before you remove the definition. If you do not, those applications might fail after you remove the custom field definition.

Perform the following steps to delete the definition of an existing custom field.

1. Delete all CSDATA values that were previously defined for the custom field you want to remove.

For example, to delete all occurrences of the CODE field in GROUP profiles, execute the database unload utility (IRRDBU00) to locate all CODE keywords in the CSDATA segments of group profiles. Then, issue the following command for each group that has a CSDATA CODE field.

**Example:**

```
ALTGROUP group CSDATA(NOCODE)
```

(See [“Using the RACF database unload utility \(IRRDBU00\)”](#) on page 371 for details about using IRRDBU00.)

- 
2. Issue the RDELETE command to remove the custom field definition.

**Example:**

```
RDELETE CFIELD GROUP.CSDATA.CODE
```

- 
3. Activate your custom field deletion by notifying your system programmer to refresh the dynamic parse definitions by issuing the IRRDPI00 UPDATE command on each system that no longer needs the custom field. (For related information about using IRRDPI00, see [“Activating a custom field”](#) on page 642.)
- 

You have now deleted a custom field and removed any CSDATA values that were previously defined in user and group profiles for the custom field.

## Common errors when defining and using custom fields

If you incorrectly define a custom field or you attempt to add unacceptable values to the CSDATA segment of a profile, you might encounter error and warning messages. Some messages might be RACF messages and some might be TSO/E messages resulting from errors detected by dynamic parse.

### Errors defining a custom field

If you attempt to define a CFIELD profile specifying an incorrect profile name, a message is issued. For example, if you specify the third qualifier with more than eight characters as shown, you receive a message similar to the following:

**Example:**

```
RDEFINE CFIELD USER.CSDATA.EMPLADDRS CFDEF UACC(NONE)
IKJ56702I INVALID ENTITY, USER.CSDATA.EMPLADDRS
```

Similarly, if you specify the first qualifier of the profile name as other than USER, GROUP, GENERAL RESOURCE, or DATASET, or specify the second qualifier as other than CSDATA, you receive a message similar to the following:

**Examples:**

```
RDEFINE CFIELD GROUPX.CSDATA.ADDR CFDEF UACC(NONE)
IKJ56702I INVALID ENTITY, GROUPX.CSDATA.ADDR

RDEFINE CFIELD USER.CFDATA.ADDR CFDEF UACC(NONE)
IKJ56702I INVALID ENTITY, USER.CFDATA.ADDR
```

### Errors adding data to a custom field

Errors when adding data to a custom field might occur when you issue one of the following commands to add data to the CSDATA of a profile:

- For user profiles: the ADDUSER and ALTUSER commands.
- For group profiles: the ADDGROUP and ALTGROUP commands.
- For data sets: the ADDSD and ALTDSD commands.
- For general resources: the RDEFINE and RALTER commands.

### Specifying an unacceptable data value

If you attempt to add an unacceptable value for a numeric custom field, a message is issued. For example, when you add an employee serial number greater than allowed by the MAXVALUE for the EMPSER field, you receive a message similar to the following:

**Example:**

```
IKJ56702I INVALID EMPSER, 9999999999
```

Similarly, if you attempt to add an unacceptable value for a character custom field, a message is issued. For example, when you add a numeric value in a character field that requires only alphabetic characters, you receive a message similar to the following:

**Example:**

```
IKJ56702I INVALID SURNAME, PATEL24
```

### Specifying an ambiguous custom field keyword

You can add data for a custom field whose name is a subset of another custom field's name, so long as you fully specify the custom field name on commands. For example, if your installation has defined two

custom fields named HOME and HOMEADDR, you can add data to the custom field HOME, so long as you fully specify HOME.

If you attempt to add data specifying a shortened custom field name that is ambiguous, messages are issued. For example, if your installation has defined two custom fields HOME and HOMEADDR, when you attempt to add data using a shortened custom field name, such as HOM, you receive a message similar to: IRR52119I Keyword name abbreviation HOM is ambiguous.

### Specifying an undefined custom field keyword

If you attempt to add data specifying an undefined custom field keyword, a message similar to the following is issued:

#### Example:

```
IKJ56702I INVALID OPERAND, HOMEADDR
```

The custom field operand might be undefined for any one of several reasons. For example, one of the following errors might have occurred:

- The custom field was not defined using the RDEFINE command.
- The custom field was defined with attributes that are inconsistent with its data type.
- The IRRDPI00 UPDATE command was not issued to refresh the dynamic parse definitions. (Use the IRRDPI00 LIST command to determine if the custom field is active.)
- The CFIELD class was not active when the IRRDPI00 UPDATE command was issued.
- The custom field keyword was incorrectly specified due to a typographical error and does not match the custom field name.

### Specifying a data value that is too long

If you attempt to specify a custom field keyword containing more characters than allowed by MAXLENGTH attribute of the custom field, messages are issued. For example, if a custom field named HOMEADDR is defined with MAXLENGTH(50) and you attempt to add HOMEADDR value containing 51 characters, you receive messages similar to the following:

#### Examples:

```
IRR52218I The value specified for HOMEADDR is not valid.  
          The maximum length allowed is 50.  
IKJ56701I MISSING HOMEADDR+
```

### Failing due to the custom field validation exit

If your installation tailored an IRRVAF01 exit routine to provide additional validation for custom field data, you might encounter a message such as the following:

#### Example:

```
IRR52217I Command failed by field validation exit.
```

(For IRRVAF01 details, see [Custom field validation exit \(IRRVAF01\)](#) in *z/OS Security Server RACF System Programmer's Guide*.)

If your installation implements validity checking using a Rexx exec, the exec can provide error text which RACF will insert into the IRR52223I message. For example, you might encounter a message such as the following:

```
IRR52223I CSDATA validation failed by exit rexex-exec-name. The  
value must start with a capital letter.
```

## RRSF considerations for custom fields

---

If your installation uses automatic direction to synchronize RACF profiles, you should also synchronize profiles in the CFIELD class once you begin to add custom field data to the CSDATA segments of RACF profiles.

When you activate new or changed custom fields, be aware that the IRRDPI00 command is not automatically propagated. Therefore, you cannot use new or changed custom fields until your system programmer executes IRRDPI00 UPDATE on each target system. (See [“Activating a custom field” on page 642.](#))

Do not use command direction to direct a command (such as ADDUSER or ALTGROUP command) that includes a custom field keyword to add or change data in the CSDATA segment of a profile when the custom field is not defined on *both* the local system and the remote system. If the custom field is not defined on either one of the systems, TSO/E dynamic parse will fail the command on the system where the custom field is undefined because the custom field keyword is unknown.



## Chapter 27. Authorizing help desk functions

This topic describes how to authorize several common security tasks to the representatives of your installation's help desk, or customer call center.

Many installations delegate certain security tasks to help desk representatives in an effort to decentralize portions of user administration and reduce cost. The most commonly delegated tasks are meant to address the most frequently reported problems related to user security, such as forgotten passwords and logon failures.

In general, help desk representatives have less authority than security administrators. Security administrators are usually authorized with the SPECIAL or group-SPECIAL attribute, which gives them full authority over user profiles within their scope. By contrast, help desk representatives are not usually authorized with these attributes. Therefore, you must delegate to them the specific security tasks they need.

The following topics describe how to delegate the following authorities to the general users or groups on the staff of your installation's help desk or call center:

- [“Delegating the authority to list user information” on page 655](#)
- [“Delegating the authority to reset passwords and password phrases” on page 660.](#)

### Delegating the authority to list user information

You can authorize a general user or group to use the LISTUSER command to list information in the base segment of user profiles. You can choose to authorize a general user or group to list user information in *any* user profile (other than the profile of a user with the SPECIAL, OPERATIONS, AUDITOR or ROAUDIT attribute), or you can limit the set of user profiles that a general user or group can list. In addition, you can delegate both authorities within your installation.

For details, see the following topics:

- [“Delegating the authority to list user information in any user profile” on page 655](#)
- [“Delegating the authority to list user information in only selected user profiles” on page 656](#)

When you limit the set of user profiles that a general user or group can list, you have the following options:

- [“Delegating the authority to list user information by owner” on page 657](#)
- [“Delegating the authority to list user information by group tree” on page 658](#)
- [“Excluding selected user profiles” on page 659.](#)

### Delegating the authority to list user information in any user profile

To authorize a general user to list user information in any user profile, define a profile to protect the IRR.LISTUSER resource in the FACILITY class. If you do not define this profile, standard LISTUSER authority checking applies when RACF determines whether the command issuer is authorized.

A general user can list the base segment of any user's profile when the command issuer has READ access to the IRR.LISTUSER resource in the FACILITY class. This authority authorizes a general user to list the base segment in the profile of any user including users with the PROTECTED attribute. **Restriction:** This authority does *not* apply when the target of the LISTUSER command has the SPECIAL, AUDITOR, ROAUDIT, or OPERATIONS attribute.

RACF does not log failed access attempts to IRR.LISTUSER. Successful accesses to IRR.LISTUSER are logged at the installation's discretion.

## Steps for delegating the authority to list user information in any user profile

**Before you begin:** Make sure an existing generic profile in the FACILITY class does not inadvertently grant this authority.

Perform the following steps to authorize a general user or group to list user information in any user profile.

1. Define a profile to protect the IRR.LISTUSER resource in the FACILITY class.

**Example:**

```
RDEFINE FACILITY IRR.LISTUSER UACC(NONE)
AUDIT(SUCCESS(READ))
```

- 
2. Authorize the general users or groups.

**Example:**

```
PERMIT IRR.LISTUSER CLASS(FACILITY) ID(HELPDESK USER19) ACCESS(READ)
```

- 
3. Activate the FACILITY class if not already active.

**Example:**

```
SETOPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

---

You have now authorized a general user or group to list the base segment of the user profile for any user, including a protected user, and excluding users with the SPECIAL, OPERATION, AUDITOR, or ROAUDIT attribute.

## Delegating the authority to list user information in only selected user profiles

You can limit the authority of a general user or group to list user information by authorizing the user or group to list only a selected set of user profiles. You can limit the selected set of user profiles in the following ways:

- **Delegating by owner**

You can limit the authority of a general user or group to list user information in user profiles based on the owner of the user profile. To do this, authorize the LISTUSER command issuer with READ authority to the IRR.LU.OWNER.*owner* resource in the FACILITY class.

For details, see [“Delegating the authority to list user information by owner” on page 657.](#)

- **Delegating by group tree**

You can limit the authority of a general user or group to list user information in only user profiles that are within the scope of a selected group tree. To do this, authorize the LISTUSER command issuer with READ authority to the IRR.LU.TREE.*owner* resource in the FACILITY class.

For details, see [“Delegating the authority to list user information by group tree” on page 658.](#)

- **Excluding user profiles**

You can exclude selected user profiles from the scope of IRR.LU.OWNER.*owner* and IRR.LU.TREE.*owner* processing. To do this, protect the IRR.LU.EXCLUDE.*user-ID* resource in the FACILITY class.

For details, see [“Excluding selected user profiles” on page 659.](#)



To authorize a general user or group to list user information in only selected user profiles, define a profile to protect the appropriate IRR.LU.OWNER or IRR.LU.TREE resource in the FACILITY class and grant READ access to authorize users and groups. If you do not define this profile, standard LISTUSER authority checking applies when RACF determines whether the command issuer is authorized.

The IRR.LU.OWNER and IRR.LU.TREE authorities authorize a general user to list the base segment in the profile of any user—based on owner or scope of the group tree—including protected users. **Restriction:** These authorities do not apply when the target of the LISTUSER command has the SPECIAL, AUDITOR, ROAUDIT or OPERATIONS attribute.

RACF does not log failed access attempts to IRR.LU resources. Successful accesses to IRR.LU resources are logged at the installation's discretion.

## Delegating the authority to list user information by owner

You can authorize a general user or group to list user information in user profiles based on the owner of the user profile. To do this, authorize the LISTUSER command issuer with READ authority to the IRR.LU.OWNER.*owner* resource in the FACILITY class. The list-of-groups checking option (SETROPTS GRPLIST) need not be active and has no effect on this authority.

### Steps for delegating the authority to list user information by owner

#### Before you begin:

- Make sure the LISTUSER command issuer does not have READ access to the IRR.LISTUSER resource in the FACILITY class.

Perform the following steps to limit the authority of a general user or group to list user information in selected user profiles based on the owner of the user profiles.

1. Define the following generic profiles in the FACILITY class, if not already defined. Doing so ensures that an existing generic profile does not inadvertently prevent you from successfully limiting this authority.

#### Example:

```
RDEFINE FACILITY IRR.LISTUSER.** UACC(NONE)
RDEFINE FACILITY IRR.LU.** UACC(NONE)
RDEFINE FACILITY IRR.LU.EXCLUDE.** UACC(READ)
```

2. Define a profile to protect the IRR.LU.OWNER.*owner* resource in the FACILITY class, where *owner* is the user ID or group that owns the user profiles.

#### Example:

```
RDEFINE FACILITY IRR.LU.OWNER.GROUP3 UACC(NONE)
AUDIT(SUCCESS(READ))
```

3. Authorize the general users or groups.

#### Example:

```
PERMIT IRR.LU.OWNER.GROUP3 CLASS(FACILITY) ID(HELPDESK USER19) ACCESS(READ)
```

4. Activate the FACILITY class if not already active.

#### Example:

```
SETROPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

You have now authorized a general user or group to list the base segment of user profiles for selected users, including protected users, and excluding users with the SPECIAL, OPERATION, AUDITOR, or ROAUDIT attribute, based on the owner of the user profile.

## Delegating the authority to list user information by group tree

You can authorize a general user or group to list user information in only user profiles that are within the scope of a selected group tree. To do this, authorize the LISTUSER command issuer with READ authority to the IRR.LU.TREE.*owner* resource in the FACILITY class, where *owner* is the group that is at the top of a group tree.

**Rule:** The list-of-groups checking option (SETROPTS GRPLIST) *must* be active.

### Scope of a group tree

The scope of a group tree includes the following user profiles:

- User profiles that are owned by the group.
- User profiles that are owned by a subgroup that is owned by the group, or by a subgroup that is owned by a subgroup that is owned by the group, and so on.

The set of user profiles within scope of a group tree is the same set that applies when you authorize a user with the group-SPECIAL attribute. When you delegate by group tree, the user has authority to only view the base information in those user profiles. By contrast, when you give a user the group-SPECIAL attribute, the user has full authority over the user profiles within the scope of the group. For this reason, delegating by group tree is usually more appropriate for help desk personnel than authorizing them with the group-SPECIAL attribute.

## Steps for delegating the authority to list user information by group tree

### Before you begin:

- Make sure the LISTUSER command issuer does not have READ access to the IRR.LISTUSER resource in the FACILITY class.
- Ensure that list-of-groups-checking (SETROPTS GRPLIST) is enabled.

Perform the following steps to authorize a general user to list user information in selected user profiles based on the scope of a group tree.

1. Define the following generic profiles in the FACILITY class, if not already defined. Doing so ensures that an existing generic profile does not inadvertently prevent you from successfully limiting this authority.

#### Example:

```
RDEFINE FACILITY IRR.LISTUSER.** UACC(NONE)
RDEFINE FACILITY IRR.LU.** UACC(NONE)
RDEFINE FACILITY IRR.LU.EXCLUDE.** UACC(READ)
```

2. Define a profile to protect the IRR.LU.TREE.*owner* resource in the FACILITY class, where *owner* is the group that is at the top of a group tree.

#### Example:

```
RDEFINE FACILITY IRR.LU.TREE.GROUP1 UACC(NONE)
AUDIT(SUCCESS(READ))
```

3. Authorize the general users or groups.

**Example:**

```
PERMIT IRR.LU.TREE.GROUP1 CLASS(FACILITY) ID(HELPDESK USER19) ACCESS(READ)
```

- 
4. Activate the FACILITY class if not already active.

**Example:**

```
SETOPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

---

You have now authorized a general user or group to list the base segment of user profiles for selected users, including protected users, and excluding users with the SPECIAL, OPERATION, AUDITOR, or ROAUDIT attribute, based on the scope of a group tree.

## Excluding selected user profiles

You can exclude selected user profiles from the scope of IRR.LU.OWNER.*owner* and IRR.LU.TREE.*owner* processing so that users authorized by these IRR.LU resources cannot list user information for the excluded user profiles. To exclude selected users, define a profile in the FACILITY class to protect the IRR.LU.EXCLUDE.*excluded-user* resource, where *excluded-user* is the user ID you are excluding.

When you protect the IRR.LU.EXCLUDE.*excluded-user* resource with UACC(NONE) and give no general users or groups access, the user information of the excluded user cannot be listed even when the command issuer has READ access to the appropriate IRR.LU.OWNER.*owner* and IRR.LU.TREE.*owner* resource in the FACILITY class.

In other words, when a general user, who has no access to the IRR.LU.EXCLUDE.*excluded-user* resource, attempts to list the user profile of an excluded user, the LISTUSER command fails.

Users and groups that you authorize with READ access to the IRR.LU.EXCLUDE.*excluded-user* resource are allowed to list the profile of the excluded user when they also have READ access to the appropriate IRR.LU resource.

**Tip:** If you want to exclude a set of users with similar user IDs, use a generic name (such as GRPADM\*) in place of the excluded user ID.

**Restriction:** Users who are authorized by the IRR.LISTUSER resource are *not* limited when you exclude user profiles with the IRR.LU.EXCLUDE.*excluded-user* resource in the FACILITY class. Excluded users are excluded *only* when the general user or group has authority through the IRR.LU.OWNER.*owner* or IRR.LU.TREE.*owner* resource in the FACILITY class.

User profiles with the SPECIAL, AUDITOR, ROAUDIT, or OPERATIONS attribute cannot be listed by users with authority through the IRR.LU resources. Therefore, you need not exclude users with these attributes using the IRR.LU.EXCLUDE.*excluded-user* resource.

## Steps for excluding selected user profiles

Perform the following steps to exclude selected user profiles from the authority of a general user or group that is authorized through the IRR.LU.OWNER.*owner* or IRR.LU.TREE.*owner* resource in the FACILITY class.

1. Define the following generic profiles in the FACILITY class, if not already defined. Doing so ensures that an existing generic profile does not inadvertently prevent you from successfully excluding selected user profiles.

**Example:**

```
RDEFINE FACILITY IRR.LISTUSER.** UACC(NONE)
RDEFINE FACILITY IRR.LU.** UACC(NONE)
RDEFINE FACILITY IRR.LU.EXCLUDE.** UACC(READ)
```

2. Define a profile to protect the IRR.LU.EXCLUDE.*excluded-user* resource in the FACILITY class using UACC(NONE), where *excluded-user* is the user ID you want to exclude.

**Examples:**

```
RDEFINE FACILITY IRR.LU.EXCLUDE.SHANNON UACC(NONE)
  AUDIT (FAILURES(NONE) SUCCESSES(READ))
RDEFINE FACILITY IRR.LU.EXCLUDE.GRPADM* UACC(NONE)
  AUDIT (SUCCESS(READ))
```

3. Optionally, authorize selected users and groups with READ access to the IRR.LU.EXCLUDE.*excluded-user* resource. Perform this step only when certain users or groups who are authorized to an IRR.LU resource need to list the profile of the excluded user.

**Example:**

```
PERMIT IRR.LU.EXCLUDE.SHANNON CLASS(FACILITY) ID(HELPMGR) ACCESS(READ)
```

4. Activate the FACILITY class if not already active.

**Example:**

```
SETOPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

You have now excluded selected user profiles from the authority of a general user or group that is authorized through the IRR.LU.OWNER.*owner* or IRR.LU.TREE.*owner* resource in the FACILITY class.

## Delegating the authority to reset passwords and password phrases

You can authorize a general user or group to use the ALTUSER command to resume user IDs and reset passwords and password phrases. You can choose to authorize a general user or group to do this for *any* user (other than users with the PROTECTED, SPECIAL, OPERATIONS, AUDITOR, or ROAUDIT attribute), or you can limit the set of users. In addition, you can provide both abilities within your installation.

For details, see the following topics:

- [“Delegating the authority to reset the password for any user” on page 662](#)
- [“Delegating the authority to reset passwords for only selected users” on page 663](#)

When you limit the set of users, you have the following options:

- [“Delegating the authority to reset passwords by owner” on page 663](#)
- [“Delegating the authority to reset passwords by group tree” on page 664](#)
- [“Excluding selected users” on page 666.](#)

## Levels of authority

When you can delegate authority to a general user or group for resuming user IDs and resetting passwords and password phrases, define profiles in the FACILITY class to protect one or more of the following resources based on the scope of authority you need to delegate.

### **IRR.PASSWORD.RESET**

Use this resource when the scope of authority includes all users.

### **IRR.PWRESET.OWNER.owner**

Use this resource when the scope of authority is a limited set of selected users based on owner of the user ID.

### **IRR.PWRESET.TREE.owner**

Use this resource when the scope of authority is a limited set of selected users based on scope of a group tree.

### **IRR.PWRESET.EXCLUDE.excluded-user**

Use this resource to exclude a user profile from the scope of IRR.PWRESET.OWNER.owner and IRR.PWRESET.TREE.owner authority.

**Restriction:** You cannot delegate authority through the IRR.PASSWORD.RESET or IRR.PWRESET resources to authorize a general user or group to resume a revoked user or reset the password or password phrase for a user with *any* of the following attributes. Only users with the SPECIAL attribute, or the appropriate group-SPECIAL attribute, have resume and reset authorities for users with these attributes:

- SPECIAL
- OPERATIONS
- AUDITOR
- ROAUDIT
- PROTECTED.

*Table 40. Authorities you can delegate based on the access level to the IRR.PASSWORD.RESET, IRR.PWRESET.OWNER, IRR.PWRESET.TREE, and IRR.PWRESET.EXCLUDE resources*

Access authority to the IRR.PASSWORD.RESET IRR.PWRESET.OWNER IRR.PWRESET.TREE IRR.PWRESET.EXCLUDE resources	Authorities for using the ALTUSER command that you can delegate to a general user or group
READ	<ul style="list-style-type: none"> <li>• Permits use of the PASSWORD operand to change a user's password (and set as expired).</li> </ul> <p><b>Restriction:</b> You cannot use the PASSWORD operand to add a password for a user who does not have one.</p> <ul style="list-style-type: none"> <li>• Permits use of the PHRASE operand to change a user's password phrase (and set as expired).</li> </ul> <p><b>Restriction:</b> You cannot use the PHRASE operand to add a password phrase for a user who does not have one.</p> <ul style="list-style-type: none"> <li>• Permits use of the RESUME operand, without specifying a date, to resume a revoked user.</li> </ul>
UPDATE	<ul style="list-style-type: none"> <li>• Permits all authorities of READ access.</li> <li>• Permits use of the NOEXPIRED operand with the PASSWORD or PHRASE operand. (See <b>Notes</b>.)</li> </ul>

Table 40. Authorities you can delegate based on the access level to the IRR.PASSWORD.RESET, IRR.PWRESET.OWNER, IRR.PWRESET.TREE, and IRR.PWRESET.EXCLUDE resources (continued)

Access authority to the IRR.PASSWORD.RESET IRR.PWRESET.OWNER IRR.PWRESET.TREE IRR.PWRESET.EXCLUDE resources	Authorities for using the ALTUSER command that you can delegate to a general user or group
CONTROL	<ul style="list-style-type: none"><li>Permits all authorities of UPDATE access.</li><li>Permits use of the PASSWORD or PHRASE operand to reset a user's password or password phrase within the system's minimum change interval.</li></ul>

**Note:**

- Neither being the owner of the user profile, nor having the group-SPECIAL attribute, provides sufficient authority to use the NOEXPIRED operand.
- Only users who have the SPECIAL attribute can use the NOEXPIRED operand for users who have the SPECIAL, OPERATIONS, AUDITOR, or ROAUDIT attribute.

**Delegating the authority to reset the password for any user**

To authorize a general user or group to use the ALTUSER command to resume a revoked user or reset a user's password or password phrase (other than for a protected user or a user with the SPECIAL, OPERATIONS, AUDITOR, or ROAUDIT attribute), define a profile to protect the IRR.PASSWORD.RESET resource in the FACILITY class. If you do not define this profile, standard ALTUSER authority checking applies when RACF determines whether the command issuer is authorized.

RACF does not log failed access attempts to IRR.PASSWORD.RESET. Rather, these attempts are logged as ALTUSER command violations. Successful accesses to IRR.PASSWORD.RESET are logged at the installation's discretion.

**Steps for delegating the authority to reset the password for any user**

Perform the following steps to authorize a general user or group to use the ALTUSER command to resume a revoked user or reset a user's password or password phrase.

- Define a profile to protect the IRR.PASSWORD.RESET resource in the FACILITY class.

**Example:**

```
RDEFINE FACILITY IRR.PASSWORD.RESET UACC(NONE)
AUDIT(SUCCESS(READ))
```

- 
- Authorize the general users or groups.

**Example:**

```
PERMIT IRR.PASSWORD.RESET CLASS(FACILITY) ID(HELPDESK USER19) ACCESS(READ)
```

See “Levels of authority” on page 661 for restrictions and details about authority based on the access level to IRR.PASSWORD.RESET.

- 
- Activate the FACILITY class if not already active.

**Example:**

```
SETOPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

-----

You have now authorized a general user or group to use the ALTUSER command to resume the user ID or reset the password or password phrase for any user, excluding protected users and users with the SPECIAL, OPERATION, AUDITOR, or ROAUDIT attribute.

## Delegating the authority to reset passwords for only selected users

You can limit the authority of a general user or group to use the ALTUSER command (to resume user IDs and reset passwords and password phrases) by authorizing the user or group to do this for only a selected set of users. You can limit the selected set of users in the following ways:

- **Delegating by owner**

You can limit the authority of a general user or group to perform resume and reset functions based on the owner of the user profile. To do this, authorize the ALTUSER command issuer with the appropriate authority to the IRR.PWRESET.OWNER.*owner* resource in the FACILITY class.

For details, see [“Delegating the authority to reset passwords by owner” on page 663](#).

- **Delegating by group tree**

You can limit the authority of a general user or group to perform resume and reset functions for only users within the scope of a selected group tree. To do this, authorize the ALTUSER command issuer with the appropriate authority to the IRR.PWRESET.TREE.*owner* resource in the FACILITY class.

For details, see [“Delegating the authority to reset passwords by group tree” on page 664](#).

- **Excluding user profiles**

You can exclude selected users from the scope of IRR.PWRESET.OWNER.*owner* and IRR.PWRESET.TREE.*owner* processing. To do this, protect the IRR.PWRESET.EXCLUDE.*user-ID* resource in the FACILITY class.

For details, see [“Excluding selected users” on page 666](#).

To authorize a general user or group to use the ALTUSER command to perform resume and reset functions for only selected users, define a profile to protect the appropriate IRR.PWRESET.OWNER or IRR.PWRESET.TREE resource in the FACILITY class and authorize users and groups. If you do not define this profile, standard ALTUSER authority checking applies when RACF determines whether the command issuer is authorized.

**Restriction:** The IRR.PWRESET.OWNER and IRR.PWRESET.TREE authorities do not apply when the target of the ALTUSER command is a protected user or has the SPECIAL, AUDITOR, ROAUDIT or OPERATIONS attribute.

RACF does not log failed access attempts to IRR.PWRESET resources. Rather, these attempts are logged as ALTUSER command violations. Successful accesses to IRR.PWRESET resources are logged at the installation's discretion.

## Delegating the authority to reset passwords by owner

You can authorize a general user or group to perform resume and reset functions for users based on the owner of the user profile. To do this, authorize the ALTUSER command issuer with the appropriate authority to the IRR.PWRESET.OWNER.*owner* resource in the FACILITY class. The list-of-groups checking option (SETOPTS GRPLIST) need not be active and has no effect on this authority.

## Steps for delegating the authority to reset passwords by owner

### Before you begin:

- Make sure the ALTUSER command issuer does not have similar access to the IRR.PASSWORD.RESET resource in the FACILITY class.

Perform the following steps to limit the authority of a general user or group to resume user IDs and reset passwords and password phrases based on the owner of the user profiles.

1. Define the following generic profiles in the FACILITY class, if not already defined. Doing so ensures that an existing generic profile does not inadvertently prevent you from successfully limiting this authority.

#### Example:

```
RDEFINE FACILITY IRR.PASSWORD.RESET.** UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.** UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.** UACC(READ)
```

If you use UPDATE or CONTROL access for any IRR.PWRESET profile, as described in [Table 40 on page 661](#), specify the higher level (UPDATE or CONTROL) with the UACC operand for the IRR.PWRESET.EXCLUDE.\*\* profile instead of the UACC(READ) option shown in this example.

2. Define a profile to protect the IRR.PWRESET.OWNER.*owner* resource in the FACILITY class, where *owner* is the user ID or group that owns the user profiles.

#### Example:

```
RDEFINE FACILITY IRR.PWRESET.OWNER.GROUP3 UACC(NONE)
AUDIT(FAILURES(NONE) SUCCESSES(READ))
```

- 
3. Authorize the general users or groups.

#### Example:

```
PERMIT IRR.PWRESET.OWNER.GROUP3 CLASS(FACILITY)
ID(HELPDESK USER19) ACCESS(READ)
```

See “Levels of authority” on [page 661](#) for restrictions and details about authority based on the access level to the IRR.PWRESET.OWNER.*owner* resource in the FACILITY class.

- 
4. Activate the FACILITY class if not already active.

#### Example:

```
SETOPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

---

You have now authorized a general user or group to resume user IDs and reset passwords and password phrases for selected users, excluding protected users, and users with the SPECIAL, OPERATION, AUDITOR, or ROAUDIT attribute, based on the owner of the user profile.

## Delegating the authority to reset passwords by group tree

You can authorize a general user or group to perform resume and reset functions for only users that are within the scope of a selected group tree. To do this, authorize the ALTUSER command issuer with the appropriate authority to the IRR.PWRESET.TREE.*owner* resource in the FACILITY class, where *owner* is the group that is at the top of a group tree.



**Rule:** The list-of-groups checking option (SETROPTS GRPLIST) *must* be active.

## Scope of a group tree

The scope of a group tree includes the following user profiles:

- User profiles that are owned by the group.
- User profiles that are owned by a subgroup that is owned by the group, or by a subgroup that is owned by a subgroup that is owned by the group, and so on.

The set of user profiles within scope of a group tree is the same set that applies when you authorize a user with the group-SPECIAL attribute. When you delegate by group tree, the user has authority only to resume user IDs and reset passwords and password phrases. By contrast, when you give a user the group-SPECIAL attribute, the user has full authority over the users within the scope of the group. For this reason, delegating by group tree is usually more appropriate for help desk personnel than authorizing them with the group-SPECIAL attribute.

## Steps for delegating the authority to reset passwords by group tree

### Before you begin:

- Make sure the ALTUSER command issuer does not have similar access to the IRR.PASSWORD.RESET resource in the FACILITY class.
- Ensure that list-of-groups-checking (SETROPTS GRPLIST) is enabled.

Perform the following steps to limit the authority of a general user or group to resume user IDs and reset passwords and password phrases based on the scope of a group tree.

1. Define the following generic profiles in the FACILITY class, if not already defined. Doing so ensures that an existing generic profile does not inadvertently prevent you from successfully limiting this authority.

#### Example:

```
RDEFINE FACILITY IRR.PASSWORD.RESET.** UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.** UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.** UACC(READ)
```

If you use UPDATE or CONTROL access for any IRR.PWRESET profile, as described in [Table 40 on page 661](#), specify the higher level (UPDATE or CONTROL) with the UACC operand for the IRR.PWRESET.EXCLUDE.\*\* profile instead of the UACC(READ) option shown in this example.

2. Define a profile to protect the IRR.PWRESET.TREE.*owner* resource in the FACILITY class, where *owner* is the group that is at the top of a group tree.

#### Example:

```
RDEFINE FACILITY IRR.PWRESET.TREE.GROUP1 UACC(NONE)
AUDIT(SUCCESS(READ))
```

- 
3. Authorize the general users or groups.

#### Example:

```
PERMIT IRR.PWRESET.TREE.GROUP1 CLASS(FACILITY)
ID(HELPDESK USER19) ACCESS(READ)
```

See “Levels of authority” on [page 661](#) for restrictions and details about authority based on the access level to the IRR.PWRESET.TREE.*owner* resource in the FACILITY class.

- 
4. Activate the FACILITY class if not already active.

**Example:**

```
SETOPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETOPTS RACLIST(FACILITY) REFRESH
```

-----

You have now authorized a general user or group to resume user IDs and reset passwords and password phrases for selected users, excluding protected users, and users with the SPECIAL, OPERATION, AUDITOR, or ROAUDIT attribute, based on the scope of a group tree.

## Excluding selected users

You can exclude selected user profiles from the scope of IRR.PWRESET.OWNER.*owner* and IRR.PWRESET.TREE.*owner* processing so that users authorized by these IRR.PWRESET resources cannot resume user IDs and reset passwords and password phrases for the excluded user profiles. To exclude selected users, define a profile in the FACILITY class to protect the IRR.PWRESET.EXCLUDE.*excluded-user* resource, where *excluded-user* is the user ID you are excluding.

When you protect the IRR.PWRESET.EXCLUDE.*excluded-user* resource with UACC(NONE) and give no general users or groups access, the excluded user's user ID cannot be resumed and the password and password phrase cannot be reset even when the command issuer has READ (or higher) access to the appropriate IRR.PWRESET.OWNER.*owner* and IRR.PWRESET.TREE.*owner* resource in the FACILITY class.

In other words, when a general user, who has no access to the IRR.PWRESET.EXCLUDE.*excluded-user* resource, attempts to resume the user ID or reset the password or password phrase of an excluded user, the ALTUSER command fails.

Users and groups that you authorize with READ access to the IRR.PWRESET.EXCLUDE.*excluded-user* resource are allowed to resume the user ID and reset the password and password phrase of the excluded user when they also have READ access to the appropriate IRR.PWRESET resource.

See “Levels of authority” on page 661 for restrictions and details about authority based on the access level to the IRR.PWRESET.EXCLUDE.*excluded-user* resource in the FACILITY class.

**Tip:** If you want to exclude a set of users with similar user IDs, use a generic name (such as GRPADM\*) in place of the excluded user ID.

**Restriction:** Users who are authorized by the IRR.PASSWORD.RESET resource are *not* limited when you exclude user profiles with the IRR.PWRESET.EXCLUDE.*excluded-user* resource. Excluded users are excluded *only* when the general user or group has authority through the IRR.PWRESET.OWNER.*owner* or IRR.PWRESET.TREE.*owner* resource.

Protected users and users with the SPECIAL, AUDITOR, ROAUDIT or OPERATIONS attribute cannot be resumed, or have their passwords or password phrases reset, by users with authority through the IRR.PWRESET resources. Therefore, you need not exclude users with these attributes using the IRR.PWRESET.EXCLUDE.*excluded-user* resource.

## Steps for excluding selected users

Perform the following steps to exclude selected user profiles from the authority of a general user or group that is authorized through the IRR.PWRESET.OWNER.*owner* or IRR.PWRESET.TREE.*owner* resource in the FACILITY class.

1. Define the following generic profiles in the FACILITY class, if not already defined. Doing so ensures that an existing generic profile does not inadvertently prevent you from successfully excluding selected users.

**Example:**

```
RDEFINE FACILITY IRR.PASSWORD.RESET.** UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.** UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.** UACC(READ)
```

If you use UPDATE or CONTROL access for any IRR.PWRESET profile, as described in [Table 40 on page 661](#), specify the higher level (UPDATE or CONTROL) with the UACC operand for the IRR.PWRESET.EXCLUDE.\*\* profile instead of the UACC(READ) option shown in this example.

2. Define a profile to protect the IRR.PWRESET.EXCLUDE.*excluded-user* resource in the FACILITY class, where *excluded-user* is the user ID you want to exclude.

**Examples:**

```
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.SHANNON UACC(NONE)
  AUDIT(SUCCESS(READ))
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.GRPADM* UACC(NONE)
  AUDIT(SUCCESS(READ))
```

3. Optionally, authorize selected users and groups with READ, UPDATE, or CONTROL access to the IRR.PWRESET.EXCLUDE.*excluded-user* resource, according to [Table 40 on page 661](#). Perform this step only when certain users or groups who are authorized to an IRR.PWRESET resource need to resume the user ID or reset the password or password phrase of the excluded user.

**Examples:**

```
PERMIT IRR.PWRESET.EXCLUDE.SHANNON CLASS(FACILITY) ID(HELPMGR) ACCESS(READ)
PERMIT IRR.PWRESET.EXCLUDE.GRPADM* CLASS(FACILITY) ID(HELPMGR) ACCESS(CONTROL)
```

4. Activate the FACILITY class if not already active.

**Example:**

```
SETROPTS CLASSACT(FACILITY)
```

If the FACILITY class is already active and RACLISTed, refresh the FACILITY class profiles.

```
SETROPTS RACLIST(FACILITY) REFRESH
```

You have now excluded selected user profiles from the authority of a general user or group that is authorized through the IRR.PWRESET.OWNER.*owner* or IRR.PWRESET.TREE.*owner* resource.

## Delegating both by owner and by group tree

You can delegate authority for the IRR.PWRESET and IRR.LU resources by profile owner, by group tree, or by a combination of both based on your installation's user profile structure.

If only a portion of your user profile structure is organized in a group-tree structure, use the IRR.LU.TREE and IRR.PWRESET.TREE resources to delegate help desk authorities to support users in that portion of the user population. For the portions of the user population that are not organized in a group-tree structure, use the IRR.LU.OWNER and IRR.PWRESET.OWNER resources to delegate help desk authorities.

[Figure 63 on page 668](#) illustrates delegating help desk authorities by group tree (GROUP1) and delegating by owner (GROUP3) within the same RACF installation.

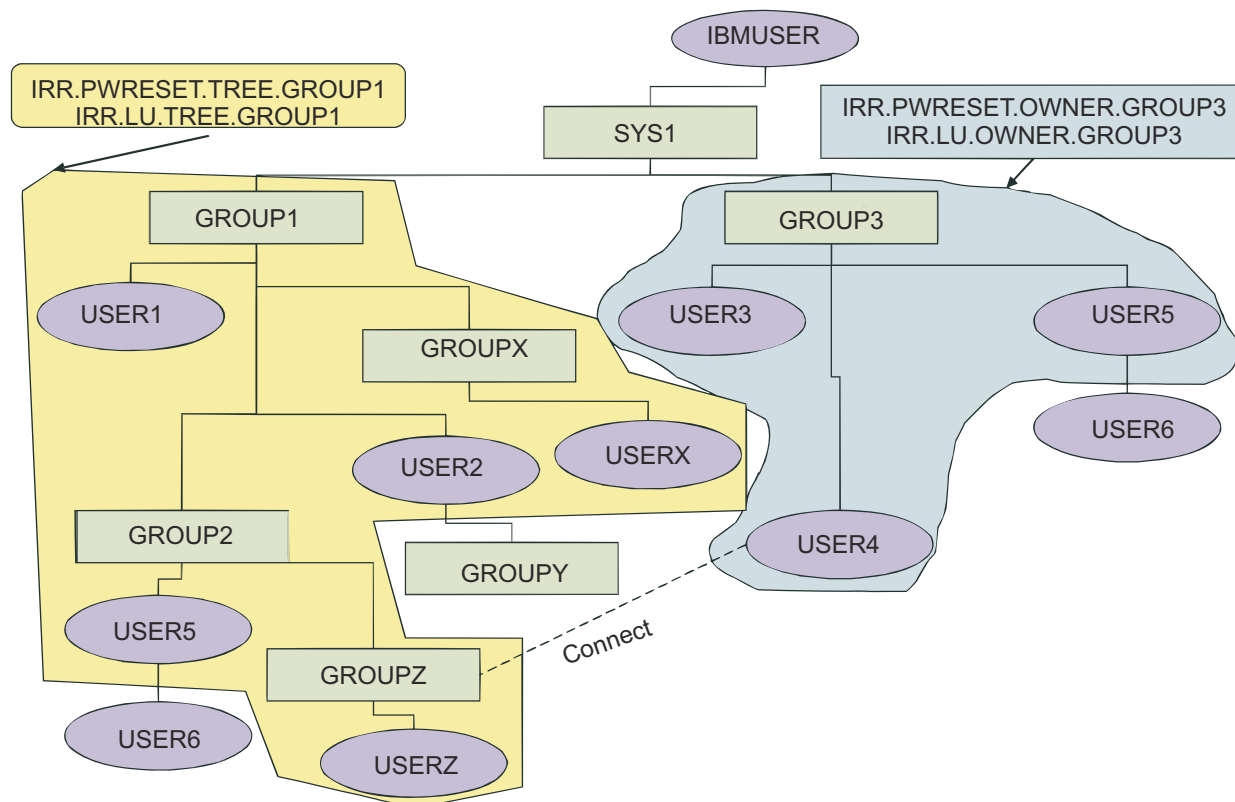


Figure 63. Sample group and user structure for delegating help desk authorities

## Examples of delegating help desk authorities

This topic contains examples of delegating various help desk authorities.

### Delegating help desk authorities by owner

The following examples delegate help desk authorities based on the owner of user profiles.

- User ANDREW needs the abilities to view user profile information, reset passwords and password phrases, resume user IDs, and use the NOEXPIRED operand for users that are owned by TEAMLDR.

#### Examples:

```
RDEFINE FACILITY IRR.LU.OWNER.TEAMLDR UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.LU.OWNER.TEAMLDR CLASS(FACILITY) ACCESS(READ) ID(ANDREW)

RDEFINE FACILITY IRR.PWRESET.OWNER.TEAMLDR UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.PWRESET.OWNER.TEAMLDR CLASS(FACILITY) ACCESS(UPDATE) ID(ANDREW)

SETOPTS CLASSACT(FACILITY)
  or, if the FACILITY class is already active and RACLISTed:
  SETOPTS RACLIST(FACILITY) REFRESH
```

- The users connected to group HLPDESK8 need the abilities to view user profile information, reset passwords and password phrases, and resume user IDs for users that are owned by group AREA8. The following commands also prevent the user profile of the help desk administration user ID (HELPAIDM) from being listed and prevent its password from being reset.

#### Examples:

```
RDEFINE FACILITY IRR.LU.OWNER.AREA8 UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.LU.OWNER.AREA8 CLASS(FACILITY) ACCESS(READ) ID(HLPDESK8)
RDEFINE FACILITY IRR.LU.EXCLUDE.HELPAIDM UACC(NONE)
```

```
RDEFINE FACILITY IRR.PWRESET.OWNER.AREA8 UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.PWRESET.OWNER.AREA8 CLASS(FACILITY) ACCESS(READ) ID(HLPDESK8)
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.HELPAIDM UACC(NONE)

SETROPTS CLASSACT(FACILITY)
  or, if the FACILITY class is already active and RACLISTed:
SETROPTS RACLIST(FACILITY) REFRESH
```

## Delegating help desk authorities by group tree

The following examples delegate help desk authorities based on the scope of a group tree.

- User USERH needs the abilities to reset passwords and password phrases and resume user IDs for users that are in the scope of group GROUP1.

### Examples:

```
RDEFINE FACILITY IRR.PWRESET.TREE.GROUP1 UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.PWRESET.TREE.GROUP1 CLASS(FACILITY) ACCESS(READ) ID(USERH)

SETROPTS CLASSACT(FACILITY)
  or, if the FACILITY class is already active and RACLISTed:
SETROPTS RACLIST(FACILITY) REFRESH
```

- The users connected to group HLPDESK8 need the abilities to reset passwords and password phrases and resume user IDs for users that are in the scope of group GROUP1. The following commands also prevent the password of a group-SPECIAL user called USER1 from being reset.

### Examples:

```
RDEFINE FACILITY IRR.PWRESET.TREE.GROUP1 UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.PWRESET.TREE.GROUP1 CLASS(FACILITY) ACCESS(READ) ID(HLPDESK8)
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.USER1 UACC(NONE)

SETROPTS CLASSACT(FACILITY)
  or, if the FACILITY class is already active and RACLISTed:
SETROPTS RACLIST(FACILITY) REFRESH
```

## Delegating help desk authorities for all users, excluding selected users

In this scenario, an installation currently delegates the ability to reset passwords and list users to a group called HELPDESK by authorizing READ access to the IRR.PASSWORD.RESET profile and the IRR.LISTUSER profile in the FACILITY class. The installation wants to continue to delegate these abilities to the HELPDESK group but now wants to prevent the passwords of two users from being reset. In other words, users who are members of the HELPDESK group need to be authorized to reset passwords and list user profiles for all users except the group-SPECIAL users SHANNON and ANDREW.

The following examples remove the previous authorities from the HELPDESK group and then delegate the authority to reset passwords and list profiles for all users, excluding the two selected users.

1. Remove the HELPDESK group from the access list of the IRR.PASSWORD.RESET and IRR.LISTUSER profiles.

### Examples:

```
PERMIT IRR.PASSWORD.RESET CLASS(FACILITY) ID(HELPDESK) RESET
PERMIT IRR.LISTUSER CLASS(FACILITY) ID(HELPDESK) RESET

SETROPTS CLASSACT(FACILITY)
  or, if the FACILITY class is already active and RACLISTed:
SETROPTS RACLIST(FACILITY) REFRESH
```

2. Delegate help desk authorities to the HELPDESK group using the IRR.LU and IRR.PWRESET profiles, excluding selected users.

### Examples:

```
RDEFINE FACILITY IRR.LU.OWNER.* UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.LU.OWNER.* CLASS(FACILITY) ACCESS(READ) ID(HELPDESK)

RDEFINE FACILITY IRR.PWRESET.OWNER.* UACC(NONE)
  AUDIT(SUCCESS(READ))
PERMIT IRR.PWRESET.OWNER.* CLASS(FACILITY) ACCESS(READ) ID(HELPDESK)

RDEFINE FACILITY IRR.PWRESET.EXCLUDE.ANDREW UACC(NONE)
RDEFINE FACILITY IRR.PWRESET.EXCLUDE.SHANNON UACC(NONE)

RDEFINE FACILITY IRR.LU.EXCLUDE.ANDREW UACC(NONE)
RDEFINE FACILITY IRR.LU.EXCLUDE.SHANNON UACC(NONE)

SETROPTS CLASSACT(FACILITY)
  or, if the FACILITY class is already active and RACLISTed:
SETROPTS RACLIST(FACILITY) REFRESH
```

**Note:** In this scenario, there are no other profiles beginning with IRR.PWRESET.OWNER or IRR.LU.OWNER. If there are, then the HELPDESK group must be given READ access to each such profile.

---

## Chapter 28. Distributed identity filters

This topic provides information about how to map a distributed identity to a RACF user ID and administer distributed identity filters using the RACMAP command.

---

### Overview of distributed identity filters

Today, many transactions that execute on z/OS subsystems originate from the Internet and are initiated by users who authenticate their identities on Web-based, or *distributed*, application servers. When a distributed application server passes a transaction to a z/OS subsystem, the transaction might be associated with the identity of the distributed application user, as defined in a user registry where the transaction originated, or it might be associated with a shared RACF user ID that was assigned by the z/OS subsystem.

To be effective, applications that audit user activities on z/OS subsystems need *both* the RACF user ID associated with a z/OS subsystem transaction *and* the user identity that was presented when the user originally accessed the distributed application server. When you implement distributed identity filters, you map the user's *distributed* identity to a RACF user ID. This allows both user identities to be recorded in the SMF records that are written during the execution of supported transactions, providing more complete auditing for z/OS subsystems.

### What is a distributed identity filter?

A distributed identity filter is a mapping association between a RACF user ID and one or more distributed user identities, as they are known to Web-based application servers and defined in distributed user registries.

A distributed identity filter consists of one or more components of a distributed user's name and the name of the registry where the user is defined. When you define the filter using the RACMAP command, you associate (or map) a distributed user identity with a RACF user ID.

When users attempt to access a z/OS subsystem using a distributed identity, RACF receives distributed user information from authorized applications and uses distributed identity filters to determine the RACF user ID. RACF also uses filter information to support SMF logging of both the RACF user ID and the original identity of the distributed user.

**Note:** Distributed identity filters are *unrelated* to certificate name filters. (See [“Certificate name filtering” on page 571](#)). An installation might choose to implement either distributed identity filters or certificate name filters, both types of filters, or neither.

### Applications that support distributed identity filters

Beginning with z/OS Version 1 Release 11, RACF accepts information about the identities of distributed users from authorized applications that issue the RACROUTE REQUEST=VERIFY request or the `initACEE` callable service (IRRSIA00).

Beginning with CICS Transaction Server (CICS TS) for z/OS Version 4 Release 1, supported z/OS transactions can pass distributed identity filters and realm information to RACF. For information about implementing this support, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).

### Overview of the RACMAP command

Use the RACMAP command to create, delete, and list a distributed identity filter. You cannot modify a distributed identity filter. If changes are required, delete the filter, and define a new one.

The RACMAP command has the following functions:

### MAP

Creates a distributed identity filter

### DELMAP

Deletes a distributed identity filter

### LISTMAP

Lists information about a distributed identity filter

### QUERY

Finds the matching RACF user ID associated with a distributed identity filter

### Example:

```
RACMAP ID(GUSKI) MAP
  USERIDFILTER(NAME('UID=RICH,OU=Web Sales,O=Rich Radio Ham,L=Internet'))
  REGISTRY(NAME('ldaps://us.richradioham.com'))
  WITHLABEL('Rich''s name filter')
```

For complete syntax, authorization requirements, and usage details for the RACMAP command, see [z/OS Security Server RACF Command Language Reference](#).

**Note:** The MAP, DELMAP and LISTMAP functions of the RACMAP command are *unrelated* to the MAP, DELMAP and LISTMAP functions of the RACDCERT command.

## Profiles in the IDIDMAP class

Each distributed identity filter is stored in a general resource profile in the IDIDMAP class. When you use the RACMAP MAP command to add a filter, RACF typically creates a general resource profile in the IDIDMAP class. When you use the RACMAP DELMAP command to delete a filter, RACF typically deletes the IDIDMAP profile that contains the filter.

An IDIDMAP profile might contain multiple filters if you define multiple filters for a particular user name with each filter specifying a different registry. In such cases, RACF does not delete the IDIDMAP profile until you delete the last filter for the user name.

The name of an IDIDMAP profile is the user name portion of the filter, specified as the USERIDFILTER value, stripped of any leading or trailing blank or null characters, normalized, and encoded as UTF-8 data. For information about the encoded UTF-8 data in IDIDMAP profiles, see [“Restrictions for UTF-8 data values”](#) on page 677.

Use the RACMAP command to administer distributed identity filters. Do not use the RDEFINE, RALTER, RDELETE, or RLIST commands to administer profiles in the IDIDMAP class.

The owner of an IDIDMAP profile is the user ID of the RACMAP MAP command issuer. The profile owner has no authority over an IDIDMAP profile or the resources it protects. RACF does not use profile owner information for authorization or any other purpose. You cannot change the profile owner of an IDIDMAP profile.

The profile owner is not listed in the RACMAP LIST output. The profile owner's user ID is listed in the output of the RLIST IDIDMAP \* command, although the RLIST command is unintended for this use. The profile owner's user ID can also be found in the output of the RACF database unload (IRRDBU00) utility.

## RACMAP command updates to user profiles

When you add a filter using the RACMAP MAP command, you map the filter to an existing RACF user ID. In addition to creating (or updating) an IDIDMAP profile, the RACMAP command also updates the profile of the RACF user ID, creating a mapping association between the IDIDMAP profile and the user profile. When you delete the filter, RACMAP deletes (or updates) the IDIDMAP profile containing the filter and updates the user profile of the mapped user ID to remove the mapping association.

If your installation implements RACF remote sharing facility (RRSF), see [“RRSF considerations for distributed identity filters”](#) on page 432 for information about enabling automatic direction of application updates for the RACMAP command to ensure that these profile changes are synchronized across nodes.



## DELUSER processing with distributed identity filters

You cannot delete a RACF user profile that is associated with a distributed identity filter. Before issuing the DELUSER command to delete the user profile of a RACF user ID that is mapped by a filter, you must first remove the mapping association from the user profile by issuing the RACMAP DELMAP command to delete the filter.

If a DELUSER command is issued from a downlevel system to delete a user ID that is mapped by a filter, the user profile might be deleted and result in a *residual* filter. In other words, a distributed identity filter might remain that maps a RACF user ID that no longer exists.

If you share the RACF database with a downlevel system that does not support distributed identity filters, do not issue the DELUSER command from the downlevel system to delete a user ID that is mapped by a filter.

## IRRRID00 considerations for distributed identity filters

To locate residual filters, use the RACF remove ID utility (IRRRID00) to search the RACF database for references to deleted user IDs in IDIDMAP profiles. IRRRID00 does not produce an RDELETE command to delete an IDIDMAP profile that contains a residual filter. Instead, it produces a RACMAP DELMAP command, specifying the user ID and label name, to delete the filter. For usage information about IRRRID00, see [“Using the RACF remove ID \(IRRRID00\) utility” on page 391](#).

For performance information about deleting residual filters, see [“Deleting a distributed identity filter” on page 681](#).

## Details about specifying user and registry names

When you define a distributed identity filter, you must specify both the user and registry name portions of the filter.

This topic includes the following subtopics to assist you in specifying these filter values:

- [“The user name portion of the filter” on page 673](#)
- [“The registry name portion of the filter” on page 674](#)
- [“How RACF matches filter values” on page 674](#)
- [“Adding a default RACMAP filter” on page 676](#)

**Guideline:** Verify your user and registry name values prior to defining the distributed identity filter. The RACMAP command provides no validity checking for the user and registry names you specify.

### The user name portion of the filter

Define the user name portion of the distributed identity filter using the USERDIDFILTER operand. You can specify the user name in any of the following three formats.

1. As a single asterisk (X'5C') to indicate that any user name matches this portion of the filter.
2. As a simple character string, such as a user ID or user name defined in a non-LDAP registry.
3. As a character string that represents an X.500 distinguished name (DN), as defined by RFC2253 from the Internet Engineering Task Force (IETF).

A DN consists of one or more relative distinguished names (RDNs). Each RDN consists of an attribute type and attribute value, separated by an equal sign (=). RDNs are separated by a comma (,).

When you specify the user name as an X.500 DN, you must specify the value in its canonical form, as it is defined within the user registry with the RDNs specified in their correct sequence.

For example, for users of WebSphere Application Server applications, the canonical form of the user name must match the value returned by the WSCredential interface method called `getUniqueSecurityName()`.

**Note:** When you specify the user name as an X.500 DN, the name is normalized before it is stored in the IDIDMAP profile. The normalized form of the DN appears in the output of the RACMAP LISTMAP command. For details about how the DN is normalized, see the description of the USERDIDFILTER operand of the RACMAP MAP function in [z/OS Security Server RACF Command Language Reference](#).

### Examples of user names:

```
USERDIDFILTER(NAME('DENICE'))
USERDIDFILTER(NAME('UID=BobC,CN=Bob Cook,OU=Accounting,O=BobsMart,C=US'))
USERDIDFILTER(NAME('OU=Accounting,O=BobsMart,C=US'))
USERDIDFILTER(NAME('*'))
```

For complete syntax details for defining the USERDIDFILTER value using the RACMAP command, see [z/OS Security Server RACF Command Language Reference](#).

The user name value is stored in the IDIDMAP profile as the profile name in UTF-8 data. For information about the encoded UTF-8 data in IDIDMAP profiles, see [“Restrictions for UTF-8 data values” on page 677](#).

For details about how RACF matches the distributed user's registry and user name with your specified filter values, see [“How RACF matches filter values” on page 674](#).

## The registry name portion of the filter

Define the registry name portion of the distributed identity filter using the REGISTRY operand. You can specify the registry name in either of the following ways.

1. As a single asterisk (X'5C') to indicate that any registry name matches this portion of the filter.

Specify the asterisk when the user is defined with the same name on multiple registries and you want to map all of those identities to the same RACF user ID.

When you want to map any user identity on any registry, see [“Adding a default RACMAP filter” on page 676](#).

2. As the name of a registry, such as an LDAP registry.

For users of WebSphere Application Server applications, the registry name must match the value returned by the WSCredential interface method called `getRealmName()`.

When the user's distributed identity is based on an LDAP registry, specify the registry name as the URL of the LDAP server where the user is defined. The URL is defined with a `listen` option in the `ds.conf` configuration file of the LDAP server, or overridden using the `-l` command-line parameter when the LDAP server is started.

For information about LDAP URLs, see [z/OS IBM Tivoli Directory Server Administration and Use for z/OS](#).

### Examples of registry names:

```
REGISTRY(NAME('ldaps://us.richradioham.com'))
REGISTRY(NAME('ldap://12.34.56.78:389'))
REGISTRY(NAME('Registry01'))
REGISTRY(NAME('*'))
```

For complete syntax details about defining the REGISTRY value using the RACMAP command, see [z/OS Security Server RACF Command Language Reference](#).

The registry name value is stored in the IDIDMAP profile as UTF-8 data. For information about the encoded UTF-8 data in IDIDMAP profiles, see [“Restrictions for UTF-8 data values” on page 677](#).

For details about how RACF matches the distributed user's registry and user name with your specified filter values, see [“How RACF matches filter values” on page 674](#).

## How RACF matches filter values

When a distributed user authenticates on a Web-based application server and takes an action that causes a supported transaction to be sent to the z/OS system, RACF receives the user's distributed user and

registry names as character strings of UTF-8 data. When the IDIDMAP class is active and RACLISTed, RACF uses the UTF-8 data to search IDIDMAP profiles for the distributed identity filter that contains the name values best matching the data. When the best matching filter is found, RACF assigns a RACF user ID.

You can specify user and registry name values in the distributed identity filter to map a RACF user ID using a *one-to-one* match or a *many-to-one* match. In other words, you can define a filter that assigns a RACF user ID to only one distributed user, or you can define a filter that assigns the same RACF user ID to multiple distributed users.

### Using a one-to-one match

A filter that maps a RACF user ID to only *one* distributed user contains a registry name value and contains a user name value that is specified in any of the following ways.

- As a user ID or user name defined in a non-LDAP registry.
  - When you specify the user name in this way, both the distributed user's registry and user name must exactly match the registry and user name values in the filter.

For an example of how RACF searches for a filter that contains a non-LDAP user name, see [“Results for defining a filter for a non-LDAP user name” on page 678](#).

- As an X.500 distinguished name (DN) that includes all RDNs necessary to uniquely identify the distributed user. Depending on the particular LDAP registry, the DN might include the UID or CN components to uniquely identify the user.
  - When you specify the user name in this way, the distributed user's registry must exactly match the registry name value in the filter, *and* the distributed user's name must exactly match all RDNs specified in the user name value in the filter.

For an example of how RACF searches for a filter that contains a full X.500 DN, see [“Results for defining a filter for a full X.500 DN” on page 680](#).

### Using a many-to-one match

A filter that maps the same RACF user ID to *multiple* distributed users contains filter values that are specified in any of the following ways.

- The registry name value is specified as a single asterisk (X'5C') to indicate that any registry name matches the registry portion of the filter.
  - When you specify the registry name in this way and you specify a user name value, the distributed user's name must exactly match the user name value in the user portion of the filter.
  - When you specify each of the user and registry name values as an asterisk, any distributed user's name from any registry matches the filter.

This type of filter is called a *default* RACMAP filter. For more information, see [“Adding a default RACMAP filter” on page 676](#).

- The user name is specified in one of the following ways:
  - As an X.500 distinguished name (DN) that includes selected RDNs that are common to multiple distributed users. Depending on the particular LDAP registry, the specified DN would likely omit the UID or CN components.
    - When you specify the user name in this way and you also specify a registry name value, the distributed user's registry must exactly match the registry name value in the filter, *and* the distributed user's name must match one or more RDNs in the user name value of the filter, in the manner described in [“Details about searching for a filter that matches a user's DN” on page 676](#).
    - When you specify the user name in this way and you specify an asterisk as the registry name, any user's DN that matches one or more RDNs in the user name value of the filter, in the manner described in [“Details about searching for a filter that matches a user's DN” on page 676](#), matches the filter regardless of user registry.

For an example of how RACF searches for a filter that contains selected RDNs, see [“Results for defining a filter using selected RDNs”](#) on page 681.

- As a single asterisk (X'5C') to indicate that any user name matches the user portion of the filter.
  - When you specify the user name as an asterisk and specify a registry name value, only the distributed user's registry must match the registry name value in the filter. Any distributed user from the specified registry matches the filter.
  - When you specify each of the user and registry name values as an asterisk, any distributed user's name from any registry matches the filter.

This type of filter is called a *default* RACMAP filter. For more information, see [“Adding a default RACMAP filter”](#) on page 676.

### ***Details about searching for a filter that matches a user's DN***

When RACF searches for the distributed identity filter that best matches a user's DN, RACF attempts to match the user's registry name and exactly match all RDNs of the user's DN. If a matching filter is found, RACF assigns the user ID specified by the filter.

If no matching filter is found, RACF ignores the most specific or first RDN of the user's DN, for example UID, and performs a second search to locate a less restrictive filter. If a less restrictive filter is found, RACF assigns the user ID specified by the filter.

If no matching filter is found, RACF ignores the first *two* RDNs, for example UID and CN, and performs a third search. If no matching filter is found, RACF iteratively ignores each subsequent RDN, searching for less restrictive filter, until the last RDN is used.

If no matching filter is found, RACF searches for a filter that matches the user's registry name and contains an asterisk as the user name. If a matching filter is found, RACF assigns the user ID specified by the filter.

If no matching filter is found, RACF searches for the default RACMAP filter. If the default filter is defined, RACF assigns the user ID it specifies. If no default filter is found, RACF assigns no user ID.

For an example of how RACF searches for a filter that contains selected RDNs, see [“Results for defining a filter using selected RDNs”](#) on page 681.

### **Adding a default RACMAP filter**

You can map all distributed user names (that are unmapped by more specific filters) by defining a default RACMAP filter. Define a default RACMAP filter by specifying a single asterisk as the user name *and* a single asterisk as the registry name.

#### **Example:**

```
RACMAP ID(WEBUSER) MAP
  USERIDFILTER(NAME('*'))
  REGISTRY(NAME('*'))
  WITHLABEL('Default filter for any WEBUSER')
```

For example, you might define a default RACMAP filter to map a RACF user ID, such as WEBUSER, to any user of z/OS transactions that access information of general or public interest. This is useful when you want to serve selected information, such as product catalogs, to any Web user. In these cases, the user's distributed identity, and the registry that was used for authentication, are unimportant.

**Guideline:** When implementing a default RACMAP filter, map the filter to a RACF user ID that is restricted and protected. For more information, see [“Defining restricted user IDs”](#) on page 75 and [“Defining protected user IDs”](#) on page 74.

#### **Example:**

```
ALTUSER WEBUSER RESTRICTED NOPASSWORD
```

## Restrictions for UTF-8 data values

User and registry names, which you specify using the USERDIDFILTER and REGISTRY operands of the RACMAP command, are not stored in IDIDMAP profiles as EBCDIC data, as most RACF profile data is stored. Instead, they are encoded as UTF-8 data and stored in hexadecimal format so that they are compatible with the typical processing of X.500 distinguished names.

RACF translates these values to the EBCDIC code page specified in the APPLDATA of the IRR.IDIDMAP.PROFILE.CODEPAGE profile in the FACILITY class.

**Note:**

- If the IRR.IDIDMAP.PROFILE.CODEPAGE profile does not exist, then RACF uses code page IBM-1047.
- If the IRR.IDIDMAP.PROFILE.CODEPAGE profile does exist, but contains no APPLDATA or the APPLDATA references a code page other than one of the supported code pages, then RACF uses code page IBM-1047 and issues informational messages.

The supported code pages are:

```
00037 - EBCDIC US 037
00870 - EBCDIC LATIN 2
00875 - EBCDIC GREEK
00924 - EBCDIC US 1047 with the Euro sign
01047 - EBCDIC US 1047
01140 - EBCDIC US 037 with the Euro sign
01153 - EBCDIC LATIN 2 with the Euro sign #2
04971 - EBCDIC GREEK with the Euro sign
```

If during conversion from UTF-8 to EBCDIC, a UTF-8 character is detected for which there is no EBCDIC equivalent, the character appears in hexadecimal format.

**Restrictions:** Because the IDIDMAP profile name (derived from the user name) and the registry name are encoded as UTF-8 data, the following restrictions apply.

- When using the RACMAP command to define user and registry names that contain multibyte characters, if the resulting UTF-8 value exceeds 246 bytes for a user name or 255 bytes for a registry name, the RACMAP MAP command fails with message IRRW213I.
- When using the SEARCH command, you cannot use the FILTER or MASK option to limit your search results based on the names of IDIDMAP profiles.

## Defining a filter for a non-LDAP user name

You can define the user name portion of the filter as a simple character string that specifies a user name or ID that is defined in a non-LDAP registry. To do this, specify the user name, as it is defined in the registry, and specify the name of the user registry or an asterisk.

For details about how RACF matches the distributed user's registry and user name with your specified filter values, see [“How RACF matches filter values” on page 674](#).

## Steps for defining a filter for a non-LDAP user name

**Before you begin:**

- Verify the distributed user name and registry name. (See [“Details about specifying user and registry names” on page 673](#).)
- Verify that the RACF user ID to be mapped by this filter is already defined to RACF. Review its user attributes, groups, and access authorities.

Perform the following steps to define a distributed identity filter that specifies the distributed user name as a simple user name, such as a user ID defined in a non-LDAP registry.

1. Issue the RACMAP command with the MAP function.

### Example:

```
RACMAP ID(DENICE) MAP
  USERIDFILTER(NAME('DENICE'))
  REGISTRY(NAME('Registry01'))
  WITHLABEL('Filter for Denice from Registry01')
```

- 
2. Activate the IDIDMAP class and enable it for RACLIST processing.

### Example:

```
SETOPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
```

If the IDIDMAP class is already active and enabled for RACLIST processing, refresh the IDIDMAP class profiles.

```
SETOPTS RACLIST(IDIDMAP) REFRESH
```

- 
3. Review the new distributed identity filter.

### Example:

```
RACMAP ID(DENICE) LISTMAP
```

### Results:

```
Mapping information for user DENICE:
Label: Filter for Denice from Registry01
Distributed Identity User Name Filter:
  >DENICE<
Registry name:
  >Registry01<
```

---

You have implemented a distributed identity filter that specifies the user name as a user ID on a non-LDAP registry. This filter assigns the RACF user ID DENICE when the distributed identity is the user name DENICE from Registry01.

## Results for defining a filter for a non-LDAP user name

Now, when the user DENICE authenticates her user identity at her Web-based application server and takes an action that causes a transaction to be sent to the z/OS system, RACF is passed the following distributed user and registry names as character strings of UTF-8 data.

- DENICE
- Registry01

When RACF uses these data values to search the IDIDMAP profiles for a matching filter, RACF finds a match to the filter labeled `Filter for Denice from Registry01` and assigns the DENICE user ID. The transaction executes with the authority of the DENICE user ID. Any audit records that are written for this transaction contain both the RACF user ID and the original distributed user and registry names that were passed to RACF when the transaction was sent.

## Defining a filter for an X.500 user identity

You can define the user name portion of the filter as a character string that specifies all or selected parts of an X.500 distinguished name (DN). To do this, specify one or more RDNs as the user name value and specify the name of the LDAP registry, or an asterisk, as the registry name.

For details about how RACF matches the distributed user's registry and user name with your specified filter values, see [“How RACF matches filter values” on page 674](#).

When specifying the user name, you can specify all or selected RDNs of an X.500 DN. The following sets of steps describe both approaches.

- “Steps for defining a filter for a full X.500 DN” on page 679 lists the steps to define the user name portion of the filter as a complete X.500 DN, specifying *all* RDNs for a given user.

The examples in these steps implement a filter that provides a *one-to-one* match, and maps a single user who has a high level of access authority to a RACF user ID.

- “Steps for defining a filter using selected RDNs” on page 680 lists the steps to define filters that specify fewer, *selected* RDNs of the X.500 DN.

The examples in these steps implement filters that provide a *many-to-one* match, and maps multiple users who have a lesser level of access authority to one RACF user ID.

## Steps for defining a filter for a full X.500 DN

### Before you begin:

- Verify the distributed user and registry names. (See “Details about specifying user and registry names” on page 673.)
- Verify that the RACF user ID mapped by this filter is already defined to RACF. Review its user attributes, groups, and access authorities.

Perform the following steps to define a distributed identity filter that specifies the distributed user's name using *all* RDNs of the user's X.500 distinguished name.

1. Issue the RACMAP command with the MAP function.

#### Example:

```
RACMAP ID(RLCOOK) MAP
  USERIDFILTER(NAME('UID=BobC,CN=Bob Cook,OU=Accounting,O=BobsMart,C=US'))
  REGISTRY(NAME('ldaps://us.bobsmarturl.com'))
  WITHLABEL('Accounting boss')
```

- 
2. Activate the IDIDMAP class and enable it for RACLIST processing.

#### Example:

```
SETROPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
```

If the IDIDMAP class is already active and enabled for RACLIST processing, refresh the IDIDMAP class profiles.

```
SETROPTS RACLIST(IDIDMAP) REFRESH
```

- 
3. Review the new distributed identity filter.

#### Example:

```
RACMAP ID(RLCOOK) LISTMAP
```

#### Results:

```
Mapping information for user RLCOOK:
Label: Accounting boss
Distributed Identity User Name Filter:
  >UID=BobC,CN=Bob Cook,OU=Accounting,O=BobsMart,C=US<
Registry name:
  >ldaps://us.bobsmarturl.com<
```



You have implemented a distributed identity filter that specifies the user name as a full X.500 distinguished name. This filter assigns the RACF user ID RLC00K to only one distributed identity that matches *all* RDNs of the user name and matches the LDAP URL specified as the registry name.

If you want to map other users in the same organization who have lesser levels of access authority, you might add additional filters. For examples, see [“Steps for defining a filter using selected RDNs”](#) on page 680.

### Results for defining a filter for a full X.500 DN

Now, when Bob Cook authenticates his LDAP user identity at his Web-based application server and takes an action that causes a transaction to be sent to the z/OS system, RACF is passed the following distributed user and registry names as character strings of UTF-8 data.

- UID=BobC,CN=Bob Cook,OU=Accounting,O=BobsMart,C=US
- ldaps://us.bobsmarturl.com

When RACF uses these data values to search the IDIDMAP profiles for a matching filter, RACF finds an exact match to the filter labeled `Accounting boss` and assigns the RLC00K user ID. The transaction executes with the authority of the RLC00K user ID. Any audit records that are written for this transaction contain both the RACF user ID and the original distributed user and registry names that were passed to RACF when the transaction was sent.

## Steps for defining a filter using selected RDNs

### Before you begin:

- Verify the distributed user and registry names. (See [“Details about specifying user and registry names”](#) on page 673.)
- Verify that the RACF user IDs mapped by these filters are already defined to RACF. Review their user attributes, groups, and access authorities.

Perform the following steps to define a distributed identity filter that specifies *selected* RDNs of an X.500 distinguished name.

1. Issue the RACMAP command with the MAP function.

#### Example:

```
RACMAP ID(ACCTUSER) MAP
  USERIDFILTER(NAME('OU=Accounting,O=BobsMart,C=US'))
  REGISTRY(NAME('ldaps://us.bobsmarturl.com'))
  WITHLABEL('Accounting office workers')
```

**Note:** The user name in this example is based on the DN from the example in [“Steps for defining a filter for a full X.500 DN”](#) on page 679, and omits the most specific RDNs for UID and CN.

- 
2. Activate the IDIDMAP class and enable it for RACLIST processing.

#### Example:

```
SETOPTS CLASSACT(IDIDMAP) RACLIST(IDIDMAP)
```

If the IDIDMAP class is already active and enabled for RACLIST processing, refresh the IDIDMAP class profiles.

```
SETOPTS RACLIST(IDIDMAP) REFRESH
```

- 
3. Review the new distributed identity filter.



**Example:**

```
RACMAP ID(ACCTUSER) LISTMAP
```

**Results:**

```
Mapping information for user ACCTUSER:
Label: Accounting office workers
Distributed Identity User Name Filter:
>OU=Accounting,O=BobsMart,C=US<
Registry name:
>ldaps://us.bobsmarturl.com<
```

You have implemented a distributed identity filter that specifies the user name as a string of selected RDNs of an X.500 distinguished name. This filter assigns the RACF user ID ACCTUSER to any distributed identity that matches the selected components specified in the user name and matches the LDAP URL specified as registry name.

If you want to map other users in the same organization who have lesser levels of access authority, you might add additional filters.

For example, if all DN's in the `us.bobsmarturl.com` registry contain the `O=BobsMart,C=US` RDNs, you might map all users in the `us.bobsmarturl.com` registry by adding another filter as follows:

```
RACMAP ID(BOBSUSER)MAP
USERIDFILTER(NAME('O=BobsMart,C=US'))
REGISTRY(NAME('ldaps://us.bobsmarturl.com'))
WITHLABEL('All BobsMart employees')
```

If general Web users also access the system, you might also consider adding a default RACMAP filter. For information, see [“Adding a default RACMAP filter”](#) on page 676.

**Results for defining a filter using selected RDNs**

Now, when the accounting office worker named Lila Jones authenticates her LDAP user identity at the Web-based application server and takes an action that causes a transaction to be sent to the z/OS system, RACF is passed the following distributed user and registry names as character strings of UTF-8 data.

- `UID=LJones,CN=Lila Jones,OU=Accounting,O=BobsMart,C=US`
- `ldaps://us.bobsmarturl.com`

When RACF uses these data values to search the IDIDMAP profiles for a matching filter, RACF finds no match. When no match is found, RACF removes the most specific portion of the user name, the first RDN of the DN (`UID=LJones`), and performs a second search of the IDIDMAP profiles. When no matching filter is found, RACF removes the first *two* RDNs (`UID=LJones,CN=Lila Jones`) and performs a third search of the IDIDMAP profiles. This time, RACF finds a match to the filter labeled `Accounting office workers` and assigns the ACCTUSER user ID.

The transaction that Lila initiated executes with the authority of the ACCTUSER user ID. Any audit records that are written for this transaction contain both the ACCTUSER user ID and the original distributed user name (including *all* RDNs) and the registry name for user Lila Jones, which were first passed to RACF when the transaction was sent.

## Deleting a distributed identity filter

You cannot modify a distributed identity filter. If you need to make changes to a filter, delete it and define a new one.

**Performance consideration:** When you issue the RACMAP DELMAP command specifying both filter label and a user ID for which no user profile exists, RACF searches all profiles in the IDIDMAP class to locate and delete all matching filters. This search might take an extended period of time.

For information about locating *residual* filters (filters that map to a user ID that no longer exists), see [“IRRRID00 considerations for distributed identity filters” on page 673](#).

### Steps for deleting a distributed identity filter

Perform the following steps to delete a distributed identity filter.

1. Issue the RACMAP command with the DELMAP function.

**Example:**

```
RACMAP ID(DENICE) DELMAP(LABEL('Filter for Denice from Registry01'))
```

---

2. Refresh the IDIDMAP class profiles.

```
SETROPTS RACLIST(IDIDMAP) REFRESH
```

---

You have now deleted a distributed identity filter.

## Appendix A. Supplied RACF resource classes

This appendix describes the general resource classes you can find in the supplied class descriptor table (CDT) and contains the following section:

- [“Supplied resource classes for z/OS systems” on page 683](#)

See [z/OS Security Server RACF Macros and Interfaces](#) to find the details (such as POSIT values) associated with the supplied CDT entry for each class.

### Supplied resource classes for z/OS systems

Table 41 on page 683 lists the supplied CDT classes that can be used on z/OS systems. Several classes are listed in categories based on their usage. See restrictions at the end of the table.

Table 41. Resource classes for z/OS systems

Class name	Description
<b>RACF, MVS, and miscellaneous classes</b>	
ALCSAUTH	Supports the Airline Control System/MVS (ALCS/MVS) product.
ACEECHK	Configuration of RACF ACEE Privilege Escalation Detection.
APPCLU	Verifying the identity of partner logical units during VTAM session establishment.
APPCPORT	Controlling which user IDs can access the system from a given LU (APPC port of entry). Also, conditional access to resources for users entering the system from a given LU.
APPCSERV	Controlling whether a program that is being run by a user can act as a server for a specific APPC transaction program (TP).
APPCSI	Controlling access to APPC side information files.
APPCTP	Controlling the use of APPC transaction programs.
APPL	Controlling access to applications.
CACHECLS	Contains profiles that are used for saving and restoring cache contents from the RACF database.
CBIND	Controlling the client's ability to bind to the server.
CDT	Contains profiles for installation-defined classes for the dynamic CDT. <a href="#">“3” on page 692</a>
CFIELD	Contains profiles that define the installation's custom fields. <a href="#">“3” on page 692</a>
CONSOLE	Controlling access to MCS consoles. Also, conditional access to other resources for commands that originate from an MCS console.
DASDVOL	DASD volumes.
DBNFORM	Reserved for future IBM use.
DEVICES	Used by MVS allocation to control who can allocate devices such as: <ul style="list-style-type: none"> <li>• Unit record devices (printers and punches) (allocated only by PSF, JES2, or JES3)</li> <li>• Graphics devices (allocated only by VTAM)</li> <li>• Teleprocessing (TP) or communications devices (allocated only by VTAM)</li> </ul>
DIGTCERT	Contains digital certificates and information that is related to them.

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
DIGTCRIT	Specifies additional criteria for certificate name filters.
DIGTNMAP	Mapping class for certificate name filters.
DIGTRING	Contains a profile for each key ring and provides information about the digital certificates that are part of each key ring.
DIRAUTH	Setting logging options for RACROUTE REQUEST=DIRAUTH requests. Also, if the DIRAUTH class is active, security label authorization checking is done when a user receives a message that is sent through the TPUT macro or the TSO SEND, or LISTBC commands. <a href="#">“5” on page 692</a>
DLFCLASS	The data lookaside facility.
FACILITY	<p>Miscellaneous uses. Profiles are defined in this class so resource managers (typically elements of z/OS) can check a user's access to the profiles when the user takes some action. Examples are the profiles that are used to control execution of RACDCERT command functions and the profiles that are used to control privileges in the z/OS UNIX environment.</p> <p>RACF does not document all of the resources used in the FACILITY class by other products. For information on the FACILITY class resources used by a specific product (other than RACF itself), see that product's documentation.</p>
FIELD	Fields in RACF profiles (field-level access checking).
GDASDVOL	Resource group class for DASDVOL class. <a href="#">“1” on page 692</a>
GLOBAL	Global access checking table entry. <a href="#">“1” on page 692</a>
GMBR	Member class for the GLOBAL class. <a href="#">“4” on page 692</a>
GSDSF	Resource group class for SDSF class. <a href="#">“1” on page 692</a>
GTERMINL	Resource group class for TERMINAL class. <a href="#">“1” on page 692</a>
GXFACILI	Grouping class for XFACILIT resources.
HBRADMIN	Controls whether server security and security for specific server resources are enabled or disabled.
HBRCONN	Specifies the user IDs that are authorized to connect to the zRule Execution Server for z/OS and execute rule sets. This class is ignored if server security is disabled.
HBRCMD	Specifies the user IDs that are authorized to issue zRule Execution Server for z/OS commands such as <b>START</b> , <b>STOP</b> , <b>PAUSE</b> , or <b>RESUME</b> from the z/OS console (or equivalent). This class is ignored if server security is disabled.
IBMOPC	Controlling access to OPC/ESA subsystems.
IDIDMAP	Contains distributed identity filters that are created with the RACMAP command.
IZP	Controls resources related to the IBM Unified Management Server.
JESINPUT	Conditional access support for commands or jobs that are entered into the system through a JES input device.
JESJOBS	Controlling the submission and cancellation of jobs by job name.
JESSPOOL	Controlling access to job data sets on the JES spool (that is, SYSIN and SYSOUT data sets).

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
KEYSMSTR	Contains profiles that hold keys to encrypt data that is stored in the RACF database, such as LDAP BIND passwords, DCE passwords, and Distributed File Service (DFS) Server Message Block (SMB) passwords.
LDAP	Controls authorization roles for LDAP administration.
LDAPBIND	Contains the LDAP server URL, bind distinguished name, and bind password.
LOGSTRM	Controls system logger resources, such as log streams and the coupling facility structures associated with log streams.
NODES	Controlling the following on MVS systems: <ul style="list-style-type: none"> <li>• Whether jobs are allowed to enter the system from other nodes</li> <li>• Whether jobs that enter the system from other nodes have to pass user identification and password verification checks</li> </ul>
NODMBR	Member class for the NODES class. <a href="#">“4” on page 692</a>
OPERCMD5	Controlling who can issue operator commands (for example, JES and MVS, and operator commands). <a href="#">“2” on page 692</a>
OPTAUDIT	Contains profiles which control RACF logging behavior.
PKISERV	Controls access to R_PKIServ administration functions.
PMBR	Member class for the PROGRAM class. <a href="#">“4” on page 692</a>
PROGRAM	Protects executable programs. <a href="#">“1” on page 692</a>
PROPCNTL	Controlling if user ID propagation can occur, and if so, for which user IDs (such as the CICS or IMS main task user ID), user ID propagation is <i>not</i> to occur.
PSFMPL	Used by PSF to perform security functions for printing, such as separator page labeling, data page labeling, and enforcement of the user printable area.
PTKTDATA	PassTicket key class enables the security administrator to associate a RACF secured signon secret key with a particular mainframe application that uses RACF for user authentication. Examples of such applications are IMS, CICS, TSO, z/VM, APPC, and MVS batch.
RACFEVNT	Contains profiles that control the following events: <ul style="list-style-type: none"> <li>• LDAP change log notification for changes to certain RACF profiles</li> <li>• New password and password phrase enveloping for a given user.</li> </ul>
RACFHC	Used by IBM Health Checker for z/OS. Contains profiles that list the resources to check for each installation-defined health check. <a href="#">“1” on page 692</a>
RACFVARS	RACF variables. In this class, profile names, which start with & (ampersand), act as RACF variables that can be specified in profile names in other RACF general resource classes.
RACGLIST	Class of profiles that hold the results of RACROUTE REQUEST=LIST,GLOBAL=YES or a SETROPTS RACLIST operation.
RACHCMBR	Used by IBM Health Checker for z/OS. Member class for the RACFHC class. <a href="#">“1” on page 692</a>
RDATA LIB	Used to control use of the R_data lib callable service (IRRSDL00 or IRRSDL64).
RRSFDATA	Used to control RACF remote sharing facility (RRSF) functions.

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
RVARSMBR	Member class for the RACFVARS class. <a href="#">“4” on page 692</a>
SCDMBR	Member class for the SECADATA class. <a href="#">“4” on page 692</a>
SDSF	Controls the use of authorized commands in the System Display and Search Facility (SDSF). See also GSDSF class.
SECADATA	Security classification of users and data (security levels and security categories). <a href="#">“1” on page 692</a>
SECLABEL	If security labels are used, and, if so, their definitions. <a href="#">“2” on page 692</a>
SECLMBR	Member class for the SECLABEL class. <a href="#">“4” on page 692</a>
SERVAUTH	Contains profiles used by servers to check a client's authorization to use the server or to use resources that are managed by the server. Also, can be used to provide conditional access to resources for users entering the system from a given server.
SERVER	Controlling the server's ability to register with the daemon.
SMESSAGE	Controlling to which users a user can send messages (TSO only).
SOMDOBJs	Controlling the client's ability to invoke the method in the class.
STARTED	Used in preference to the started procedures table to assign an identity during the processing of an MVS START command.
SURROGAT	If surrogate submission is allowed, and if allowed, which user IDs can act as surrogates.
SYSAUTO	IBM Automation Control for z/OS resources
SYSMVIEW	Controlling access by the SystemView for MVS Launch Window to SystemView for MVS applications.
TAPEVOL	Tape volumes.
TEMPDSN	Controlling who can access residual temporary data sets. <a href="#">“5” on page 692</a>
TERMINAL	Terminals (TSO). See also GTERMINL class.
VTAMAPPL	Controlling who can open ACBs from non-APF authorized programs.
WBEM	Controls access to the Common Information Model (CIM) functions.
WRITER	Controlling the use of JES writers.
XFACILIT	Miscellaneous uses. Profile names in this class can be longer than 39 characters in length. Profiles are defined in this class so that resource managers (typically elements of z/OS) can check a user's access to the resources when the users take some action.
ZOWE	Controls resources related to the Zowe™ project.

**CICS classes**

ACICSPCT	CICS program control table. <a href="#">“2” on page 692</a>
BCICSPCT	Resource group class for the ACICSPCT class. <a href="#">“1” on page 692</a>
CCICSCMD	Used to verify that a user is permitted to use CICS system programmer commands such as INQUIRE, SET, PERFORM, and COLLECT. <a href="#">“1” on page 692</a>

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
CPSMOBJ	Used by CICSplex® System Manager, which provides a central point of control when running multiple CICS systems, to determine operational controls within a CICS complex.
CPSMXMP	Used by CICSplex System Manager to identify exemptions from security controls within a CICS complex.
DCICSDCT	CICS destination control table. <a href="#">“2” on page 692</a>
ECICSDCT	Resource group class for the DCICSDCT class. <a href="#">“1” on page 692</a>
FCICSFCT	CICS file control table. <a href="#">“2” on page 692</a>
GCICSTRN	Resource group class for TCICSTRN class. <a href="#">“2” on page 692</a>
GCPSMOBJ	Resource grouping class for CPSMOBJ.
HCICSFCT	Resource group class for the FCICSFCT class. <a href="#">“1” on page 692</a>
JCICSJCT	CICS journal control table. <a href="#">“2” on page 692</a>
KCICSJCT	Resource group class for the JCICSJCT class. <a href="#">“1” on page 692</a>
MCICSPPT	CICS processing program table. <a href="#">“2” on page 692</a>
NCICSPPT	Resource group class for the MCICSPPT class. <a href="#">“1” on page 692</a>
PCICSPSB	CICS program specification blocks (PSBs).
QCICSPSB	Resource group class for the PCICSPSB class. <a href="#">“1” on page 692</a>
RCICSRES	CICS document templates.
SCICSTST	CICS temporary storage table. <a href="#">“2” on page 692</a>
TCICSTRN	CICS transactions.
UCICSTST	Resource group class for SCICSTST class. <a href="#">“1” on page 692</a>
VCICSCMD	Resource group class for the CCICSCMD class. <a href="#">“1” on page 692</a>
WCICSRES	Resource group class for the RCICSRES class.
<b>Db2z/OS classes</b>	
DSNADM	Db2z/OS administrative authority class.
DSNR	Controls access to Db2z/OS subsystems.
DSNRAUTH	Controls access to Db2z/OS features.
GDSNBP	Grouping class for Db2z/OS buffer pool privileges.
GDSNCL	Grouping class for Db2z/OS collection privileges.
GDSNDB	Grouping class for Db2z/OS database privileges.
GDSNGV	Grouping class for Db2z/OS global variables.
GDSNJR	Grouping class for Java archive files (JARs).
GDSNPK	Grouping class for Db2z/OS package privileges.
GDSNPN	Grouping class for Db2z/OS plan privileges.
GDSNSC	Grouping class for Db2z/OS schemas privileges.

Table 41. Resource classes for z/OS systems (continued)

<b>Class name</b>	<b>Description</b>
GDSNSG	Grouping class for Db2z/OS storage group privileges.
GDSNSM	Grouping class for Db2z/OS system privileges.
GDSNSP	Grouping class for Db2z/OS stored procedure privileges.
GDSNSQ	Grouping class for Db2z/OS sequences.
GDSNTB	Grouping class for Db2z/OS table, index, or view privileges.
GDSNTS	Grouping class for Db2z/OS tablespace privileges.
GDSNUF	Grouping class for Db2z/OS user-defined function privileges.
GDSNUT	Grouping class for Db2z/OS user-defined distinct type privileges.
MDSNBP	Member class for Db2z/OS buffer pool privileges.
MDSNCL	Member class for Db2z/OS collection privileges.
MDSNDB	Member class for Db2z/OS database privileges.
MDSNGV	Member class for Db2z/OS global variables.
MDSNJR	Member class for Java archive files (JARs).
MDSNPK	Member class for Db2z/OS package privileges.
MDSNPN	Member class for plan privileges.
MDSNSC	Member class for Db2z/OS schema privileges.
MDSNSG	Member class for Db2z/OS storage group privileges.
MDSNSM	Member class for Db2z/OS system privileges.
MDSNSP	Member class for Db2z/OS stored procedure privileges.
MDSNSQ	Member class for Db2z/OS sequences.
MDSNTB	Member class for Db2z/OS table, index, or view privileges.
MDSNTS	Member class for Db2z/OS tablespace privileges.
MDSNUF	Member class for Db2z/OS user-defined function privileges.
MDSNUT	Member class for Db2z/OS user-defined distinct type privileges.
<b>DCE class</b>	
DCEUUIDS	Used to define the mapping between a user's RACF user ID and the corresponding DCE principal UUID. Also, used to enable encrypted password support for Distributed File Service (DFS) Server Message Block (SMB) users.
<b>Enterprise Identity Mapping (EIM) class</b>	
RAUDITX	Controls auditing for Enterprise Identity Mapping (EIM).
<b>Enterprise Java Beans classes</b>	
EJBROLE	Member class for Enterprise Java Beans authorization roles.
GEJBROLE	Grouping class for Enterprise Java Beans authorization roles.
JAVA	Contains profiles that are used by Java for z/OS applications to perform authorization checking for Java for z/OS resources.



Table 41. Resource classes for z/OS systems (continued)

Class name	Description
<b>IMS classes</b>	
AIMS	Application group names (AGN).
CIMS	Command.
DIMS	Grouping class for command.
FIMS	Field (in data segment).
GIMS	Grouping class for transaction.
HIMS	Grouping class for field.
IIMS	Program specification block (PSB).
JIMS	Grouping class for program specification block (PSB).
LIMS	Logical terminal (LTERM).
MIMS	Grouping class for logical terminal (LTERM).
OIMS	Other.
PIMS	Database.
QIMS	Grouping class for database.
RIMS	Open Transaction Manager Access (OTMA) transaction pipe (TPIPE).
SIMS	Segment (in database).
TIMS	Transaction (trancode).
UIMS	Grouping class for segment.
WIMS	Grouping class for other.
<b>Integrated Cryptographic Service Facility (ICSF) classes</b>	
CRYPTOZ	Controls access to PKCS #11 tokens.
CSFKEYS	Controls access to ICSF cryptographic keys.
CSFSERV	Controls access to ICSF cryptographic services.
GCSFKEYS	Resource group class for the CSFKEYS class. <a href="#">“1” on page 692</a>
GXCSFKEY	Resource group class for the XCSFKEY class. <a href="#">“1” on page 692</a>
XCSFKEY	Controls the exportation of ICSF cryptographic keys.
<b>Infoprint Server class</b>	
PRINTSRV	Controls access to printer definitions for Infoprint Server.
<b>Information/Management (Tivoli Service Desk) classes</b>	
GINFOMAN	Grouping class for Information/Management (Tivoli Service Desk) resources.
INFOMAN	Member class for Information/Management (Tivoli Service Desk) resources.
<b>LFS/ESA classes</b>	
LFSCCLASS	Controls access to file services provided by LFS/ESA.
<b>License Manager class</b>	

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
ILMADMIN	Controls access to the administrative functions of IBM License Manager.
<b>Lotus Notes for z/OS and Novell Directory Services for OS/390® classes</b>	
NDSLINK	Mapping class for Novell Directory Services for OS/390 user identities.
NOTELINK	Mapping class for Lotus Notes for z/OS user identities.
<b>MFA class</b>	
MFADEF	Contains profiles that define MFA factors. This class can also be used to define MFA application bypass profiles.
<b>IBM MQ</b>	
GMQADMIN	Grouping class for IBM MQ administrative options. <a href="#">“1” on page 692</a>
GMQCHAN	Reserved for IBM MQ.
GMQNLIST	Grouping class for IBM MQ namelists. <a href="#">“1” on page 692</a>
GMQPROC	Grouping class for IBM MQ processes. <a href="#">“1” on page 692</a>
GMQQUEUE	Grouping class for IBM MQ queues. <a href="#">“1” on page 692</a>
MQADMIN	Protects IBM MQ administrative options.
MQCHAN	Reserved for IBM MQ
MQCMDSD	Protects IBM MQ commands.
MQCONN	Protects IBM MQ connections.
MQNLIST	Protects IBM MQ namelists.
MQPROC	Protects IBM MQ processes.
MQQUEUE	Protects IBM MQ queues.
<b>NetView classes</b>	
NETCMDSD	Controlling which NetView commands the NetView operator can issue.
NETSPAN	Controlling which NetView commands the NetView operator can issue against the resources in this span.
NVASAPDT	NetView/Access Services.
PTKTVAL	Used by NetView/Access Services Secured Single Signon to store information needed when generating a PassTicket.
RMTOPS	NetView Remote Operations.
RODMMGR	NetView Resource Object Data Manager (RODM).
<b>z/OS Network Authentication Service classes</b>	
KERBLINK	Contains profiles that map local and foreign principals to RACF user IDs. Also controls which users are authorized to use the SKRBKDC started procedure to decrypt service tickets for a given principal. <a href="#">“3” on page 692</a>
REALM	Used to define the local and foreign realms. <a href="#">“3” on page 692</a>
<b>SMS (DFSMSdfp) classes</b>	
MGMTCLAS	SMS management classes.

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
STORCLAS	SMS storage classes.
SUBSYSNM	Authorizes a subsystem (such as a particular instance of CICS) to open a VSAM ACB and use VSAM record level sharing (RLS) functions.
<b>Tivoli classes</b>	
ROLE	Specifies the complete list of resources and associated access levels that are required to perform the particular job function this role represents and defines which RACF groups are associated with this role.
TMEADMIN	Maps the user IDs of Tivoli administrators to RACF user IDs.
<b>TSO classes</b>	
ACCTNUM	TSO account numbers.
PERFGRP	TSO performance groups.
TSOAUTH	TSO user authorities such as OPER and MOUNT.
TSOPROC	TSO logon procedures.
<b>IBM MQ classes</b>	
GMXADMIN	Grouping class for IBM MQ administrative options.
GMXNLIST	Grouping class for IBM MQ namelists.
GMXPROC	Grouping class for IBM MQ processes.
GMXQUEUE	Grouping class for IBM MQ queues.
GMXTOPIC	Grouping class for IBM MQ topics.
MXADMIN	Protects IBM MQ administrative options.
MXNLIST	Protects IBM MQ namelists.
MXPROC	Protects IBM MQ processes.
MXQUEUE	Protects IBM MQ queues.
MXTOPIC	Protects IBM MQ topics.
<b>z/OSMF classes</b>	
ZMFAPLA	Member class for z/OSMF authorization roles.
GZMFAPLA	Grouping class for z/OSMF authorization roles.
ZMFCLLOUD	Protects z/OS cloud resources.
<b>z/OS UNIX classes</b>	
DIRACC	Controls auditing (using SETROPTS LOGOPTIONS) for access checks for read/write access to z/OS UNIX directories. This class need not be active to control auditing. <a href="#">“5” on page 692</a>
DIRSRCH	Controls auditing (using SETROPTS LOGOPTIONS) of z/OS UNIX directory searches. This class need not be active to control auditing. <a href="#">“5” on page 692</a>
FSACCESS	Controls access to z/OS UNIX file systems.
FSEXEC	Controls execute access to z/OS UNIX file systems.

Table 41. Resource classes for z/OS systems (continued)

Class name	Description
FSOBJ	Controls auditing (using SETROPTS LOGOPTIONS) of all access checks for z/OS UNIX file system objects except directory searches. Controls auditing (using SETROPTS AUDIT) of creation and deletion of z/OS UNIX file system objects. This class need not be active to control auditing. <a href="#">"5" on page 692</a>
FSSEC	Controls auditing (using SETROPTS LOGOPTIONS) of changes to the security data (FSP) for z/OS UNIX file system objects. This class need not be active to control auditing. When this class is active, it also controls whether ACLs are used during authorization checks to z/OS UNIX files and directories. <a href="#">"5" on page 692</a>
IPCOBJ	Controls auditing (using SETROPTS LOGOPTIONS) of access checks for interprocess communication (IPC) objects and changes to security information of IPC objects. Controls auditing (using SETROPTS AUDIT) of the creation and deletion of IPC objects. This class need not be active to control auditing. <a href="#">"5" on page 692</a>
PROCACT	Controls auditing (using SETROPTS LOGOPTIONS) of functions that look at data from, or affect the processing of, z/OS UNIX processes. This class need not be active to control auditing. <a href="#">"5" on page 692</a>
PROCESS	Controls auditing (using SETROPTS LOGOPTIONS) of changes to UIDs and GIDs of z/OS UNIX processes. Controls auditing (using SETROPTS AUDIT) of dubbing and undubbing of z/OS UNIX processes. This class need not be active to control auditing. <a href="#">"5" on page 692</a>
UNIXMAP	Contains profiles that are used to map z/OS UNIX UIDs to RACF user IDs and z/OS UNIX GIDs to RACF group names.
UNIXPRIV	Contains profiles that are used to grant z/OS UNIX privileges.

**Restrictions:**

1. Do not specify this class name on the GENCMD, GENERIC, and GLOBAL/NOGLOBAL operands of the SETROPTS command.
2. Do not specify this class name on the GLOBAL operand of SETROPTS or, if you do, the GLOBAL checking is not performed.
3. Do not specify this class name on the GENCMD and GENERIC operands of the SETROPTS command.
4. Do not specify this class name with any RACF command. This is a member class associated with a grouping class that has a special use.
5. Profiles are not allowed in this class.

## Appendix B. Summary of RACF commands and authorities

This topic summarizes the RACF commands and authorities.

### Summary of commands and their functions

RACF commands allow you to list, modify, add, and delete profiles for users, groups, connect entries, and resources. [Table 42 on page 693](#) shows, in alphabetic order, each of the commands and its functions.

*Table 42. Functions of RACF commands*

<b>RACF command</b>	<b>Command functions</b>
ADDGROUP	<ul style="list-style-type: none"> <li>• Define one or more new groups as a subgroup of an existing group.</li> <li>• Specify a model data set profile for a group.</li> <li>• Add a custom field for a group.</li> <li>• Define default DFP information for a group.</li> <li>• Define the z/OS UNIX information for a group.</li> <li>• Define a group as a universal group.</li> </ul>
ADDSD	<ul style="list-style-type: none"> <li>• RACF-protect one or more existing data sets.</li> <li>• RACF-define one or more data sets brought from another system where they were RACF-protected.</li> <li>• RACF-define generic data set profiles.</li> <li>• Create a new data set model profile.</li> </ul>
ADDUSER	<ul style="list-style-type: none"> <li>• Define one or more new users and connect the users to their default connect group.</li> <li>• Define a password, or a password phrase, for one or more users.</li> <li>• Specify a model data set profile for a user.</li> <li>• Add a custom field for a user.</li> <li>• Specify information related to one or more segments, such as the TSO and OMVS segments, of the user profile.</li> </ul>
ALTDSD	<ul style="list-style-type: none"> <li>• Change one or more discrete or generic data set profiles.</li> <li>• Protect a single volume of a multivolume, non-VSAM DASD data set.</li> <li>• Remove protection from a single volume of a multivolume, non-VSAM DASD data set.</li> </ul>
ALTGROUP	<ul style="list-style-type: none"> <li>• Change information in one or more group profiles (such as the superior group, owner, or model profile name).</li> <li>• Change or delete a custom field for a group.</li> <li>• Change or delete the default DFP information for a group.</li> <li>• Add, change, or delete information for the z/OS UNIX group.</li> </ul>

*Table 42. Functions of RACF commands (continued)*

<b>RACF command</b>	<b>Command functions</b>
ALTUSER	<ul style="list-style-type: none"> <li>• Change information in one or more user profiles (such as the owner, universal access authority, or security level).</li> <li>• Revoke or reestablish one or more users' privileges to access the system.</li> <li>• Specify logging of information about the user, such as the commands the user issues.</li> <li>• Change the password or password phrase for one or more users.</li> <li>• Add, change, or delete information related to one or more segments, such as the TSO and OMVS segments, of the user profile.</li> </ul>
CONNECT	<ul style="list-style-type: none"> <li>• Connect one or more users to a group.</li> <li>• Modify one or more users' connection to a group.</li> <li>• Revoke or reestablish one or more users' privileges to access the system.</li> </ul>
DELDSD	<ul style="list-style-type: none"> <li>• Delete one or more discrete or generic data set profiles.</li> <li>• Delete a discrete data set profile for a tape data set, while retaining the data set name in the TVTOC.</li> <li>• Remove a data set profile, but leave the data set RACF-indicated, when moving a RACF-protected data set to another system that has RACF.</li> </ul>
DELGROUP	<ul style="list-style-type: none"> <li>• Delete one or more groups and their relationship to the superior group.</li> </ul>
DELUSER	<ul style="list-style-type: none"> <li>• Delete one or more users and remove all of their connections to RACF groups.</li> </ul>
DISPLAY	<ul style="list-style-type: none"> <li>• Display users signed on to a RACF subsystem.</li> </ul>
HELP	<ul style="list-style-type: none"> <li>• Display the function and proper syntax of RACF commands.</li> </ul>
LISTDSD	<ul style="list-style-type: none"> <li>• List the details of one or more discrete or generic data set profiles, including the users and groups authorized to access the data sets.</li> <li>• Determine the most specific matching generic profile for a data set.</li> <li>• Perform a local refresh of generic DATASET profiles.</li> </ul>
LISTGRP	<ul style="list-style-type: none"> <li>• List the details of one or more group profiles, including the users connected to the group.</li> <li>• List only the information contained in a specific segment (for example, OMVS or CSDATA) of the group profile.</li> <li>• Display limited information if the group is a UNIVERSAL group.</li> </ul>
LISTUSER	<ul style="list-style-type: none"> <li>• List the details of one or more user profiles, including all of the groups to which each user is connected.</li> <li>• List only the information contained in a specific segment (for example, OMVS or CSDATA) of the user profile.</li> </ul>
PASSWORD or PHRASE	<ul style="list-style-type: none"> <li>• Change your own user password or password phrase.</li> <li>• Change one or more users' change interval for passwords and password phrases.</li> </ul>

Table 42. Functions of RACF commands (continued)

RACF command	Command functions
PERMIT	<ul style="list-style-type: none"> <li>• Give or remove authority to access a resource to specific users or groups.</li> <li>• Change the level of access authority to a resource for specific users or groups.</li> <li>• Copy the list of authorized users from one resource profile to another.</li> <li>• Delete an existing access list.</li> </ul>
RACDCERT	<ul style="list-style-type: none"> <li>• List information about the certificates for a specified RACF-defined user ID, or your own user ID.</li> <li>• Add a certificate and associate it with a specified RACF-defined user ID, or your own user ID, and set the TRUST status.</li> <li>• Check to see if a certificate has been defined to RACF.</li> <li>• Alter the TRUST status or label for a certificate.</li> <li>• Delete a certificate.</li> <li>• List a certificate contained in a data set and determine if it is associated with a RACF-defined user ID.</li> <li>• Add or remove a certificate from a key ring.</li> <li>• Create, delete, or list a key ring.</li> <li>• Generate a public/private key pair and certificate, replicate a digital certificate with a new public/private key pair, or retire the use of an existing private key.</li> <li>• Write (export) a certificate or certificate package to a data set.</li> <li>• Create a certificate request.</li> <li>• Create, alter, delete, or list a certificate name filter (user ID mapping).</li> <li>• Add, delete, or list a z/OS PKCS #11 token.</li> <li>• Bind a certificate to a z/OS PKCS #11 token.</li> <li>• Remove (unbind) a certificate from a z/OS PKCS #11 token.</li> <li>• Import a certificate (with its private key, if present) from a z/OS PKCS #11 token and add it to RACF.</li> </ul>
RACLINK	<ul style="list-style-type: none"> <li>• Define, approve, and delete (undefine) a user ID association.</li> <li>• List information related to a user ID association.</li> <li>• Establish password synchronization between user IDs.</li> </ul>
RACMAP	<ul style="list-style-type: none"> <li>• Create an association between a distributed user identity and a RACF user ID.</li> <li>• Define, delete, list, and query a distributed identity filter.</li> </ul>
RACPRIV	<ul style="list-style-type: none"> <li>• List, activate, and inactivate the user's write-down setting.</li> <li>• Reset the user's write-down setting to the installation-defined default.</li> </ul>
RACPRMCK	<ul style="list-style-type: none"> <li>• Validate the syntax of one or more RACF parmlib members.</li> <li>• Verify that the data within the RACF parmlib member is valid for the data set name table and range table.</li> </ul>

Table 42. Functions of RACF commands (continued)

RACF command	Command functions
RALTER	<ul style="list-style-type: none"> <li>• Change the discrete or generic profiles for one or more resources whose class is defined in the class descriptor table.</li> <li>• Define, change, or delete attributes for classes in the dynamic class descriptor table.</li> <li>• Maintain the global access checking table.</li> <li>• Maintain security categories and security levels.</li> <li>• Define, change, or delete information related to one or more segments of a general resource profile.</li> </ul>
RDEFINE	<ul style="list-style-type: none"> <li>• RACF-protect by a discrete or generic profile any resource whose class is defined in the class descriptor table.</li> <li>• Define attributes for classes in the dynamic class descriptor table.</li> <li>• Define entries in the global access checking table.</li> <li>• Define security categories and security levels.</li> <li>• Define information related to one or more segments of a general resource profile.</li> </ul>
RDELETE	<ul style="list-style-type: none"> <li>• Remove RACF-protection from one or more resources whose class is defined in the class descriptor table.</li> <li>• Delete the global access checking tables.</li> <li>• Delete the security category and security level tables.</li> <li>• Delete a class from the list of classes for which RACF saves RACLISTed results on the RACF database.</li> </ul>
REMOVE	<ul style="list-style-type: none"> <li>• Remove one or more users from a group and assign a new owner for any group data sets owned by the users.</li> </ul>
RESTART	<ul style="list-style-type: none"> <li>• Restart a function in the RACF subsystem address space.</li> <li>• Restart the connection to a specific member system on a multisystem node.</li> </ul>
RLIST	<ul style="list-style-type: none"> <li>• List the details of discrete or generic profiles for one or more resources whose class is defined in the class descriptor table.</li> <li>• List the contents of one or more segments of a general resource profile.</li> <li>• Perform a local refresh of generic general resource profiles.</li> </ul>
RVARY	<ul style="list-style-type: none"> <li>• Dynamically deactivate and reactivate the RACF function.</li> <li>• Dynamically deactivate and reactivate the RACF primary and backup database.</li> <li>• Switch the primary and backup RACF databases.</li> <li>• Deactivate resource protection, for any resource whose class is defined in the class descriptor table, while RACF is deactivated.</li> <li>• Select operational mode when RACF is enabled for sysplex communication.</li> </ul>



Table 42. Functions of RACF commands (continued)

RACF command	Command functions
SEARCH	<ul style="list-style-type: none"> <li>• Obtain a list of RACF profile names that meet the search criteria for a class of, resources, users, or groups. These profile names can then be displayed on your terminal. <ul style="list-style-type: none"> <li>– Profile names that contain a specific character string</li> <li>– Profiles for resources that have not been referenced for more than a specific number of days</li> <li>– Profiles that RACF recognizes as model profiles</li> <li>– Data set and general resource profiles that contain a level equal to or greater than the level you specify</li> <li>– User and resource profiles that contain a security label that matches the security label you specify.</li> <li>– User and resource profiles that contain a security level that matches the security level that you specify</li> <li>– User and resource profiles that contain an access category that matches the access category that you specify.</li> <li>– User profiles that contain an OMVS UID equal to the UID you specify.</li> <li>– Group profiles that contain an OMVS GID equal to the GID you specify.</li> <li>– Profiles for tape volumes that contain only data sets with an expiration date that matches the criteria you specify.</li> <li>– Profiles for data sets that reside on specific volumes (or VSAM data sets that are cataloged in catalogs on specific volumes).</li> <li>– Profiles for tape data sets, non-VSAM DASD data sets, or VSAM data sets.</li> </ul> </li> <li>• Format the selected profile names with specific character strings into a series of commands or messages and retain them in a CLIST data set.</li> <li>• Create a CLIST of the RACF profile names that meet a search criteria for a class of resources.</li> </ul>
SET	<ul style="list-style-type: none"> <li>• List information related to RACF remote sharing facility (RRSF) on the local node.</li> <li>• List the value for the template version following the FMID/APAR value.</li> <li>• Specify the name of a member of the RACF parameter library to be processed by RACF.</li> <li>• Enable and disable tracing for specified events.</li> <li>• Specify options for automatic command direction.</li> <li>• Improve performance of generic profiles by specifying GENERICANCHOR options.</li> </ul>

Table 42. Functions of RACF commands (continued)

RACF command	Command functions
SETROPTS	<p data-bbox="418 237 1382 264">Dynamically set system-wide options relating to resource protection, specifically:</p> <ul data-bbox="418 285 1464 1913" style="list-style-type: none"> <li data-bbox="418 285 1057 312">• Choose the resource classes that RACF is to protect.</li> <li data-bbox="418 323 854 350">• Gather and display RACF statistics.</li> <li data-bbox="418 361 1094 388">• Set the universal access authority (UACC) for terminals.</li> <li data-bbox="418 399 1089 426">• Specify logging of certain RACF commands and events.</li> <li data-bbox="418 436 894 464">• Permit list-of-groups access checking.</li> <li data-bbox="418 474 846 501">• Display options currently in effect.</li> <li data-bbox="418 512 1240 539">• Enable or disable generic profile checking on a class-by-class basis.</li> <li data-bbox="418 550 862 577">• Control user password syntax rules.</li> <li data-bbox="418 588 1208 615">• Activate checking for previous passwords and password phrases.</li> <li data-bbox="418 625 1398 701">• Limit unsuccessful attempts to access the system using incorrect passwords and password phrases.</li> <li data-bbox="418 711 1370 787">• Control maximum and minimum change intervals for passwords and password phrases.</li> <li data-bbox="418 798 813 825">• Control mixed-case passwords.</li> <li data-bbox="418 835 1036 863">• Enable the use of special characters in passwords.</li> <li data-bbox="418 873 911 900">• Enable the KDFAES password algorithm</li> <li data-bbox="418 911 786 938">• Warn of password expiration.</li> <li data-bbox="418 949 1464 1024">• Control global access checking for selected individual resources or generic names with selected generalized access rules.</li> <li data-bbox="418 1035 1175 1062">• Set the passwords for authorizing use of the RVARY command.</li> <li data-bbox="418 1073 1459 1100">• Initiate refreshing of in-storage generic profile lists and global access checking tables.</li> <li data-bbox="418 1110 1419 1138">• Enable or disable shared generic profiles for general resources in common storage.</li> <li data-bbox="418 1148 1435 1176">• Enable or disable shared profiles through RACLIST processing for general resources.</li> <li data-bbox="418 1186 1464 1262">• Activate or deactivate auditing of access attempts to RACF-protected resources based on installation-defined security levels.</li> <li data-bbox="418 1272 854 1299">• Activate enhanced generic naming.</li> <li data-bbox="418 1310 1105 1337">• Control the use of automatic data set protection (ADSP).</li> <li data-bbox="418 1348 1159 1375">• Activate profile modeling for GDG, group, and user data sets.</li> <li data-bbox="418 1386 1110 1413">• Activate protection for data sets with single-level names.</li> <li data-bbox="418 1423 894 1451">• Control logging of real data set names.</li> <li data-bbox="418 1461 987 1488">• Control the job entry subsystem (JES) options.</li> <li data-bbox="418 1499 834 1526">• Activate tape data set protection.</li> <li data-bbox="418 1537 1127 1564">• Control whether or not data sets must be RACF-protected.</li> <li data-bbox="418 1575 1013 1602">• Control the erasure of scratched DASD data sets.</li> <li data-bbox="418 1612 737 1640">• Activate program control.</li> <li data-bbox="418 1650 1386 1726">• Control whether a profile creator's user ID is automatically added to the profile's access list.</li> <li data-bbox="418 1736 1232 1764">• Make the name of the local RACF registry available to EIM services.</li> <li data-bbox="418 1774 1024 1801">• Control use of the dynamic class descriptor table.</li> <li data-bbox="418 1812 850 1839">• Control multilevel security options.</li> </ul>

Table 42. Functions of RACF commands (continued)

RACF command	Command functions
SIGNOFF	<ul style="list-style-type: none"> <li>Sign off users from a RACF subsystem.</li> </ul>
STOP	<ul style="list-style-type: none"> <li>Stop the RACF subsystem address space.</li> </ul>
TARGET	<ul style="list-style-type: none"> <li>List the operational and network protocol attributes of one or more RRSF nodes.</li> <li>Add or modify an RRSF node.</li> <li>Convert a remote RRSF node from one network protocol to another.</li> <li>Add a network protocol or modify protocol attributes for an RRSF node.</li> <li>Activate or inactivate an RRSF node or a protocol instance for an RRSF node.</li> <li>Specify a prefix and other attributes for the workspace data sets allocated and used by each RRSF node.</li> <li>Purge a workspace data set for an RRSF node.</li> <li>Delete an RRSF node or a protocol instance for an RRSF node.</li> </ul>

## Summary of authorities and commands

This topic summarizes the attributes and authorities that can be assigned to users, and the RACF commands and operands that can be issued for each authority. It provides information for the following categories:

- User attributes (SPECIAL or group-SPECIAL, AUDITOR or group-AUDITOR, ROAUDIT, OPERATIONS or group-OPERATIONS, and CLAUTH)
- Group authorities (USE, CREATE, CONNECT, JOIN)
- Access authorities (NONE, EXECUTE, READ, UPDATE, CONTROL, and ALTER)
- Other authorities not specified.

## The SPECIAL or group-SPECIAL attribute

If you have the SPECIAL or group-SPECIAL attribute, you can issue the commands and operands shown in [Table 43 on page 699](#).

Table 43. Commands and operands you can issue if you have the SPECIAL or group-SPECIAL attribute	
Command	Operands
ADDSD	With all operands
ADDGROUP	With all operands
ADDUSER	With all operands, but for group-SPECIAL user only when user also has CLAUTH(USER)
ALTDSD	With all operands except GLOBALAUDIT
ALTGROUP	With all operands
ALTUSER	With all operands except UAUDIT or NOUAUDIT. Also, you must have the SPECIAL attribute to use the NOEXPIRED operand or to issue the NOCLAUTH operand for a class name that is not in the class descriptor table (group-SPECIAL does not suffice).
CONNECT	With all operands
DELDSD	With all operands
DELGROUP	With all operands

*Table 43. Commands and operands you can issue if you have the SPECIAL or group-SPECIAL attribute (continued)*

<b>Command</b>	<b>Operands</b>
DELUSER	With all operands
LISTDSD	With all operands
LISTGRP	With all operands
LISTUSER	With all operands
PASSWORD or PHRASE	With all operands
PERMIT	With all operands
RALTER	With all operands except GLOBALAUDIT
RACDCERT	With all operands. You must have the SPECIAL attribute to issue the RACDCERT command. Group-SPECIAL does not suffice.
RACLINK	With all operands
RACMAP	With all operands. You must have the SPECIAL attribute to issue the RACMAP command. Group-SPECIAL does not suffice.
RDEFINE	With all operands
RDELETE	With all operands
REMOVE	With all operands
RLIST	With all operands
SEARCH	With all operands
SETROPTS	With all operands except AUDIT, NOAUDIT, CMDVIOL, NOCMDVIOL, APPLAUDIT, NOAPPLAUDIT, LOGOPTIONS, OPERAUDIT, NOOPERAUDIT, SAUDIT, NOSAUDIT, SECLABELAUDIT, NOSECLABELAUDIT, SECLEVELAUDIT, and NOSECLEVELAUDIT, which require the AUDITOR attribute. Users with the group-SPECIAL attribute can only issue REFRESH GENERIC and LIST.

## The AUDITOR or group-AUDITOR attribute

If you have the AUDITOR or group-AUDITOR attribute, you can issue the commands and operands shown in [Table 44 on page 700](#).

*Table 44. Commands and operands you can issue if you have the AUDITOR or group-AUDITOR attribute*

<b>Command</b>	<b>Operands</b>
ALTDSD	Only with GLOBALAUDIT
ALTUSER	Only with UAUDIT or NOUAUDIT
LISTDSD	With all operands, lists GLOBALAUDIT option
LISTGRP	With all operands
LISTUSER	With all operands, lists UAUDIT or NOUAUDIT operand
RALTER	Only with GLOBALAUDIT
RLIST	With all operands, lists GLOBALAUDIT option
SEARCH	With all operands

*Table 44. Commands and operands you can issue if you have the AUDITOR or group-AUDITOR attribute (continued)*

<b>Command</b>	<b>Operands</b>
SETROPTS	Only with APPLAUDIT, NOAPPLAUDIT, AUDIT, NOAUDIT, CMDVIOL, NOCMDVIOL, LOGOPTIONS, OPERAUDIT, NOOPERAUDIT, SAUDIT, NOSAUDIT, SECLABELAUDIT, NOSECLABELAUDIT, SECLEVELAUDIT, NOSECLEVELAUDIT, LIST, REFRESH GENERIC, or REFRESH RACLIST

## The ROAUDIT attribute

If you have the ROAUDIT attribute, you can issue the commands and operands shown in [Table 45 on page 701](#).

*Table 45. Commands and operands you can issue if you have the ROAUDIT attribute*

<b>Command</b>	<b>Operands</b>
LISTDSD	With all operands, lists GLOBALAUDIT option
LISTGRP	With all operands
LISTUSER	With all operands, lists UAUDIT or NOUAUDIT operand
RLIST	With all operands, lists GLOBALAUDIT option
SEARCH	With all operands
SETROPTS	Only with LIST

## The OPERATIONS or group-OPERATIONS attribute

If you have the OPERATIONS or group-OPERATIONS attribute, you can issue the commands and operands shown in [Table 46 on page 701](#).

*Table 46. Commands and operands you can issue if you have the OPERATIONS or group-OPERATIONS attribute*

<b>Command</b>	<b>Operands</b>
ADDSD	When adding new profiles for group data sets
LISTDSD <sup>1</sup>	With all operands except GLOBALAUDIT
RLIST	With all operands except GLOBALAUDIT
SEARCH	With all operands
SETROPTS	Only with REFRESH

## The CLAUTH attribute

If you have the CLAUTH attribute, you can issue the commands and operands shown in [Table 47 on page 701](#).

*Table 47. Commands and operands you can issue if you have the CLAUTH attribute*

<b>Command</b>	<b>Operands</b>
ADDUSER <sup>1</sup>	With all operands except OPERATIONS, NOOPERATIONS, SPECIAL, NOSPECIAL, AUDITOR, NOAUDITOR, ROAUDIT, or NOROAUDIT
ALTUSER <sup>2</sup>	Only with CLAUTH or NOCLAUTH
RALTER <sup>3, 5</sup>	Only with ADDVOL

Table 47. Commands and operands you can issue if you have the CLAUTH attribute (continued)	
Command	Operands
RDEFINE <sup>4, 6</sup>	With all operands (special rules apply to ADDMEM)
1	This command applies when you have the CLAUTH attribute of USER and you either are the owner of a group, have JOIN authority in the default group specified in the command, or the profile is within the scope of a group in which you have the group-SPECIAL attribute.
2	This command applies when you have the CLAUTH attribute for the class to be added or deleted, the class name is in the class descriptor table (CDT), and either you are the owner of the user's profile, or the profile is within the scope of a group in which you have the group-SPECIAL attribute.
3	This command applies when you have the CLAUTH attribute of TAPEVOL and you also have sufficient authority to issue the command.
4	These commands apply when you have the CLAUTH attribute for the specified class.
5	For ADDMEM, special rules apply, depending on access to individual resources. For more information, see the description of the ADDMEM operand in <a href="#">z/OS Security Server RACF Command Language Reference</a> .
6	The ADDMEM keyword adds members to a profile in a grouping class. Classes that have grouping classes are called member classes. For ADDMEM, special rules apply, depending on access to individual resources. Creating a profile in a member class is similar to specifying the ADDMEM keyword for a profile in the corresponding grouping class. In both cases, if the name being defined is RACF-protected either by a member class profile or by a member of a profile in the related grouping class, more than CLAUTH authority is required. For more information, see the description of the ADDMEM operand in <a href="#">z/OS Security Server RACF Command Language Reference</a> .

## Group authority

If you have a group authority, you can issue the commands and operands shown in [Table 48 on page 702](#).

Table 48. Commands and operands you can issue if you have a group authority	
Group authorities	Commands and operands you can issue if you have this authority
USE	For group resources, the authority allowed the group.
CREATE	<b>ADDSD<sup>1</sup></b> with all operands except NOSET
CONNECT	<b>ADDSD<sup>1</sup></b> with all operands except NOSET <b>ALTUSER</b> only with GROUP, AUTHORITY or UACC <b>CONNECT</b> with all operands except SPECIAL, NOSPECIAL, OPERATIONS, NOOPERATIONS, AUDITOR, or NOAUDITOR <b>LISTGRP</b> only with group name <b>REMOVE</b> with all operands

Table 48. Commands and operands you can issue if you have a group authority (continued)	
Group authorities	Commands and operands you can issue if you have this authority
JOIN	<b>ADDGROUP<sup>2</sup></b> with all operands <b>ADDSD<sup>1</sup></b> with all operands except NOSET <b>ADDUSER<sup>3</sup></b> with all operands except OPERATIONS, SPECIAL, AUDITOR, or ROAUDIT <b>ALTGROUP<sup>4</sup></b> only with SUPGROUP <b>ALTUSER</b> only with GROUP, AUTHORITY or UACC <b>CONNECT</b> with all operands except SPECIAL, NOSPECIAL, OPERATIONS, NOOPERATIONS, AUDITOR, or NOAUDITOR <b>DELGROUP<sup>2</sup></b> with all operands <b>LISTGRP</b> only with group name <b>REMOVE</b> with all operands
<p><b>1</b> This command applies to group data sets only.</p> <p><b>2</b> This command applies to the superior group.</p> <p><b>3</b> This command applies only if you have the JOIN group authority in the default group specified in the ADDUSER command and if you also have the CLAUTH(USER) attribute.</p> <p><b>4</b> This command applies to current and new superior groups. You can have JOIN authority in one group and be owner of or be connected with the group-SPECIAL attribute to another group.</p>	

## Access authority

If you have an access authority, you can issue the commands and operands shown in [Table 49 on page 703](#).

Table 49. Commands and operands you can issue if you have an access authority	
Access authorities	Commands and operands you can issue if you have this authority
NONE EXECUTE	None
READ UPDATE CONTROL	<b>LISTDSD</b> with all operands except AUTHUSER or ALL <b>RLIST</b> with all operands except AUTHUSER or ALL <b>SEARCH</b> with all operands

*Table 49. Commands and operands you can issue if you have an access authority (continued)*

<b>Access authorities</b>	<b>Commands and operands you can issue if you have this authority</b>
ALTER	<b>ALTDSD<sup>1</sup></b> with all operands except OWNER, NOSET or GLOBALAUDIT <b>DELDSD<sup>1,2</sup></b> with all operands except NOSET <b>LISTDSD<sup>1,2</sup></b> with all operands <b>PERMIT<sup>1</sup></b> with all operands <b>RALTER<sup>1</sup></b> with all operands except OWNER, ADDVOL <sup>2</sup> or GLOBALAUDIT <b>RDELETE<sup>1</sup></b> with all operands <b>RLIST<sup>1</sup></b> with all operands
<b>1</b>	This command applies to discrete profiles only. Note that ALTER access can be restricted. See, <a href="#">Chapter 9, “Limiting ALTER access authority in discrete profiles,”</a> on page 259 for more information.
<b>2</b>	This command applies to ADDVOL operand only if you also have CLAUTH attribute for TAPEVOL.

## Profile ownership authority

If you own a profile, you can issue the commands and operands shown in [Table 50 on page 704](#).

*Table 50. Commands and operands you can issue if you own a profile*

<b>Owner of RACF profile</b>	<b>Commands and operands you can issue if you have this authority</b>
Owner of user profile	<b>ALTUSER<sup>1</sup></b> only with user ID, NAME, OWNER, DFLTGRP, DATA, GRPACC, NOGRPACC, ADSP, NOADSP, REVOKE, NOREVOKE, RESUME, NORESUME, PASSWORD, NOPASSWORD, PHRASE, NOPHRASE, OIDCARD, NOOIDCARD, CLAUTH or NOCLAUTH <b>DELUSER</b> with all operands <b>LISTUSER</b> with all operands <b>RACLINK</b> with all operands



Table 50. Commands and operands you can issue if you own a profile (continued)	
Owner of RACF profile	Commands and operands you can issue if you have this authority
Owner of group profile	<b>ADDGROUP<sup>2</sup></b> with all operands <b>ADDUSER<sup>3</sup></b> with all operands except OPERATIONS, SPECIAL, AUDITOR, or ROAUDIT <b>ALTGROUP<sup>4</sup></b> with all operands <b>ALTUSER</b> only with GROUP, AUTHORITY or UACC <b>CONNECT</b> with all operands except SPECIAL, NOSPECIAL, OPERATIONS or NOOPERATIONS <b>DELGROUP<sup>5</sup></b> with all operands <b>LISTGRP</b> with all operands <b>REMOVE</b> with all operands
Owner of resource profile	<b>ALTDSD</b> with all operands except NOSET or GLOBALAUDIT <b>DELDSD</b> with all operands except NOSET <b>LISTDSD</b> with all operands <b>PERMIT</b> with all operands <b>RALTER<sup>6</sup></b> with all operands except GLOBALAUDIT <b>RDELETE</b> with all operands <b>RLIST</b> with all operands <b>SEARCH</b> with all operands

*Table 50. Commands and operands you can issue if you own a profile (continued)*

<b>Owner of RACF profile</b>	<b>Commands and operands you can issue if you have this authority</b>
<b>1</b>	This command applies to CLAUTH or NOCLAUTH only if you have the CLAUTH attribute for the class to be added or deleted, and the class name is in the class descriptor table (CDT).
<b>2</b>	This command applies to the superior group.
<b>3</b>	This command applies to the default group specified and only if you have the CLAUTH attribute of USER.
<b>4</b>	This command applies to current and new superior groups. You can have JOIN authority in one group and be owner of another group.
<b>5</b>	This command applies to the superior group or group to be deleted.
<b>6</b>	This command applies to the ADDVOL operand only when you also have CLAUTH attribute of TAPEVOL.

## Other authorities

Table 51 on page 706 shows the commands and operands you can issue for reasons not already covered previously.

*Table 51. Commands and operands you can issue for miscellaneous reasons*

<b>User ID relationship</b>	<b>Commands and operands you can issue if you have this authority</b>
User ID is current user	<b>ALTUSER</b> only with NAME or DFLTGRP <b>LISTUSER</b> only with user ID <b>PASSWORD or PHRASE</b> only with PASSWORD or INTERVAL
User ID is high-level qualifier of data set name (or qualifier supplied by a command installation exit)	<b>ADDSD</b> with all operands <b>ALTDSD</b> with all operands except OWNER or GLOBALAUDIT <b>DELDSD</b> with all operands <b>LISTDSD</b> with all operands <b>PERMIT</b> with all operands <b>SEARCH</b> with all operands
None	<b>RVARY<sup>1</sup></b> with all operands

Table 51. Commands and operands you can issue for miscellaneous reasons (continued)

User ID relationship	Commands and operands you can issue if you have this authority
None	<p>RACF MVS Operator Commands:</p> <p><b>DISPLAY</b>            Authority granted by OPERCMDS class:            See <a href="#">Table 22 on page 242</a> and <a href="#">z/OS Security Server RACF Command Language Reference</a>.</p> <p><b>RACPRMCK</b>            Authority granted by OPERCMDS class</p> <p><b>RESTART</b>            Authority granted by OPERCMDS class</p> <p><b>SET</b>            Authority granted by OPERCMDS class</p> <p><b>SIGNOFF</b>            Authority granted by OPERCMDS class:            See <a href="#">Table 22 on page 242</a> and <a href="#">z/OS Security Server RACF Command Language Reference</a>.</p> <p><b>STOP</b>            Authority granted by OPERCMDS class</p> <p><b>TARGET</b>            Authority granted by OPERCMDS class</p> <p><b>Any RACF TSO command issued as an operator command</b></p>
1	<p>Although no special authority is needed to issue this command, the system operator must supply the appropriate RVARYPW password, as established by the SETROPTS command with the RVARYPW operand, to approve any change in RACF status. Authority granted by OPERCMDS class, or the system operator must supply the appropriate RVARYPW password, as established by the SETROPTS command with the RVARYPW operand, to approve any change in RACF status.</p>



## Appendix C. Listings of RACF supplied certificates

This topic contains a complete listing for each supplied certificate-authority (CA) certificate as it is supplied with RACF and does not reflect any changes you might have made to these certificates. The supplied CA certificates are the following:

1. STG Code Signing Certificate Authority (for RACF program signature verification for release z/OS V2R1 and earlier)
2. STG Code Signing Certificate Authority - G2 (for RACF program signature verification for release z/OS V2R2 and later)

For more information about the certificate authority certificates that RACF supplies, see [“Supplied digital certificates”](#) on page 589. For instructions about how to use a supplied certificate, see [“Steps to begin using a supplied CA certificate”](#) on page 589.

For steps to begin using the STG Code Signing Certificate Authority (for RACF program signature verification for release z/OS V2R1 and earlier), see [“Steps for preparing RACF to verify signed programs \(one-time setup\)”](#) on page 340.

The following listings of the supplied CA certificates were created using the RACDCERT LIST command, which is the preferred method for listing certificate information. The RACF profiles that contain the supplied certificates can also be listed using the RLIST and SEARCH commands. For examples, see [Figure 50](#) on page 555 and [Figure 51](#) on page 556.

**Note:** Start times and end times are listed in Greenwich Mean Time (GMT).

1. **STG Code Signing Certificate Authority (for RACF program signature verification for release z/OS V2R1 and earlier)**

```
Label: STG Code Signing CA
Certificate ID: 2Q1JmZmDhZmjgeLjx0DD1oSFQ0KJh5WJ1YdAw8FA
Status: NOTRUST
Start Date: 2008/07/02 04:00:00
End Date: 2028/07/01 03:59:59
Serial Number:
>00<
Issuer's Name:
>CN=STG Code Signing CA.OU=IBM Code Signing.O=IBM Corporation.C=US<
Subject's Name:
>CN=STG Code Signing CA.OU=IBM Code Signing.O=IBM Corporation.C=US<
Signing Algorithm: sha1RSA
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 2048
Private Key: NO
Certificate Fingerprint (SHA256):
A0:88:FC:E8:DD:EB:38:10:79:AD:F1:F4:84:43:D2:27:
65:48:7B:18:AB:3A:BE:21:2F:E1:6A:45:F6:8C:A3:51
Ring Associations:
*** No rings associated ***
```

2. **STG Code Signing Certificate Authority - G2 (for RACF program signature verification for release z/OS V2R2 and later)**

```
Label: STG Code Signing CA - G2
Certificate ID: 2Q1JmZmDhZmjgeLjx0DD1oSFQ0KJh5WJ1YdAw8FAYEDH8kBA
Status: NOTRUST
Start Date: 2013/06/02 04:00:00
End Date: 2033/06/01 03:59:59
Serial Number:
>00<
Issuer's Name:
>CN=STG Code Signing CA - G2.OU=IBM Code Signing.O=IBM Corporation.C=U<
>S<
Subject's Name:
>CN=STG Code Signing CA - G2.OU=IBM Code Signing.O=IBM Corporation.C=U<
>S<
Signing Algorithm: sha256RSA
```

## Certificate listings

```
Key Usage: CERTSIGN
Key Type: RSA
Key Size: 2048
Private Key: NO
Certificate Fingerprint (SHA256):
    38:AF:8E:66:CB:9B:36:97:82:EF:0A:7D:51:D5:2A:61:
    65:D9:E8:67:A8:8A:53:0B:31:FE:6B:80:59:4C:74:E5
Ring Associations:
*** No rings associated ***
```

## Appendix D. Security for system data sets

This topic contains some guidelines for defining UACC values for system data sets.

Table 52 on page 711 lists the UACC values that should be assigned to many of the system data sets. Note that your security policy might require a UACC of NONE for some data sets. For example, you can specify UACC(NONE) for macro libraries if you give READ access to programmers who need to assemble or compile programs that use those libraries. For a discussion of system data sets, see [“Protecting DASD system data sets”](#) on page 166.

For guidelines about the security labels to assign system data sets on a multilevel-secure system, see [z/OS Planning for Multilevel Security and the Common Criteria](#).

You should consider creating a generic profile to protect system data sets, as follows:

```
ADDSD 'SYS1.*' UACC(NONE) SECLABEL(SYSHIGH)
```

Specifying a UACC of NONE for the SYS1.\* profile protects any system data sets that are added to the system by new products. If new system data sets need a UACC higher than NONE or a SECLABEL of SYSLOW, you can create more specific profiles for them.

You should also create specific profiles for particular data sets (or groups of data sets, such as SYS1.DUMPxx data sets), using the information in [Table 52 on page 711](#).

For any data set that is listed with a UACC of READ or higher, you should consider creating an entry in the global access checking table. For more information, see [“Setting up the global access checking table”](#) on page 194.

For system data sets that are listed in the table with a UACC higher than NONE, you might prefer to specify UACC(NONE) and create an access list entry of ID(\*) ACCESS(*access-authority*). This entry prevents users who are not defined to RACF from accessing the data sets.

Restricted users enter the system with a user ID that is defined with the RESTRICTED attribute, and might be shared by many users. Restricted users are prevented from gaining access to protected resources through the global access checking table, UACC, or the ID(\*) entry on the access list. User IDs defined with the RESTRICTED attribute must be specifically authorized with sufficient authority on the access list of any protected resource they must access. Therefore, to allow restricted users to access any data set listed with UACC of READ or higher in [Table 52 on page 711](#), each user ID with the RESTRICTED attribute must be specifically authorized at the level of access indicated by the UACC value.

*Table 52. UACC values for system data sets*

Data set	UACC	Comments
APF libraries	NONE	Authorized program facility libraries.
Checkpoint data sets	NONE	
Distribution library data sets	NONE	
ISPF panel libraries	READ	Panel definitions, skeletons, CLISTs, and so forth. Specify UACC(NONE) if access must be restricted.
JES2 offload data sets	NONE	
Load libraries	READ	
Master catalog	READ	

Table 52. UACC values for system data sets (continued)

Data set	UACC	Comments
Page data sets	NONE	Includes PLPA, common, and local page data sets. See <a href="#">z/OS MVS Initialization and Tuning Guide</a> .
PSF secure font data sets	NONE	
PSF secure overlay data sets	NONE	
PSF secure page segment data sets	NONE	
RMF data sets	NONE	VSAM data sets.
Security definitions data sets	NONE	
SMP data sets	NONE	
Swap data sets	NONE	
SYS1.AMACLIB	READ	
SYS1.AMODGEN	READ	
SYS1.ASAMPLIB	READ	
SYS1.BROADCAST	READ	
SYS1.CMDLIB	READ	
SYS1.DAE	NONE	
SYS1.DUMPxx	NONE	See <a href="#">z/OS MVS Initialization and Tuning Reference</a> .
SYS1.HELP	READ	TSO online help.
SYS1.IMAGELIB	NONE	
SYS1.JESPARM	NONE	
SYS1.LINKLIB	READ	
SYS1.LOGREC	NONE	
SYS1.LPALIB	READ	UACC can be NONE or READ depending on your installation's security policy.
SYS1.MACLIB	READ	
SYS1.MANx	NONE	SMF data sets. See <a href="#">z/OS MVS Initialization and Tuning Reference</a> .
SYS1.MIGLIB	READ	
SYS1.MODGEN	READ	
SYS1.NUCLEUS	READ	
SYS1.OVERLIB	READ	
SYS1.PARMLIB	NONE	NONE for all of the data sets in the current PARMLIB concatenation.
SYS1.PROCLIB	READ	



Table 52. UACC values for system data sets (continued)

<b>Data set</b>	<b>UACC</b>	<b>Comments</b>
SYS1.RACF	NONE	Includes data sets that comprise the active and backup RACF databases, split data sets created with the IRRUT400 utility, and archival copies. Your installation might use other data set names.
SYS1.SAMPLIB	READ	
SYS1.SAXREXEC	READ	System REXX library, or any libraries defined in the REXXLIB concatenation. UACC can be NONE or READ depending on your installation's security policy.
SYS1.SIEALNKE	READ	
SYS1.SIEAMIGE	READ	
SYS1.SIEASID	READ	
SYS1.STGINDEX	NONE	
SYS1.SVCLIB	NONE	
SYS1.TELCMLIB	READ	
SYS1.UADS	NONE	
SYS1.VTOCIX...	NONE	
SYS1.VVDS...	NONE	
SYS1.VTAMLIB	READ	
SYS1.VTAMLST	NONE	
Trace data sets	NONE	
User catalogs	READ	When a user catalog contains non SMS managed data sets, its UACC should be UPDATE.
User dump data sets	NONE	



## Appendix E. Debugging problems in the RACF database

This topic explains how to debug the profile definitions in the RACF database and the current RACF options. It also discusses RACF authorization checking and refresh timing.

This topic describes how to debug problems with your profile definitions. It is designed to help you determine how your current RACF options and profiles work for you. For example, if users have access to resources that they should not have, or if users do not have access to resources that they should have, this topic might be helpful in determining how to correct the problem.

**Note:** If you have a problem that you suspect is caused by RACF, and not by your use of RACF, see [z/OS Security Server RACF Diagnosis Guide](#) for information on correcting the problem.

### Checklist: Resolving problems when access is denied unexpectedly

When a user or job requires access to a protected resource, and RACF denies the requested access, you must often analyze the problem before deciding what action to take. Here are some basic steps to take when analyzing access problems:

- First, get the complete text of the ICH408I message that RACF issued when denying the access. Also, take note of any other messages that were issued. The ICH408I message often indicates what profile RACF checked, and what class the profile was in, when denying access. For a description of the ICH408I message, see [z/OS Security Server RACF Messages and Codes](#).

**Note:** List the profile you believe should have given access. If it contains an **\***, **%**, or **&**, make sure it is also followed by a **G**. If you do not have the **G**, it is not generic and would not be granting any access.

- If the ICH408I message indicates that access was denied because of a revoked user ID, you might want to resume that user ID. Check if the user ID is associated with the started procedure. If there was a user ID associated with the started procedure, this started procedure could not have begun successfully. After you resume the user ID, you must restart the started procedure or re-IPL.
- If the ICH408I message indicates that access was denied because of a profile, check the profile listing to make sure the user or job should have access. You should check not only the UACC and access lists, but the security classification of the resource profile and the user. Note that installation exits (both RACF exits and certain exits in products that call RACF, such as JES) can affect a user's access to resources.

To check the user's access to the resource, ask the user who had the problem to list the profile protecting the resource and check the YOUR ACCESS field of the listing.

- For general resources, the user can issue the RLIST command.
- For data sets, the user can issue the LISTDSD command.

If no data set profile is found, or the data set profile allows the user sufficient access, check if the TAPEVOL class is active. If active, ask the user to issue the RLIST command against the FROM profile in the TAPEVOL class.

- If the user cannot issue the LIST command, do it yourself. In the listing supplied by RACF, the following fields can, by themselves, deny access to a user:
  - The security level or security category, or both
  - The security label
  - The standard access list (ID and ACCESS)
  - The universal access authority (UACC)

- If the profile listing indicates that the user or job should have access, and the profile is in a class for which SETROPTS RACLIST processing or SETROPTS GENLIST processing is in effect, make sure that any in-storage profiles are refreshed by doing the following:
  - If the resource is protected by a generic profile in a class that is not RACLISTed or GENLISTed, ask the user to log off and log on again, or resubmit the job. This refreshes the user's copy of the profile.
  - Issue the SETROPTS GENERIC(*classname*) REFRESH or SETROPTS RACLIST(*classname*) REFRESH command again.

For information about SETROPTS REFRESH processing on shared systems, see [“Refreshing shared systems \(REFRESH option\)”](#) on page 131.

- You can use audit records to gather information that you would not otherwise have. In particular, you can request that audit records be generated for all accesses to protected resources, or for only failed accesses. You can also request that audit records be kept for a particular user. With the auditing in effect, you can use the RACF report writer to view the access requests associated with the access requests.

**Note:** In some cases (such as some resources in the OPERCMDS class), a RACROUTE request from a resource manager can include a "log string", which is a string of characters to be placed in the SMF record if the access is audited. The log string can be useful in determining what kind of action the user was taking. For example, the log string might include the exact operator command, as the operator issued it.

## Checklist: Resolving problems when access is allowed incorrectly

You might see occurrences when a user or job obtains access to a protected resource and you believe that the user should not have that access. There are many reasons why this could happen. The following checklist can eliminate some of them:

- Make sure that RACF is active, and that message ICH520I has been issued for this IPL. (To see if RACF is active, issue a RACF command, such as the LISTUSER command. If RACF is not active, the command will fail.)

**Note:** Even though RACF is active, resource managers for which external security is optional (such as CICS) might not be using RACF. You must ensure that such resource managers are calling RACF. Also, there might be other options that control which general resource classes are protected by RACF. Therefore, you must make sure that the resource manager is calling RACF for the particular resource class. For CICS, you must ensure that the particular CICS region is using external security. For information specifically related to RACF and CICS, visit [CICS Transaction Server for z/OS](http://www.ibm.com/docs/en/cics-ts) ([www.ibm.com/docs/en/cics-ts](http://www.ibm.com/docs/en/cics-ts)).

- If the resource is a general resource (in other words, not protected by a profile in the DATASET class), make sure that the general resource class is active. For example, for tape volumes, make sure that the TAPEVOL class is active. To do this, issue the SETROPTS LIST command.
- Check for a global access checking table entry for the resource. An entry in the global access checking table can allow access for all users, except those with the RESTRICTED attribute, when a profile protecting the resource would deny access. For example, for data sets, enter:

```
RLIST GLOBAL DATASET
```

**Note:** Do not ignore the presence of entries containing &RACGPID or &RACUID.

- If the profile is a generic profile, use the SETROPTS LIST command to ensure that generic profile checking is in effect for the class.
- Make sure you know which profile is actually protecting the resource. For example, a more specific profile might actually be used instead of the generic profile you believe protects the resource. The more specific profile might grant the user the access. To do this, issue the LISTDSD and RLIST command, specifying the resource name. For the LISTDSD command, if you receive a message indicating that no profile is found, issue the command again with the GENERIC operand to check for any generic profiles that might protect the resource.

- Check the user's access to the resource. You can do this in two ways:

- Ask the user to list the profile protecting the resource. For example, the user can issue the LISTDSD and RLIST command, specifying the resource name. For the LISTDSD command, if the user receives a message indicating that no profile is found, have the user issue the command again with the GENERIC operand to check for any generic profiles that might protect the resource.

Have the user check the YOUR ACCESS field in the profile listing. If this field indicates that the user or job should have access, use the steps described in [“Authorizing access to RACF-protected resources” on page 719](#) to analyze the profile for reasons why the user or job has that access.

- If the user cannot issue the LIST command, do it yourself. In the listing supplied by RACF, the following fields can, by themselves, allow access to a user:
  - For data sets and minidisks, the high-level qualifier
  - The standard access list
  - The conditional access list
  - UACC
  - WARNING

If list-of-groups processing is in effect on your system, check to see if a group of which the user is a member is in the access list. Check both the standard access list and the conditional access list. Also, note that installation exits (both RACF exits and certain exits in products that call RACF, such as JES) can affect a user's access to resources.

- If your analysis of the protecting profile shows that the user should not have access, continue with the following checks.
- If the SETROPTS RACLIST processing or SETROPTS GENLIST processing is in effect, make sure that any in-storage profiles are refreshed.
  - If the resource is protected by a generic profile in a class that is not RACLISTed or GENLISTed, ask the user to log off and log on again, or resubmit the job. This refreshes the user's copy of the profile.
  - Issue the SETROPTS GENERIC(classname) REFRESH or SETROPTS RACLIST(classname) REFRESH command again.

For information about SETROPTS REFRESH processing on shared systems, see [“Refreshing shared systems \(REFRESH option\)” on page 131](#).

- You can use audit records to gather information that you would not otherwise have. In particular, you can request that audit records be generated for all accesses to protected resources, or for only successful accesses. You can also request that audit records be kept for a particular user. With the auditing in effect, you can use the RACF report writer to view the access requests associated with the access requests.

**Note:** In some cases (such as some resources in the OPERCMDS class), a RACROUTE request from a resource manager can include a "log string", which is a string of characters to be placed in the SMF record if the access is audited. The log string can be useful in determining what kind of action the user was taking. For example, the log string might include the exact operator command, as the operator issued it.

## When changes to data set profiles take effect

If a user is currently using a data set, changing the data set profile protecting the data set might not affect the user's current access until that user logs on again.

The change affects the user's access immediately in the following cases:

- If the user is not logged on. You can check to see if a user is logged on with the TSO STATUS command:

```
STATUS userid
```

If the user is logged on, the system displays a message indicating that a job with the letters TSU in it is executing.

- If the user is logged on and has not yet opened the data set or a data set protected by the same generic profile (for example, by browsing or editing).

If the user is logged on and has opened the data set, and you change his access, two situations could occur:

- If the profile is a discrete profile, the user's access changes after closing the data set.
- If the profile is a generic profile, the user's access changes after *one* of the following occurs:
  - The user issues the LISTDSD command as follows:

```
LISTDSD DATASET(data-set-protected-by-the-profile) GENERIC
```

This places a fresh copy of the profile in the user's address space.

- A SETROPTS GENERIC(DATASET) REFRESH is issued on the system the user is logged on to or from another system in the RACF sysplex data sharing group, if RACF is enabled for sysplex communication.
- The user references more than the default or installation-specified number of data sets with different high-level qualifiers for this address space, and the data sets are protected by generic profiles. (The installation specifies the number with the GENERICANCHOR option of the SET command. Four is the default number.)
- The user logs off and then logs back on.

## Authorization checking for RACF-protected resources

---

This topic includes the following information:

- [“When authorization checking takes place and why” on page 718](#)
- [“Authorizing access to RACF-protected resources” on page 719](#)
- [“Pictorial view of RACF authorization checking” on page 723](#)
- [“Authorizing access to z/OS UNIX files and directories” on page 728](#)
- [“Authorizing access to RACF-protected terminals” on page 730](#)
- [“Authorizing access to consoles, JES input devices, APPC partner LUs, or IP addresses” on page 731](#)
- [“Authorization checking for RACROUTE REQUEST=FASTAUTH requests” on page 732](#)
- [“Authorizing access to RACF-protected applications” on page 733](#)
- [“Security label authorization checking” on page 733](#)

## When authorization checking takes place and why

When a user requests access to a RACF-protected resource (such as a data set), the resource manager issues the RACROUTE macro with REQUEST=AUTH specified (or the RACHECK macro<sup>17</sup>). For ease of reference, this topic calls such a request a RACF authorization request.

Based on the specifications on the RACF authorization request, RACF determines whether the requesting user is authorized to access the resource.

- If the user is authorized to the resource, RACF returns a "successful" return code to the resource manager. The resource manager then allows the request to complete.
- If the user is not authorized to the resource, RACF returns an "unauthorized" return code to the resource manager. The resource manager then fails the request.

---

<sup>17</sup> If the RACHECK macro is issued instead of RACROUTE, authorization processing begins at Step [“7” on page 719](#).

RACF issues a message indicating that the user is not authorized to the resource.

- If the resource is not protected (for example, if no profile exists for it), RACF returns the default return code for the class.

For general resource classes, the default return code is the "not protected" return code, unless otherwise specified in the class descriptor table (CDT) entry for the class.

For the DATASET class, the default return code is the "not protected" return code, unless the SETROPTS PROTECTALL(FAILURES) option is in effect, in which case the default return code is the "not authorized" return code.

If the "not protected" return code is issued, the resource manager then either fails or allows the request. Most resource managers allow the request.

RACF issues a message indicating that the resource is not protected.

**Note:**

1. SMF log records or messages might be generated, depending on the options in effect and whether RACF granted or denied access to the resource.
2. When checking authorization for a directed command, RACF uses the authorization of the target user ID, not the issuing user ID.

## Authorizing access to RACF-protected resources

To perform authorization checking for RACF-protected resources, RACF makes the following checks. RACF stops processing when the request is granted or denied.

For a pictorial view of the following steps, see [Figure 65 on page 724](#) through [Figure 67 on page 726](#).

**Note:** The following steps do not apply to accessing z/OS UNIX resources. For authorization steps related to accessing z/OS UNIX resources, see [“Authorizing access to z/OS UNIX files and directories” on page 728](#).

1. The SAF router exits can grant or deny access before RACF authorization processing occurs.

**Note:** Before master scheduler initialization (MSI), this is the ICHRTX01 exit routine. After master scheduler initialization (MSI), this is the ICHRTX00 exit routine.

2. If the entry for the class in the RACF router table has NONE specified, RACF returns the "not protected" return code. This also occurs if the caller specified the REQSTOR and SUBSYS operands on the RACROUTE macro, did not also specify DECOUPL=YES or give the REQSTOR and SUBSYS information in the RACF router table entry.
3. If the caller is a contained user ID, RACF returns the "caller contained" return code. If an ACEE was passed in for a contained user ID, RACF returns the "contained delegate" return code.
4. If RACF is not active, RACF returns the "not protected" return code.
5. For general resource classes, if the class of the resource is not active, RACF returns the "not protected" return code.
6. If the RACF class must be RACLISTed, as specified in the class descriptor table (CDT), but is not currently RACLISTed, RACF returns the "not protected" return code.
7. If the user requesting access is "trusted" or "privileged", RACF grants the request. See the following:
  - If the user has the trusted attribute, RACF grants the request (unless the CSA or PRIVATE operand was specified on the authorization request).

Such requests can be audited only by using the LOGOPTIONS operand on the SETROPTS command (which audits access requests issued by all users) or the UAUDIT operand on the ALTUSER command (which audits all access requests by a particular user).

  - If the user has the privileged attribute, RACF grants the request (unless the CSA or PRIVATE operand was specified on the authorization request). Such requests cannot be audited.
8. RACF invokes the naming convention table if:

- The naming convention routine exists
- The resource being checked is a CLASS data set

The naming convention table can continue REQUEST=AUTH processing or deny the request.

9. The REQUEST=AUTH preprocessing exit (ICHRX01) can grant or deny the request.
10. If the requesting user has a default user token (created by SAF), RACF denies the request.
11. If SETROPTS MLQUIET is in effect (see [“Quiescing RACF activity \(MLQUIET option\)”](#) on page 134), RACF denies the request unless the user has the SPECIAL attribute, has the privileged or trusted attribute, or is logged on at a system console.
12. If the user ID in the RTOKEN for the resource matches the user ID in the UTOKEN for the user making a request, RACF grants the request. This allows a user to cancel one of the user's own jobs using the TSO CANCEL command without being affected by the user's current security label.

RTOKEN processing typically applies to resources in the JESSPOOL class, but it might not apply to all JESSPOOL resources based on processing by the application or resource manager.

13. If global access checking is active for the class, RACF searches the global access table (unless the CSA or PRIVATE operand was specified on the authorization request). If RACF finds a matching entry that allows access to the resource, RACF grants the request for all users, except those with the RESTRICTED attribute.
14. RACF looks for a profile in storage or in the RACF database. If no profile is found that protects the resource, RACF tentatively sets the return code to 4 for the DATASET class, or, for a general resource class, to the default return code defined for the class. Skip to step [“31”](#) on page 723, where the post processing exit may change the return code. See the entry for the class in the class descriptor table (CDT) described in [z/OS Security Server RACF Macros and Interfaces](#).) Specifically, no profile is found in the following cases:

- Profiles for the class exist in the user's storage or in a data space, but no profile matches the resource name.
- Profiles for the class do not exist in the user's storage, in a data space, or in the RACF database.

**Note:** If you expect generic profiles to be used by RACF authorization checking, list their profile names using the SEARCH command. If the profile names listed by the SEARCH command are *not* followed by (G), RACF does not treat them as generic profiles. To recover, perform the following steps:

- a. If the profiles have complicated specifications, such as long access lists, you might want to define temporary profiles modeled on them before deleting them. These may be used in step [“14.c”](#) on page 720 to recreate the profiles. Specify the FROM operand on the RDEFINE command:

```
RDEFINE dclass dprofilename FROM(profilename) FCLASS(classname)
```

**Note:** The FROM operand does not preserve all fields and segments of the profile. See [“Profile modeling”](#) on page 36 to determine if this will work for the profile in question.

- b. Delete the profiles using the NOGENERIC operand:

```
RDELETE classname profilename NOGENERIC
```

- c. Define the profiles again, possibly using the temporary profiles you created in step [“14.a”](#) on page 720:

```
RDEFINE classname profilename FROM(dprofilename) FCLASS(dclassname).
```

- d. Delete the temporary profile:

```
RDELETE dclassname dprofilename
```

15. If your installation has activated the SECLABEL class, RACF performs security label authorization checking. For a complete description, see [“Security label authorization checking”](#) on page 733.



If security label authorization checking succeeds, RACF authorization checking continues with Step “17” on page 721.

16. If the SECLABEL class is *not* active, the SECDATA class is active, and the requested resource has a security level or security category specified, RACF makes two checks in the following described sequence.

- a. RACF compares the security level (SECLEVEL) in the user profile with the security level in the resource profile. If the resource has a higher security level than the user, or if the user has no security level, RACF denies the request.

For a terminal session, RACF uses the lesser of the user's SECLEVEL and the terminal's SECLEVEL when authorizing access to a resource. For example, if the user has a SECLEVEL of 100 and the terminal has a SECLEVEL of 50, RACF uses the terminal's SECLEVEL during authorization checking. Thus, in this case, the user cannot access, through the terminal, any resource with a SECLEVEL greater than 50. (If the terminal is not defined to RACF or is defined without a SECLEVEL, RACF uses the user's SECLEVEL to determine the resources that can be accessed.)

If the security level check passes, authorization checking continues with the following check.

- b. RACF compares the list of security categories in the user profile with the list of security categories in the resource profile. If the resource profile contains a security category that is not in the user's profile, RACF denies the request.

Unlike the security level check, RACF uses *only* the user's security category list for a terminal session.

If both checks succeed, RACF continues authorization checking with Step “17” on page 721.

17. If users attempt to access their own resources, RACF grants the request. For example:

- For tape and DASD data sets, if the user ID of the requesting user is the high-level qualifier of the data set name, RACF grants the request.
- For spool data sets, if the JESSPOOL class is active, RACF compares the user ID and node of the requester with the user ID and node of the creator of the spool data set (using the security token). If the user IDs match, RACF grants the request.

18. RACF checks the user's access authority in the standard access list. If the user is in the list and if the specified access authority is sufficient to allow access, RACF grants the request. If the user is in the list and if the specified access authority is *less than* the requested access, RACF continues processing at Step “23” on page 722 (conditional access list checking). This prevents access based on ID(\*), UACC, or the OPERATIONS attribute.

This could happen if, for example, user JOE requests UPDATE access, and the standard access list includes ID(JOE) ACCESS(READ).

19. RACF determines whether the user has access to the resource because the user is a member of a group and the group is on the standard access list. Which group is used depends on whether list-of-groups processing is in effect. (List-of-groups processing is in effect if the SETROPTS command has been issued with the GRPLIST operand.) RACF determines which group to use according to the following rules:

- If list-of-groups processing is not in effect, RACF uses only the user's current connect group.
- If list-of-groups processing is in effect, RACF finds all of the groups to which the user is connected that are also in the access list. Of these groups, RACF uses the group that has the highest access authority to the resource. (For example, assume that a user is a member of groups A, B, and C. If group A has NONE access authority, group B has READ access authority, and group C has UPDATE access authority, RACF uses group C to determine the user's access.)

If the highest access authority is sufficient to allow the requested access, RACF grants the request. If the highest group that was found in the list does not have the requested authority, RACF continues processing at Step “23” on page 722 (conditional access list checking). This prevents access based on ID(\*), UACC, or the OPERATIONS attribute.

20. If a user ID of \* is found on the standard access list, the current user is defined to RACF without the RESTRICTED attribute, and the access authority granted to \* is:
  - Sufficient to allow the requested access, RACF grants the request.
  - *Not* sufficient to allow the requested access, RACF continues processing at Step [“22” on page 722](#) (OPERATIONS attribute checking).
21. If the universal access authority (UACC) for the resource provides sufficient access authority and the requesting user is not defined with the RESTRICTED attribute, RACF grants the request.
22. If the requesting user has the OPERATIONS attribute (or group-OPERATIONS if the resource is within the scope of that group) and OPERATIONS access is allowed for the class, RACF grants the request.
23. RACF checks the user's access authority in the conditional access list specified with WHEN(TERMINAL), WHEN(CONSOLE), WHEN(APPCPORT), WHEN(JESINPUT) or WHEN(SERVAUTH). If the user is in the list, if the user meets the specified condition (such as logged on at the specified terminal), and if the specified access authority is sufficient to allow access, RACF grants the request. If the user is in the list with insufficient access authority, RACF authorization processing continues at Step [“26” on page 722](#).
24. RACF determines whether the user has access to the resource because the user is a member of a group that meets a condition specified on the conditional access list specified with WHEN(TERMINAL), WHEN(CONSOLE), WHEN(APPCPORT), WHEN(JESINPUT) or WHEN(SERVAUTH). Which group is used depends on whether list-of-groups processing is in effect. (List-of-groups processing is in effect if the SETROPTS command has been issued with the GRPLIST operand). RACF determines which group to use according to the following rules:
  - If list-of-groups processing is not in effect, RACF uses only the user's current connect group.
  - If list-of-groups processing is in effect, RACF finds all of the groups to which the user is connected that are also in the access list. Of these groups, RACF uses the group that has the highest access authority to the resource. (For example, assume that a user is a member of groups A and B. If A has READ access authority and B has UPDATE access authority, RACF uses group B to determine the user's access.)

If the group to be used according to the preceding rules has sufficient access authority to allow the requested access, RACF authorization processing continues at Step [“26” on page 722](#). If none of the user's groups has sufficient authority, RACF does not grant the request, and continues with the next step.
25. If a user ID of \* is found on the conditional access list specified with WHEN(TERMINAL), WHEN(CONSOLE), WHEN(APPCPORT), WHEN(JESINPUT) or WHEN(SERVAUTH), and if the current user is defined to RACF without the RESTRICTED attribute, and if the current user meets the specified condition (such as logged on at the specified terminal), and the access authority granted to \* is sufficient to allow the requested access, RACF grants the request.
26. RACF checks the user's access authority in the conditional access list specified with WHEN(PROGRAM). If the user is in the list, if the user meets the specified condition (such as running the specified program), and if the specified access authority is sufficient to allow access, RACF grants the request.

**Note:** For DASD data sets, if program control is active and a controlled program is executing, RACF performs authorization checking for program access to data sets. If the user/program combination is in the conditional access list with sufficient authority to allow access to the data sets, RACF grants the request. If the user/program combination is in the conditional access list with insufficient authority, RACF denies the request. For more information, see [“Authorization checking for access control to data sets” on page 320](#).
27. RACF determines whether the user has access to the resource because the user is a member of a group that meets a condition specified on the conditional access list (such as running a specified program). Which group is used depends on whether list-of-groups processing is in effect. (List-of-groups processing is in effect if the SETROPTS command has been issued with the GRPLIST operand.) RACF determines which group to use according to the following rules:
  - If list-of-groups processing is not in effect, RACF uses only the user's current connect group.

- If list-of-groups processing is in effect, RACF finds all of the groups to which the user is connected that are also in the access list. Of these groups, RACF uses the group that has the highest access authority to the resource. (For example, assume that a user is a member of groups A and B. If A has READ access authority and B has UPDATE access authority, RACF uses group B to determine the user's access.)

If the group to be used according to the preceding rules has sufficient access authority to allow the requested access, RACF grants the request. If the group is specified in the list with insufficient access authority, RACF denies the request.

28. If a user ID of \* is found on the conditional access list specified with WHEN(PROGRAM), and if the current user is defined to RACF without the RESTRICTED attribute, and if the current user meets the specified condition (such as logged on at the specified terminal or running the specified program), and the access authority granted to \* is sufficient to allow the requested access, RACF grants the request.
29. If the WARNING flag is ON in the profile (set using the WARNING operand on the ADDSD, ALTDSD, RDEFINE, or RALTER command), RACF grants the request.
30. If SETROPTS CATDSNS(FAILURES) is in effect, RACF denies the request *unless* at least one of the following is true:
  - The data set is newly created in this job, or is a system temporary data set.
  - The data set is on tape, and the request is for UPDATE access.
  - The data set is protected by a discrete profile.
  - The data set is cataloged in the master catalog.
  - The user has access to FACILITY resource ICHUNCAT.*data-set-name* (truncated to 39 characters, if necessary).
  - The user has the SPECIAL attribute.

**Note:** If the user gains access through having the SPECIAL attribute and none of the prior conditions were true, RACF issues a warning message and creates an SMF record as though CATDSNS(WARNING) were in effect.
31. The postprocessing exit (ICHRCX02) can grant or deny the request.
32. For the DATASET class, if no profile is found and the SETROPTS PROTECTALL(FAILURES) option is in effect, RACF denies the request. If no profile is found and the SETROPTS PROTECTALL(WARNING) option is in effect, RACF grants the request.

## Pictorial view of RACF authorization checking

For a complete description of the numbered steps described in the following figures, see [“Authorizing access to RACF-protected resources”](#) on page 719.

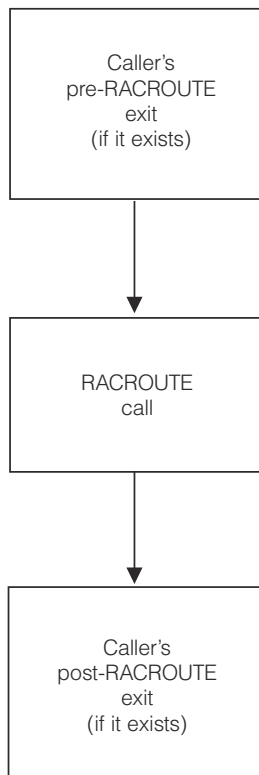


Figure 64. Process flow of callers of RACF for RACROUTE REQUEST=AUTH requests

**Note:** For information about exits that affect RACROUTE calls, see the documentation for the calling product.

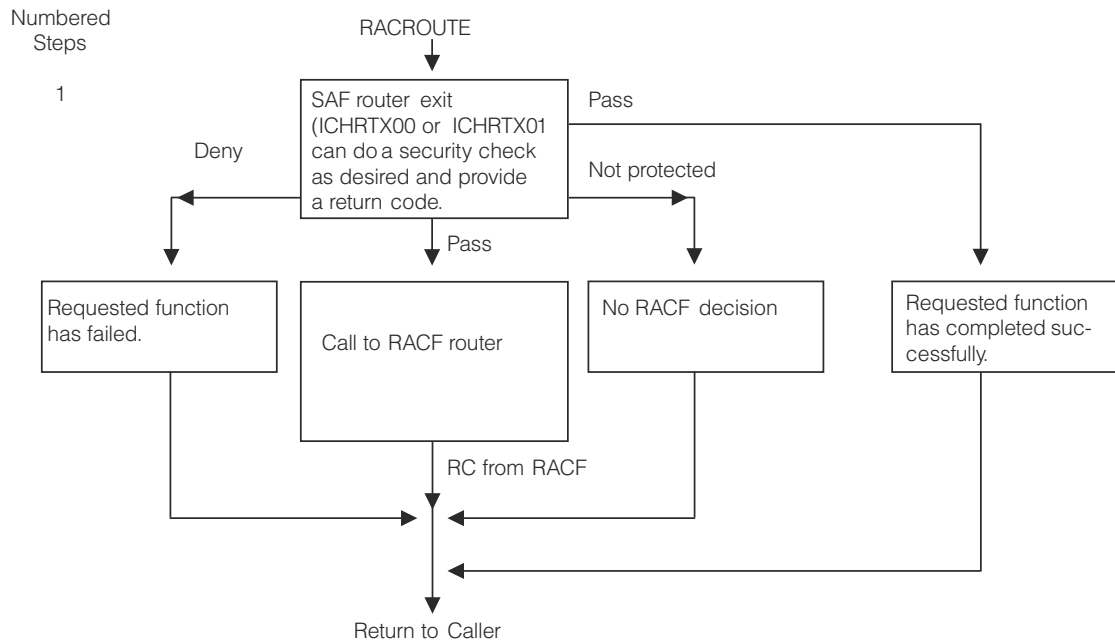


Figure 65. Process flow of SAF router for RACROUTE REQUEST=AUTH requests

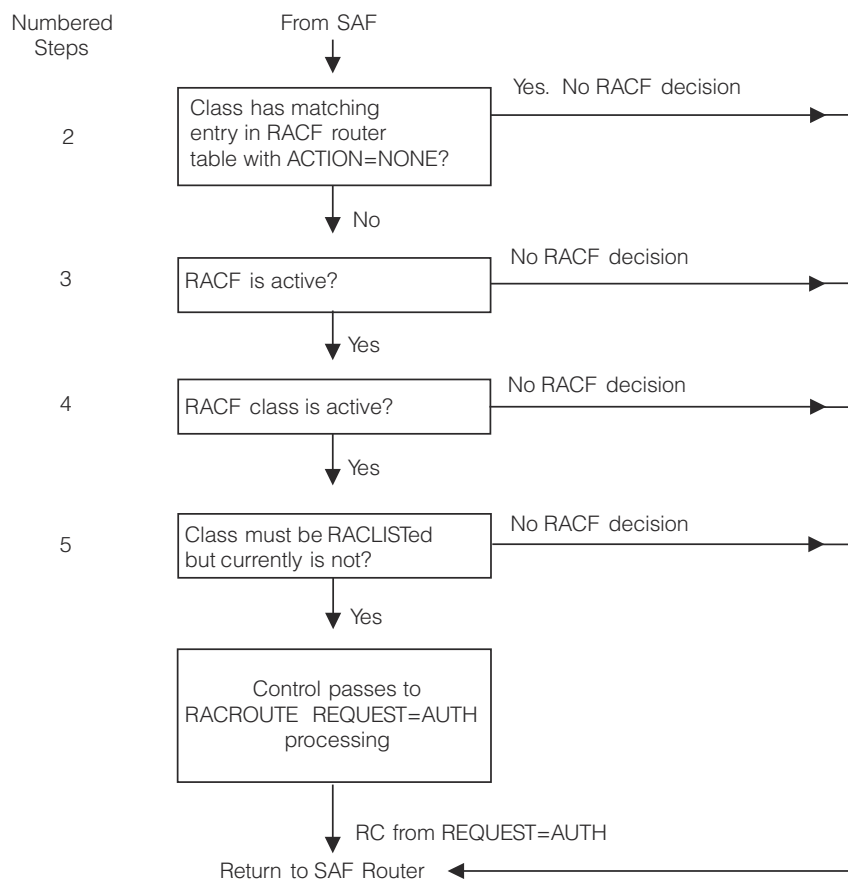


Figure 66. Process flow of RACF router

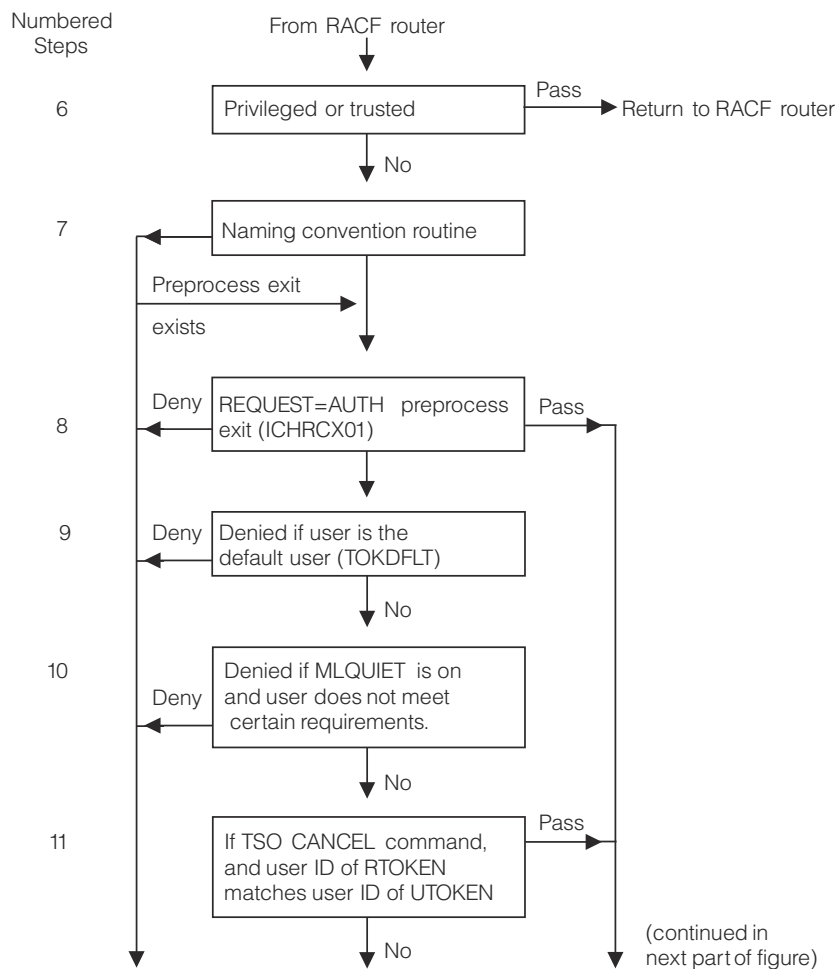


Figure 67. Process flow of RACF authorization checking, part 1 of 3

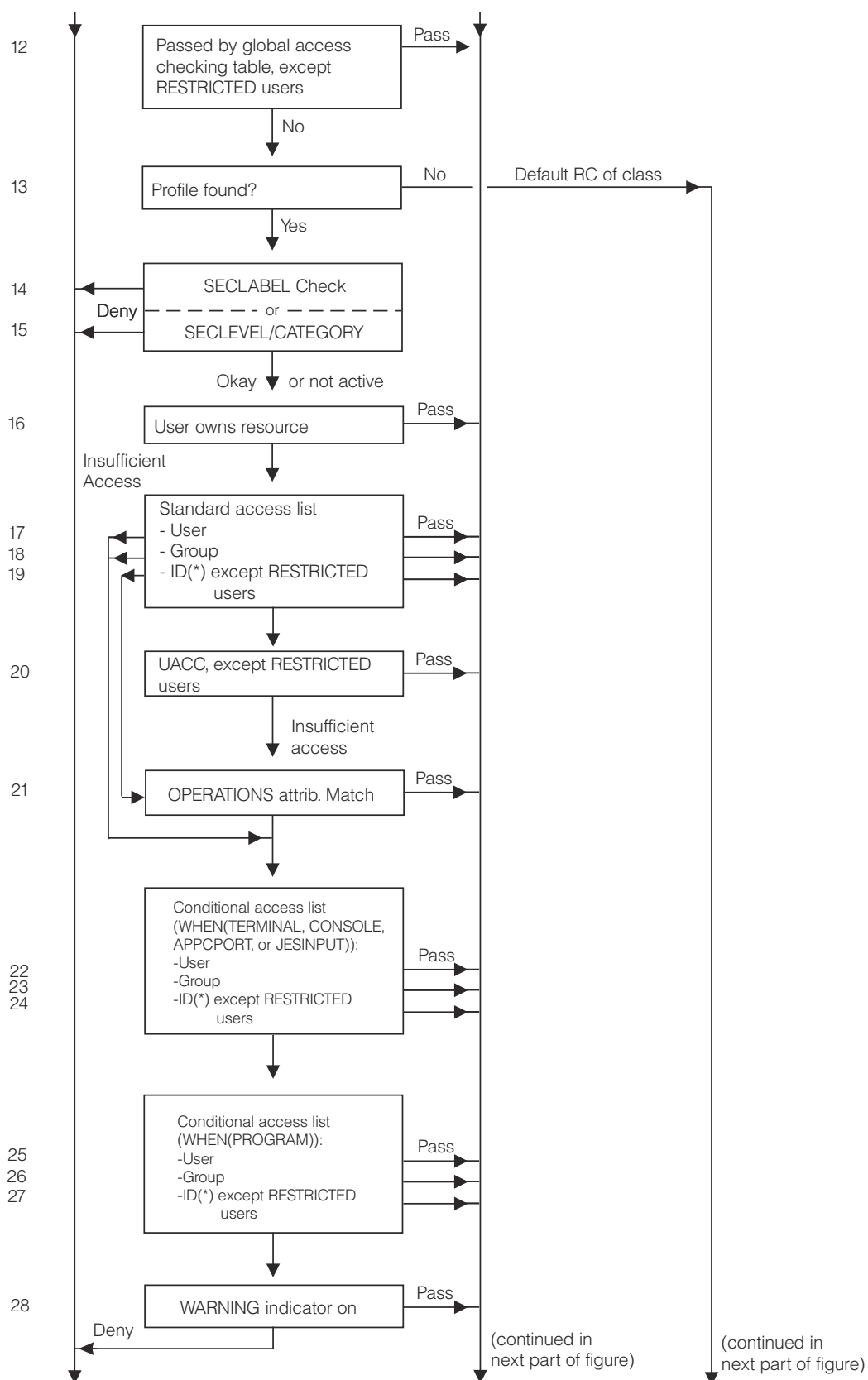


Figure 68. Process flow of RACF authorization checking, part 2 of 3

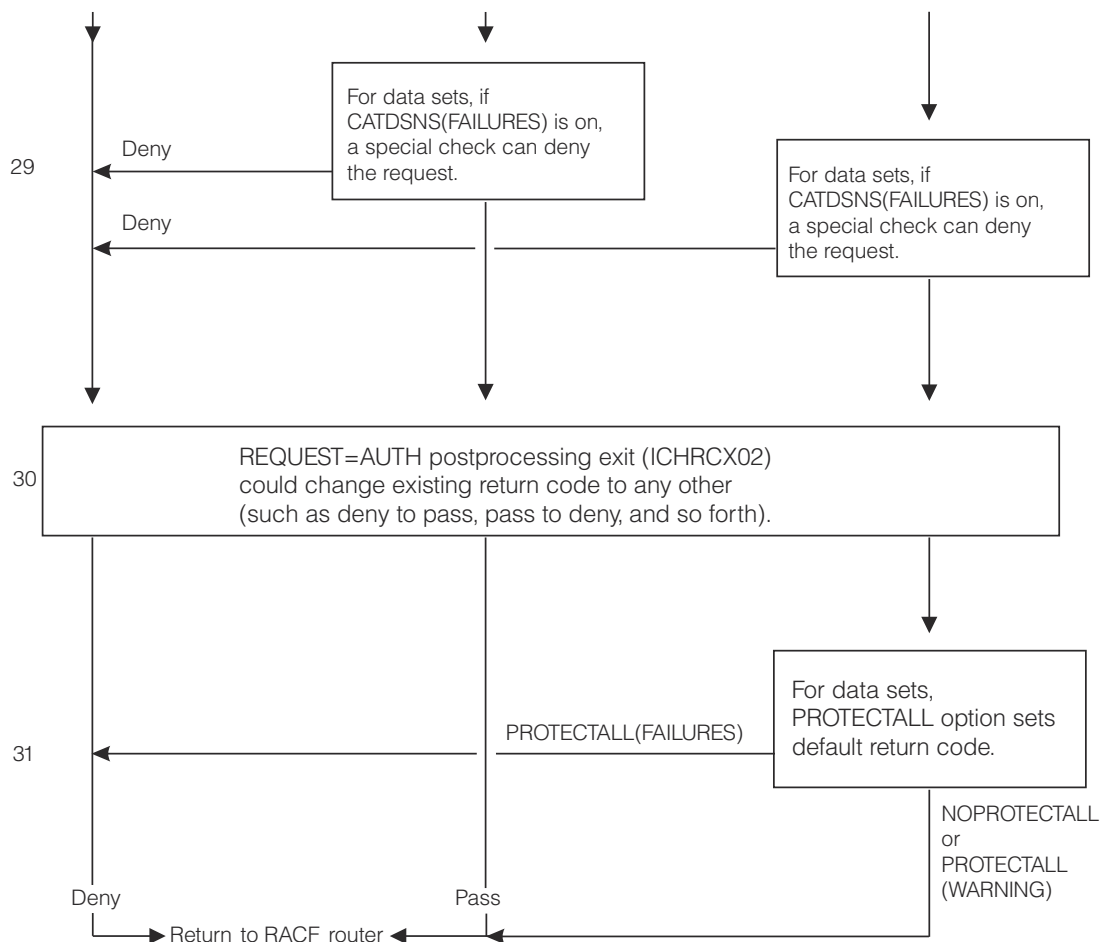


Figure 69. Process flow of RACF authorization checking, part 3 of 3

## Authorizing access to z/OS UNIX files and directories

To perform authorization checking for z/OS UNIX files and directories, RACF makes the following checks. RACF stops processing when the request is granted or denied.

### Notes:

- The effective UID and effective GID of the process is used in determining access decisions. The only exception is that when CREDFUNCTION is AFC\_ACCESS, the real UID and real GID are used. In other words, if file access is being tested, rather than requested, the real UID and GID are used instead of the effective UID and GID. The real and effective IDs are generally the same for a process, but if a set-uid or set-gid program is executed, they can be different.
- If the requesting user is represented by an unauthenticated client ACEE, then Steps “2” on page 728–“29” on page 730 are performed first for the client, and then, if successful, for the server. Both client and server must have access to the file in order for the request to succeed.
  - The SAF callable services router exit (IRRSXT00) can grant or deny access before RACF authorization processing occurs.
  - If the system (kernel) is the caller, then access is failed if either of the following conditions occurs:
    - The request includes execute authority for a file and execute authority cannot be granted. In this condition, none of the permissions bits grant execute access, and, if an ACL is present and the FSSEC class is active, no ACL entry grants execute access.
    - Security label authorization checking fails. In this condition, the SECLABEL class is active, the object being accessed is a directory, the directory's SECLABEL is not SYSMULTI, and the CRED contains a SECLABEL.



Otherwise, access is granted.

3. RACF checks for a profile in the FSACCESS class that covers the file system name when all of the following conditions are met:

- The user does not have the AUDITOR or ROAUDIT attribute.
- A file system name was specified in the CRED.
- The FSACCESS class is active and RACLISTed.

If a matching profile is found and the user does not have at least UPDATE authority, access is denied. Otherwise, access checking continues.

4. RACF checks for a profile in the FSEXEC class that covers the file system name when all of the following conditions are met:

- The request is for file execution access.
- A file system name was specified in the CRED.
- The FSEXEC class is active and RACLISTed.

If a matching profile is found and the user does not have at least UPDATE authority, access is denied. Otherwise, access checking continues.

5. If the SECLABEL class is not active, then go to Step “16” on page 729.
6. If the user has the TRUSTED or PRIVILEGED attribute, then access is granted automatically unless the user is executing a file. If the user is executing a file, access is denied only if none of the permissions bits grant execute access, and, if an ACL is present and the FSSEC class is active, no ACL entry grants execute access. Otherwise, access is granted.
7. If the user has the RACF AUDITOR or ROAUDIT attribute, and has read or search access for a directory is requested, access is granted.
8. If SETROPTS MLFSOBJ is active and the file does not have a security label, the request is failed.
9. If SETROPTS MLS is active (either in WARNING or FAILURES mode) and all of the following conditions occur, the request is failed.
  - The user has a security label.
  - The file has no security label.
  - The user explicitly requested write access but is not in writedown mode.

**Note:** The SETROPTS MLS(WARNING) option is not supported for UNIX files and directories, and it is treated the same as MLS(FAILURES).

10. If the file has a security label but the user does not, then the request is failed.
11. If the user's security label is equivalent to the security label of the file (this condition is also satisfied if either security label is SYSMULTI), then continue at Step “17” on page 729.
12. If ANY access is requested, then two security label dominance checks (RACROUTE REQUEST=DIRAUTH) are performed: one for READ and one for WRITE. If either succeeds, then continue at Step “17” on page 729. Otherwise, the request is failed.
13. If the user is requesting write access along with read or search/execute access, then a READWRITE dominance check is performed. If it succeeds, then continue at Step “17” on page 729. Otherwise, the request is failed.
14. If the user is requesting only read or search/execute access, then a READ dominance check is performed. If it succeeds, then continue at Step “17” on page 729. Otherwise, the request is failed.
15. If the user is requesting only write access, then a WRITE dominance check is performed. If it succeeds, then continue at Step “17” on page 729. Otherwise, the request is failed.
16. If the user has the RACF AUDITOR or ROAUDIT attribute, and read or search access for a directory is requested, access is granted.
17. If the user has UID(0), then access is granted automatically unless the user is executing a file. If the user is executing a file, access is denied only if none of the permissions bits grant execute

access, and, if an ACL is present and the FSSEC class is active, no ACL entry grants execute access. Otherwise, access is granted.

18. If the UID matches the file owner UID, the file's "owner" permission bits are checked. If the "owner" bits allow the requested access, then access is granted. Otherwise, go to Step "28" on page 730.
19. If the FSSEC class is active, and an ACL exists, and there is an ACL entry for the requesting UID, then the permission bits of that ACL entry are checked. If the ACL entry allows the requested access, then access is granted. Otherwise, go to Step "27" on page 730.
20. If the GID matches the file owner GID, the file's "group" permission bits are checked. If the "group" bits allow the requested access, then access is granted.
21. If the FSSEC class is active, and an ACL exists, and there is an ACL entry for the requesting GID, then the permission bits of that ACL entry are checked. If the ACL entry allows the requested access, then access is granted. If not, then the next ACL entry is checked until there are no more entries.
22. If any of the user's supplemental GIDs match the file owner GID, the file's "group" permission bits are checked. If the "group" bits allow the requested access, then access is granted.
23. If the FSSEC class is active, and an ACL exists, and there is an ACL entry for any of the user's supplemental GIDs, then the permission bits of that ACL entry are checked. If the ACL entry allows the requested access, then access is granted. If not, then the next ACL entry is checked until there are no more entries.
24. If at least one matching ACL entry was found for the GID, or any of the supplemental GIDs, then processing continues with Step "27" on page 730. If the GID, or any of the supplemental GIDs, matched the file owner GID, then processing continues with Step "28" on page 730. Otherwise (neither the GID nor any of the supplemental GIDs matched either the file owner GID or an ACL entry), processing continues with the next step.
25. If the requesting user has the RESTRICTED attribute, and the UNIXPRIV class is active and RACLISed, and the RESTRICTED.FILESYS.ACCESS resource is protected by a profile in the UNIXPRIV class, and the user does not have at least READ access, then go to Step "28" on page 730.
26. The file's "other" permission bits are checked. If the "other" bits allow the requested access, then access is granted. Otherwise, go to Step "28" on page 730.
27. If the UNIXPRIV class is active and RACLISed, and if the SUPERUSER.FILESYS.ACLOVERRIDE resource is protected by a profile in the UNIXPRIV class, then the user must have the correct access level as documented for the ck\_access (IRRSKA00) callable service in *z/OS Security Server RACF Callable Services*. If the profile exists, it determines whether file access is granted or denied.
28. If the request is for directory read or search, the UNIXPRIV class is active and RACLISed, and the user has at least read permission to the SUPERUSER.FILESYS.DIRSRCH resource, then access is granted. Otherwise, if the UNIXPRIV class is active and RACLISed, and if the SUPERUSER.FILESYS resource is protected by a profile in the UNIXPRIV class, then the user must have the correct access level as documented for the ck\_access (IRRSKA00) callable service in *z/OS Security Server RACF Callable Services*. If the profile exists, it determines whether file access is granted or denied.
29. Access is denied.
30. The SAF callable services router exit (IRRSXT00) can grant or deny access after RACF authorization processing occurs.

## Authorizing access to RACF-protected terminals

When a RACF-defined user logs on to TSO or signs on to IMS or CICS using a terminal protected by a profile in the TERMINAL or GTERMINL class and the TERMINAL class is active, RACF performs authorization checking to verify that the user is permitted use of the terminal. RACF performs this authorization checking during REQUEST=VERIFY processing at the same time as it performs user identification and verification.

RACF performs terminal authorization checking in the following sequence:

1. If your installation has activated the SECLABEL class, RACF performs security label authorization checking. For a complete description, see [“Security label authorization checking”](#) on page 733. If security label authorization checking succeeds, RACF authorization checking continues with the next step.
2. If the requesting user has at least READ access authority to the terminal, RACF processing continues at Step [“5”](#) on page 731. If the user's access authority is NONE, RACF denies use of the terminal and stops terminal authorization checking.
3. If the requesting user's current connect group (or, if you activate list-of-groups checking, one of the user's other connect groups) has at least READ access authority to the terminal, RACF processing continues at Step [“5”](#) on page 731. If the group's access authority is NONE, RACF denies use of the terminal and stops terminal authorization checking.
4. If the profile has a universal access authority (UACC) of at least READ and your installation has not specified NOTERMUACC for the user's current connect group, RACF processing continues at Step [“5”](#) on page 731. Otherwise, RACF denies use of the terminal and stops terminal authorization checking.

**Note:** For defined terminals, you can specify the universal access authority (UACC) with the RDEFINE or RALTER command. For undefined terminals, you can specify the universal access authority with the TERMUACC operand of the SETROPTS command.

For more information, see [“Limiting specific groups of users to specific terminals”](#) on page 227.

5. If your installation authorizes the use of the terminal on this particular day and time, RACF grants access to the terminal. (You can specify the terminal time and day-of-week restrictions with the RDEFINE and RALTER commands.) RACF also checks whether your installation has authorized the user to access the system on this particular day and time. (You can specify the user time and day-of-week restrictions with the ADDUSER and ALTUSER commands.)

**Note:**

1. The REQUEST=AUTH and REQUEST=VERIFY preprocessing and postprocessing exit routines are available during terminal authorization checking.
2. Global access checking is not available during terminal authorization checking performed by REQUEST=VERIFY.
3. Profiles in the GTERMINL class are ignored unless SETROPTS RACLIST processing is in effect.

## Authorizing access to consoles, JES input devices, APPC partner LUs, or IP addresses

When a RACF-defined user logs on to a JES or MCS console, submits a batch job from a JES input device, submits an APPC request from a partner LU, or accesses the network through an IP address, RACF can perform authorization checking to verify that the user is permitted use of the RACF-protected console, JES input device, partner LU, or IP address. RACF performs this authorization checking during REQUEST=VERIFY processing at the same time as it performs user identification and verification. For RACF to perform this authorization checking, your installation must activate the appropriate class, as follows:

- For JES or MCS consoles, activate the CONSOLE class.
- For JES input devices, activate the JESINPUT class.
- For APPC partner LUs, activate the APPCPORT class.
- For IP addresses, activate the SERVAUTH class.

The resource must be protected by a profile in the appropriate class. If no profile exists for the resource, RACF fails the request.

**Note:**

1. If the resource is a terminal, console, partner LU, JES writer, or IP address, RACF compares the security level of the user with the security level of the resource. If the resource has a higher security level than the user, RACF denies the request. For a terminal session, the security level that RACF uses

for the user is the lower of the user's SECLEVEL and the terminal's SECLEVEL. Thus, if the terminal has a SECLEVEL of 50 and the user has a SECLEVEL of 100, the user cannot access through that terminal any data that has a SECLEVEL of over 50.

2. RACF compares the list of security categories in the user's profile with the list of security categories in the resource's profile. If RACF finds any security category in the resource profile that is not in the user's profile, RACF denies the request. If RACF does not deny the request, RACF continues with authorization processing. If there are no categories in the resource profile, RACF continues with authorization processing.

The rest of this topic uses the term *device authorization checking* to refer to the authorization checking done for any of the above resources.

RACF performs device authorization checking in the following sequence:

1. If your installation has activated the SECLABEL class, RACF performs security label authorization checking. For a complete description, see [“Security label authorization checking” on page 733](#). If security label authorization checking succeeds, RACF authorization checking continues with the next step.
2. If the requesting user has at least READ access authority to the device, RACF grants the request with no further processing. If the user's access authority is NONE, RACF denies use of the device and stops device authorization checking. If the requesting user is not in the access list, device authorization checking continues with the next step.
3. If the requesting user's current connect group (or, if you activate list-of-groups checking, one of the user's other connect groups) has at least READ access authority to the device, RACF grants the request with no further processing. If the group's access authority is NONE, RACF denies use of the device and stops device authorization checking. If the group is not in the access list, device authorization checking continues with the next step.
4. If the profile has a universal access authority (UACC) of at least READ, RACF grants the request with no further processing. Otherwise, RACF denies use of the device and stops device authorization checking.

**Note:**

- a. The TERMUACC operand of the SETROPTS command has no effect on consoles or JES input devices.
- b. You cannot specify time or day-of-week restrictions for consoles or JES input devices. (You can specify user time and day-of-week restrictions with the ADDUSER and ALTUSER commands.)

**Note:**

1. The REQUEST=AUTH and REQUEST=VERIFY preprocessing and postprocessing exit routines are available during device authorization checking.
2. Global access checking is not available during device authorization checking performed by REQUEST=VERIFY.

## Authorization checking for RACROUTE REQUEST=FASTAUTH requests

Some resource managers, such as CICS, have high performance requirements. In order to do resource authorization checking with RACF, they use RACF facilities to load all of the profiles for a given class into the user's storage area or into a common storage area called a data space. The resource managers can do a *fast* authorization check against profiles in the user's storage, or profiles in the data space, or both.

Fast authorization checking is different from normal authorization checking as follows:

- The global access checking table is not used.
- Security labels are only used for READ and READWRITE mandatory access checking (MAC) requests.
- When a user has a security label but the accessed resource does not, mandatory access checking is bypassed, and only discretionary access checking is done to grant or deny access to the resource, even when SETROPTS MLS is in effect.
- WHEN(PROGRAM) conditional access checking is done for SERVAUTH class resources.

- If reverification is required for an IMS transaction, the user must also enter the SIGN ON password with the transaction request.
- Authorization checking for nested ACEEs and access to delegated resources is processed using RACROUTE REQUEST=FASTAUTH.
- WHEN(CRITERIA) conditional access checking is done.
- When the caller specifies the AUTHCHKS=CRITONLY keyword and provides a valid CRITERIA, only the following subset of authorization checks are performed:
  - Enforcement of the rules for SETROPTS MLQUIET, when SETROPTS MLQUIET is in effect.
  - Security label authorization checks, when the SECLABEL class is active.
  - Security level and security category authorization checks, when the SECLABEL class is not active and the SECDATA class is active.
  - Search of the conditional access list for a matching criteria as specified by the CRITERIA keyword.

For additional information about the following topics, see the resources listed:

- Processing RACLISTed profiles:
  - See *z/OS Security Server RACF System Programmer's Guide*.
- Using RACROUTE REQUEST=LIST to place profiles in storage:
  - See *z/OS Security Server RACROUTE Macro Reference*.
- Using RACF to provide security for CICS:
  - Visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).
- Nested ACEEs and delegated resources:
  - See [“Defining delegated resources” on page 257](#).

## Authorizing access to RACF-protected applications

You can control access to some applications (for example, IMS and CICS regions) by defining them to RACF as resources in the APPL class. When the user attempts to sign on the application, the application uses RACF to verify the user's identity and his authority to use that application. RACF does an authorization check to determine the user's authorization to the application.

- If there is a matching profile in the APPL class, RACF performs normal authorization checking as described in [“Authorizing access to RACF-protected resources” on page 719](#).
- If there is no matching profile in the APPL class, RACF allows the user to access the application.

### Note:

1. The REQUEST=AUTH and REQUEST=VERIFY preprocessing and postprocessing exit routines are available during application authorization checking.
2. Global access checking is not available during application authorization checking performed by REQUEST=VERIFY.
3. For information about using IMS with RACF, visit [IMS in IBM Documentation \(www.ibm.com/docs/en/ims\)](http://www.ibm.com/docs/en/ims).
4. For information about using CICS with RACF, visit [CICS Transaction Server for z/OS \(www.ibm.com/docs/en/cics-ts\)](http://www.ibm.com/docs/en/cics-ts).

## Security label authorization checking

This sequence of authorization checks begins in one of the sequences in [“Authorization checking for RACF-protected resources” on page 718](#) and continues the RACROUTE REQUEST=AUTH authorization checking service that is begun there.

When the SECLABEL class is active on your system, and a user or job requests access to a resource, RACF compares the security label of the user with the security label of the resource. For a general description of these comparisons, see [“Comparing security labels” on page 98](#).

1. If the user requesting access does not have a security label and the resource does have a security label, RACF fails the request.
2. If the SETROPTS MLACTIVE(FAILURES) option is in effect and the resource does not have a security label associated with it, and the resource class is DATASET or another class that requires security labels as defined in the class descriptor table (CDT), RACF fails the request.
3. If the SETROPTS MLACTIVE(WARNING) option is in effect, RACF makes the same checks as in Step “2” on page 734. If the access check fails because the resource does not have a security label, RACF issues a warning message and grants the request.
4. If the SETROPTS MLS(FAILURES) option is in effect, RACF makes the tests shown in [Table 54 on page 735](#) and fails the request if the test fails.

If the SETROPTS COMPATMODE option is in effect, RACF checks to see if the user's UTOKEN indicates that the ACEE was created with an older protocol (pre-RACF 1.9.0). If both are true, then RACF checks to see if the user has access to a security label that could allow the requested access to the resource.

- If the user has no access to any such security label, RACF fails the request.
  - If the user does have access to such a security label, RACF continues authorization checking and logs the request.
5. If the SETROPTS MLS(WARNING) option is in effect for this resource class, RACF makes the same checks as in Step “4” on page 734. If any test fails the request, RACF issues a warning message and grants the request.
  6. If the SETROPTS NOMLS option is in effect, RACF makes the tests shown in [Table 55 on page 736](#) and fails the request if the test fails.

If the SETROPTS COMPATMODE option is in effect, RACF checks to see if the user's UTOKEN indicates that the ACEE was created with an older protocol (pre-RACF 1.9.0). If both are true, then RACF checks to see if the user has access to a security label that could allow the requested access to the resource.

- If the user has no access to any such security label, RACF fails the request.
- If the user does have access to such a security label, RACF continues authorization checking and logs the request.

If the resource is a JES spool data set, RACF uses the security label in the token associated with the data set (specified on the RTOKEN parameter of the RACROUTE REQUEST=AUTH macro). Otherwise, RACF uses the security label kept in the resource profile protecting the resource, in the FSP for files, or in the ISP for IPC objects.

## Types of security label authorization checking

When the SECLABEL class is active on your system, RACF authorization checking uses mandatory access control (MAC), in addition to discretionary access controls (DAC). (See [“Characteristics of a multilevel-secure environment” on page 8](#).) There are three types of MAC for security label authorization checking. The type of checking used for a particular resource depends on how the resource class is defined in the class descriptor table (CDT).

Table 53 on page 735 lists the mandatory access control (MAC) authorization types defined in the class descriptor table (CDT), and specifies the relationship that must exist between the user's security label and the security label of the resource in order for the security label authorization to be successful.



Table 53. Required relationship between security levels for each MAC checking type

Type of MAC checking	Relationship between security labels
normal MAC	The security label of the <i>user</i> must dominate the security label of the resource in order for the user to be granted access to the resource. This is the <i>default</i> attribute in the class descriptor table and is in effect when neither RVRSMAC nor EQUALMAC is defined as an attribute.
reverse MAC	The security label of the <i>resource</i> must dominate the security label of the user in order for the user to be granted access to the resource. This is defined by the RVRSMAC attribute in the class descriptor table.
equal MAC	The security label of the user and the security label of the resource must be <i>equivalent</i> in order for the user to be granted access to the resource. This is defined by the EQUALMAC attribute in the class descriptor table.

The MAC authorization types described in Table 53 on page 735 are used for the following levels of authorization request for resource access. (See “Security labels” on page 96 for descriptions and examples of each requested access level.)

- Read-only
- Read-write
- Write-only

The security label relationships for each MAC authorization type are applied differently depending on the setting of the SETROPTS MLS option. See “Authorization summary for SETROPTS MLS(FAILURES) and MLS(WARNINGS)” on page 735 and “Authorization summary for SETROPTS NOMLS” on page 736.

### Authorization summary for SETROPTS MLS(FAILURES) and MLS(WARNINGS)

Table 54 on page 735 shows the required relationship that must exist between the user's security label and the security label of the resource in order for user to gain access to the resource while the SECLABEL class is active and SETROPTS MLS(FAILURES) or MLS(WARNING) is in effect, based on the type of MAC checking and the requested access level.

When SETROPTS MLS is in effect and a user has a security label but the resource does not, the user will fail to gain access to the resource because the authorization checking is done using RACROUTE REQUEST=AUTH. The user will be successful in gaining access when the authorization check is done using RACROUTE REQUEST=FASTAUTH, even when SETROPTS MLS is in effect.

Table 54. Security label authorization checking when SECLABEL class is active and either SETROPTS MLS(FAILURES) or MLS(WARNING) is in effect

Requested access	Normal MAC	Reverse MAC	Equal MAC
<b>Read-only</b>	User dominant	Resource dominant	Equivalent
<b>Read-write</b>	Equivalent	Equivalent	Equivalent
<b>Write-only</b> <sup>1</sup>	Resource dominant <sup>2</sup>	Unpredictable <sup>3</sup>	Equivalent

#### Notes:

- <sup>1</sup> z/OS does not support write-only requests for data sets or tape volumes. All write-only requests are tested as both read-only and write-only requests. Therefore, the security labels must be equivalent.
- <sup>2</sup> Users cannot write to a resource that has a lesser security label than the user's current security label. This inability to *writedown* is enforced when SETROPTS MLS(FAILURES) is in effect to ensure that a user does not declassify data.
- <sup>3</sup> The test for write-only is not supported for classes defined with the reverse MAC attribute.

## Authorization summary for SETROPTS NOMLS

Table 55 on page 736 shows the required relationship that must exist between the user's security label and the security label of the resource in order for user to gain access to the resource while the SECLABEL class is active and SETROPTS NOMLS is in effect, or the user is in *writedown* mode, based on the type of MAC checking and the requested access level.

*Table 55. Security label authorization checking when SECLABEL class is active and either SETROPTS NOMLS is in effect or the user is in "writedown" mode.*

Requested access	Normal MAC	Reverse MAC	Equal MAC
<b>Read-only</b>	User dominant <sup>2</sup>	Resource dominant	Equivalent
<b>Read-write</b>	User dominant <sup>2</sup>	Resource dominant	Equivalent
<b>Write-only</b> <sup>1</sup>	User dominant or resource dominant	Unpredictable <sup>3</sup>	Equivalent

### Notes:

- <sup>1</sup> z/OS does not support write-only requests for data sets or tape volumes. All write-only requests are tested as both read-only and write-only requests. Therefore, the security labels must be equivalent.
- <sup>2</sup> If SETROPTS MLS(WARNING) is active instead of NOMLS in these cases, an ICH408I warning message is written to the security console.
- <sup>3</sup> The test for write-only is not supported for classes defined with the reverse MAC attribute.

## Authorization summary for SETROPTS MLACTIVE

Table 56 on page 737 describes the results of security label authorization when the SECLABEL class is active and either the user's or resource's security label is missing. The results vary depending on the SETROPTS MLACTIVE setting and whether or not the resource class being checked requires security labels. The supplied class descriptor table (ICHRRCDX) specifies which resource classes require security labels. For a listing of the supplied class descriptor table (CDT) entries, see the [z/OS Security Server RACF Macros and Interfaces](#).

**Attention:** Do not issue the SETROPTS MLACTIVE(FAILURES) command unless you have assigned appropriate security labels to users and to the resources they must access. To recover from such a situation, logon as a user with the SPECIAL attribute, specifying SYSHIGH as the current security label. Then, either assign security labels or issue SETROPTS NOMLACTIVE. If you turn on MLACTIVE and do not correctly define all profiles that need SECLABELs, IPL failures, or other serious problems can occur.

### Guidelines:

- Back up your RACF database with a database that you know you can use to IPL.
- Define new system profiles (including classes such as DATASET, TERMINAL, TAPEVOL, APPL or any other active class that has SLBLREQ=YES in the class descriptor table) and ensure they have the correct security labels.
- Turn MLACTIVE on in WARNING mode.
- Watch out for relevant warning messages.

**Data set and general resource profiles in WARNING mode:** A user or task *can* access a resource that is in WARNING mode and has no security label even when MLACTIVE(FAILURES) is in effect and the class requires security labels. The user or task receives a warning message and gains access. (A data set or general resource is in WARNING mode when you define or modify the profile that protects it and you specify the WARNING operand.)



Table 56. Effects of MLACTIVE settings on security label authorization

Environment	Missing user security label (resource security label is present)	Missing resource security label (user security label is present)	Missing both user and resource security labels
MLACTIVE(FAILURES) and resource class requires security labels	Fail <sup>1</sup>	Fail	Fail <sup>1</sup>
MLACTIVE(WARNING) and resource class requires security labels	Fail	Pass and warning message sent to security console	Pass and warning message sent to security console
NOMLACTIVE and resource class requires security labels	Fail	Pass	Pass
MLACTIVE(FAILURES) and resource class does not require security labels	Fail <sup>1</sup>	Pass	Pass <sup>1</sup>
MLACTIVE(WARNING) and resource class does not require security labels	Fail	Pass	Pass
NOMLACTIVE and resource class does not require security labels	Fail	Pass	Pass

**Note:** <sup>1</sup> In these cases, the user has a missing security label while SETROPTS MLACTIVE(FAILURES) is in effect because the user logged in without a security label before SETROPTS MLACTIVE(FAILURES) was activated. Authorization requests are passed or failed according to the entries in [Table 56 on page 737](#). If such a user attempts to log on to the system while SETROPTS MLACTIVE(FAILURES) was in effect, the user is not allowed to log on unless the user has access to the SYSLOW security label. Users who have access to SYSLOW at logon time when MLACTIVE(FAILURES) is active will be assigned and run with SYSLOW.

## Special access rule for SPECIAL users

If SETROPTS MLACTIVE(FAILURES) and SETROPTS MLS(FAILURES) are active, a SPECIAL user logged on with the security label SYSHIGH is allowed to access resources as if SETROPTS MLS(WARNING) and MLACTIVE(WARNING) were both in effect. In this case, RACF issues warning messages instead of failing the requests. If SETROPTS MLACTIVE(FAILURES) and SETROPTS MLS(FAILURES) have been turned on without sufficient preparation, such as without assigning security labels to resources, this special access rule enables the security administrator to access resources on the system.

**Restriction:** To prevent inadvertent declassifications or *writedowns*, read-only access to resources is allowed with a warning message, while write accesses fail. For more information about authorizing users with the writedown privilege, see [“Controlling the write-down privilege” on page 101](#).

## Relationships among the SECLABEL class, SETROPTS MLS(FAILURES), SETROPTS MLACTIVE(FAILURES) and SETROPTS MLQUIET

Table 57 on page 737 shows the relationships of the SECLABEL class and the SETROPTS MLS, MLACTIVE(FAILURES) and MLQUIET options.

Table 57. Relationships among the SECLABEL class, SETROPTS MLS(FAILURES), SETROPTS MLACTIVE(FAILURES), and SETROPTS MLQUIET

SECLABEL class	MLS (FAILURES)	MLACTIVE (FAILURES)	MLQUIET	Effect
Inactive	Off	Off	Off	Security labels have no effect on authorization checking.

Table 57. Relationships among the SECLABEL class, SETROPTS MLS(FAILURES), SETROPTS MLACTIVE(FAILURES), and SETROPTS MLQUIET (continued)

SECLABEL class	MLS (FAILURES)	MLACTIVE (FAILURES)	MLQUIET	Effect
Active	Off	Off	Off	RACF uses security labels and allows writing to a lower security label.
Active	On	Off	Off	RACF uses security labels and prevents writing to a lower security label ("no write down").
Active	On	On	Off	All resources must be labeled, RACF uses security labels, and RACF prevents writing to a lower security label.
Active	Off	On	Off	Those resources required to have security labels by definition in the class descriptor table (CDT), resources in the DATASET class, and users must have security labels.
Active	Either	Either	On *	All attempts to access the system or resources fail (unless the attempt is made by the trusted computing base, a security administrator, or a console operator).

**Note:** \* To activate SETROPTS MLQUIET, you must also enable SETROPTS MLSTABLE.

## Problems with user ID authentication

This topic includes the following information:

- [“When logon or job initialization processing takes place and why” on page 738](#)
- [“Logon/job initialization processing” on page 739](#)

### When logon or job initialization processing takes place and why

When a user requests access to the system, the application controlling the user's access can issue the RACROUTE macro with REQUEST=VERIFY or REQUEST=VERIFYX specified (or the RACINIT macro). For ease of reference, this topic calls any such a request a verify request, and the program issuing the request is called an application.

Some of the places verify requests occur are:

- When interactive users log on (through TSO)
- When batch jobs are submitted through JES
- When NJE jobs or SYSOUT are received
- When APPC/MVS allocation requests are received
- When CICS, IMS, or NetView/Access Services allow users to sign on
- When other APF-authorized applications allow users to access the system.

Based on the specifications on the verify request, RACF determines whether the requesting user is authorized to enter the system.

- If the user is authorized to enter the system, RACF returns a "successful" return code (return code 0) to the application. The application then allows the request to complete.
- If the user is not authorized to enter the system, RACF returns an "unauthorized" return code (other than 0) to the application. In general, the application then fails the request.

**Note:**

1. The REQUEST=VERIFY and REQUEST=VERIFYX preprocessing and postprocessing exit routines are available during verification processing.
2. RACF authorization checks can be requested by RACF or the application (for example, to determine if a user is authorized to use a particular terminal). REQUEST=AUTH preprocessing and postprocessing exits are available during this authorization processing.
3. SMF log records or messages can be generated. (Failures are always recorded. Successes can be recorded if the application requests it on the REQUEST=VERIFY request).

## Logon/job initialization processing

When users cannot log on (or jobs cannot be initiated) or started procedures fail, check the following:

- For all types of users and jobs, check for an authorization message that indicates the cause of the failure, such as:
  - User profile not defined
  - User ID revoked
  - Incorrect or no password
  - Incorrect group name
  - Incorrect or no security label (depending on RACF options)
  - Attempt to change password or password phrase when RACF is in read-only mode or when the RACF database is locked.

If the application's message does not clearly indicate the source of the problem, check the RACF message. This message (ICH408I or ICH409I) might provide more information.

### Note:

1. You can find the message in one of the following places:

- The user's terminal
- The job log
- The security console
- The system log.

Also, equivalent information is in audit records generated by RACF. Some information might be in audit records generated by the caller of RACF.

2. For NJE jobs and SYSOUT, be aware that NODES profiles can cause the user ID, connect group, and security label to be translated to local values.
  3. For NJE jobs, if password verification is required by the NODES profile used to verify the user ID, any password sent with the job must be the password associated with the user ID on the execution node.
  4. If the ICH408I message indicates that access was denied because of a revoked user ID, you might want to resume that user ID. Check if the user ID is associated with the started procedure. If there was a user ID associated with the started procedure, this started procedure could not have begun successfully. After you resume the user ID, you must restart the started procedure or re-IPL.
- REQUEST=VERIFY processing might do some RACF authorization checks for the user. Also, the caller of RACF, or initial EXECs or procedures that are invoked automatically might require RACF authorization checking.

See [Table 58 on page 740](#) to see which resource classes could be checked from various types of sessions.

- Check whether an installation exit is causing the problem. Candidates include:
  - The SAF exits
  - Exits in the caller of RACF, such as JES or TSO
  - The REQUEST=VERIFY exits.

Table 58. Resource classes checked for logon and job initialization requests

Type of session	Classes that might be checked
TSO logons	TERMINAL, SECLABEL, TSOPROC, ACCTNUM, PERFGRP, TSOAUTH, and (depending on the user's TSO logon procedure) DATASET or others
CICS signons	TERMINAL, SECLABEL, and APPL
IMS signons	TERMINAL, SECLABEL, and APPL
Operators logging on to MCS consoles	CONSOLE and SECLABEL
Batch jobs	JESINPUT, SECLABEL, JESJOBS, SURROGAT
Inbound NJE jobs	NODES, JESINPUT, SECLABEL, JESJOBS, SURROGAT
Inbound SYSOUT	NODES, JESINPUT, SECLABEL
RJE remote signons or logons	JESINPUT, SECLABEL, FACILITY (checks for existence of RJE.userid profile)
For NJE and RJE remote (commands)	CONSOLE, NODES, SECLABEL, OPERCMDS, FACILITY (for each command, a check is made against the NJE.userid or RJE.userid profile in the FACILITY class)
MOUNT (MVS operator requests that a DASD device be made active), system address space, and started procedures	Check the STARTED class or started procedures table (ICHRIN03) entry
APPC/MVS allocation requests	APPCPORT, APPCLU, APPCTP, APPCSERV, APPCSI, SECLABEL, APPL, DATASET

---

## Appendix F. Accessibility

Accessible publications for this product are offered through [IBM Documentation for z/OS \(www.ibm.com/docs/en/zos\)](http://www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS documentation see [How to Send Feedback to IBM](#) to leave documentation feedback.



## Notices

---

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing  
IBM Corporation  
North Castle Drive, MD-NC119  
Armonk, NY 10504-1785  
United States of America*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Corporation  
Site Counsel  
2455 South Road*

Poughkeepsie, NY 12601-5400  
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

## Terms and conditions for product documentation

---

Permissions for the use of these publications are granted subject to the following terms and conditions.

### **Applicability**

These terms and conditions are in addition to any terms of use for the IBM website.

### **Personal use**

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

### **Commercial use**

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or



reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

## Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

## IBM Online Privacy Statement

---

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at [ibm.com/privacy](http://ibm.com/privacy) and IBM's Online Privacy Statement at [ibm.com/privacy/details](http://ibm.com/privacy/details) in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at [ibm.com/software/info/product-privacy](http://ibm.com/software/info/product-privacy).

## Policy for unsupported hardware

---

Various z/OS elements, such as DFSMSdfp, JES2, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

## Minimum supported hardware

---

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those

products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: [IBM Lifecycle Support for z/OS \(www.ibm.com/software/support/systemsz/lifecycle\)](http://www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

## RSA Secure code

---

This product contains code licensed from RSA Data Security Incorporated.



## Trademarks

---

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at [Copyright and Trademark information \(www.ibm.com/legal/copytrade.shtml\)](http://www.ibm.com/legal/copytrade.shtml).

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

UNIX is a registered trademark of The Open Group in the United States and other countries.

# Glossary

---

This glossary includes technical terms and abbreviations for RACF.

The following cross-references are used in this glossary:

- **See** refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- **See also** refers you to a related or contrasting term.

## access

The ability to use a protected resource.

## access authority

The privileges granted to a particular user or group when accessing a protected resource (such as the ability to read or to update a data set). For resources protected by RACF profiles, the access authorities are NONE, EXECUTE, READ, UPDATE, CONTROL, and ALTER. These authorities are hierarchical, with READ also granting EXECUTE, UPDATE granting READ, and so forth.

RACF also has access authorities of READ, WRITE, and EXECUTE (or SEARCH) when dealing with z/OS UNIX files and directories. Note that these authorities are not hierarchical, and that z/OS UNIX files are not protected by RACF profiles, although they do have access authorities.

## access ACL

An ACL that is used to provide protection for a file system object.

## access control

In computer security, ensuring that the resources of a computer system can be accessed only by authorized users in authorized ways.

## access control list (ACL)

(1) In computer security, a collection of all access rights for one object. In computer security, a list associated with an object that identifies all the subjects that can access the object and their access rights; for example, a list associated with a file that identifies users who can access the file and identifies their access rights to that file. (2) In z/OS UNIX, an extension to the base POSIX permission bits. Similar to the access list of a RACF profile, an ACL for a file system object contains entries that specify access permissions for individual users and groups.

## ACL

See access control list.

## access list

Synonym for *standard access list*. Contrast with *conditional access list*.

## ACEE

(accessor environment element) A control block that contains a description of the current user's security environment, including user ID, current connect group, user attributes, and group authorities. An ACEE is constructed during user identification and verification. See *ENVV object*.

## ADAU

See *automatic direction of application updates*.

## ADSP

See *automatic data set protection*.

## ADSP attribute

A user attribute that establishes an environment in which all permanent DASD data sets created by the user are automatically defined to RACF and protected with a discrete profile. See *automatic data set protection*.

## Advanced Program-to-Program Communication (APPC)

A set of interprogram communication services that support cooperative transaction processing in an SNA network. APPC is the implementation, on a given system, of SNA's LU type 6.2. See *LU type 6.2* and *APPC/MVS*.

**AIM**

See *application identity mapping (AIM)*.

**APF-authorized**

A type of system authorization using the authorized program facility (APF) that allows an installation to identify system or user programs that can use sensitive system functions. To maintain system security and integrity, a program must be authorized by the APF before it can access restricted functions, such as supervisor calls (SVC) or SVC paths.

**API**

See *application programming interface*.

**APPC**

See *Advanced Program-to-Program Communication*.

**APPC application**

See *transaction program (TP)*.

**APPC/MVS**

The implementation of SNA's LU 6.2 and related communication services in the MVS base control program.

**application identity mapping (AIM)**

Allows mapping between RACF user IDs and various application identities, such as those associated with z/OS UNIX, Novell Directory Services, and Lotus Notes.

**application programming interface (API)**

A software interface that enables applications to communicate with each other. An API is the set of programming language constructs or statements that can be coded in an application program to obtain the specific functions and services provided by an underlying operating system or service program.

**application user identity**

An alternate name by which a RACF user can be known to an application.

**appropriate privileges**

Describes which users can perform an action (such as execute a command, issue a syscall, and so forth) in a UNIX environment. Usually refers to having superuser authority or an appropriate subset of superuser authority.

**attribute**

See *user attribute* and *group-related user attribute*.

**AUDIT request**

The issuing of the RACROUTE macro with REQUEST=AUDIT specified. An AUDIT request is a general-purpose security request that a resource manager can use to audit.

**AUDITOR attribute**

A user attribute that allows the user to specify logging options on the RACF commands and list any profile (including its auditing options) using the RACF commands. Contrast with *group-AUDITOR attribute*.

**auditor authority**

Authority granted to a user with the AUDITOR attribute, or with a group-AUDITOR attribute in a RACF group.

**AUTH request**

The issuing of the RACROUTE macro with REQUEST=AUTH specified. The primary function of an AUTH request is to check a user's authorization to a RACF-protected resource or function. The AUTH request replaces the RACHECK function. See *authorization checking*.

**authentication**

Verification of the identity of a user or the user's eligibility to access an object.

Verification that a message has not been altered or corrupted.

A process used to verify the user of an information system or protected resources. See also *password*.

**authority**

The right to access objects, resources, or functions. See *access authority*, *class authority*, and *group authority*.

**authorization checking**

The action of determining whether a user is permitted access to a protected resource. Authorization checking refers to the use of RACROUTE REQUEST=AUTH, RACROUTE REQUEST=FASTAUTH, or any of the RACF callable services unless otherwise stated. Note, however, that other RACF functions can also perform authorization checking as a part of their processing. For example, RACROUTE REQUEST=VERIFY can also check a user's authority to use a terminal or application.

**automatic command direction**

An RRSF function that enables RACF to automatically direct certain commands to one or more remote nodes after running the commands on the issuing node. Commands can be automatically directed based on who issued the command, the command name, or the profile class related to the command. Profiles in the RRSFDATA class control to which nodes commands are automatically directed. See *automatic direction of application updates*, *automatic password direction* and *command direction*.

**automatic data set protection (ADSP)**

A system function, enabled by the SETROPTS ADSP specification and the assignment of the ADSP attribute to a user with ADDUSER or ALTUSER, that causes all permanent data sets created by the user to be automatically defined to RACF with a discrete RACF profile.

**automatic direction**

See *automatic command direction*, *automatic password direction*, and *automatic direction of application updates*.

**automatic direction of application updates**

An RRSF function that automatically directs ICHEINTY and RACROUTE macros that update the RACF database to one or more remote systems. Profiles in the RRSFDATA class control which macros are automatically directed, and to which nodes. See *automatic command direction* and *automatic password direction*.

**automatic password direction**

An RRSF function that extends password synchronization and automatic command direction to cause RACF to automatically change the password for a user ID on one or more remote nodes after the password for that user ID is changed on the local node. Profiles in the RRSFDATA class control for which users and nodes passwords are automatically directed. See *password synchronization*, *automatic command direction*, and *automatic direction of application updates*.

**automatic profile**

A tape volume profile that RACF creates when a RACF-defined user protects a tape data set. When the last data set on the volume is deleted, RACF automatically deletes the tape volume profile. Contrast with *nonautomatic profile*.

**backup data set**

A data set in the backup RACF database. For each data set in the primary RACF database, an installation should define a corresponding backup data set. See *backup RACF database*.

**backup RACF database**

A RACF database that reflects the contents of the primary RACF database. Backup RACF databases might be designated in the data set name table or specified at IPL time. You can switch to a backup database without a re-IPL if the primary RACF database fails. See *primary RACF database*.

**base ACL entry**

Same as permission bits (owner, group, other). The permissions can be changed using chmod. They are not physically part of the ACL.

**base segment**

The portion of a RACF profile that contains the fundamental information about a user, group, or resource. The base segment contains information that is common to all applications that use the profile.

**BER**

This term represents the Basic Encoding Rules specified in ISO 8825 for encoding data units described in abstract syntax notation 1 (ASN.1). See also *DER*.

**block update command (BLKUPD)**

A RACF diagnostic command used to examine or modify the content of individual physical records in a RACF data set.

**cache structure**

A coupling facility structure that contains data accessed by systems in a sysplex.

**callable service**

In z/OS UNIX, a request by an active process for a service. Synonymous with *syscall*.

**category**

See *security category*.

**CDMF**

See *Commercial Data Masking Facility*.

**CDT**

See *class descriptor table*.

**certificate**

See *digital certificate*.

**certificate authority**

An organization that issues digital certificates. The certificate authority authenticates the certificate owner's identity and the services that the owner is authorized to use, issues new certificates, renews existing certificates, and revokes certificates belonging to users who are no longer authorized to use them.

**certificate-authority certificate**

A type of certificate managed by RACF. See *digital certificate*.

**certificate name filter**

A general resource profile created by the RACDCERT MAP command that maps multiple user IDs to a digital certificate in order to simplify administration of certificates, conserve storage space in the RACF database, maintain accountability, or maintain access control granularity.

**CICS**

See *Customer Information Control System*.

**class**

A collection of RACF-defined entities (users, groups, and resources) with similar characteristics. Classes are defined in the class descriptor table (CDT), except for the USER, GROUP, and DATASET classes.

**class authority (CLAUTH)**

An attribute enabling a user to define RACF profiles in a class defined in the class descriptor table. A user can have class authorities to zero or more classes.

**class descriptor table (CDT)**

A table consisting of an entry for each class except the USER, GROUP, and DATASET classes. The CDT contains the classes supplied by IBM and the installation-defined classes. When this term appears without the preceding modifiers *dynamic* or *static*, it refers to the combination of the dynamic CDT, if it exists, and the static CDT.

**classification model 1**

See *single-subsystem scope*.

**classification model 2**

See *multiple-subsystem scope*.

**CLAUTH attribute**

See *class authority*.

**command direction**

An RRSF function that allows a user to issue a command from one user ID and direct that command to run in the RACF address space on the same system or on a different RRSF node, using the same or a different user ID. Before a command can be directed from one user ID to another, a user ID association must be defined between them using the RACLINK command.

**command prefix facility (CPF)**

An MVS facility that provides a registry for command prefixes. CPF ensures that two or more subsystems do not have the same or overlapping command prefixes for MVS operator commands.

**Commercial Data Masking Facility (CDMF)**

An encryption function that uses a weaker key (40 bit) of the Data Encryption Standard (DES) algorithm. RACF uses CDMF to mask the data portion of RRSF transaction processing message packets. CDMF is part of the IBM Common Cryptographic Architecture.

**common programming interface (CPI)**

An evolving application programming interface (API), supplying functions to meet the growing demands from different application environments and to achieve openness as an industry standard for communications programming. CPI-C provides access to interprogram services such as sending and receiving data, synchronizing processing between programs, and notifying a partner of errors in the communication.

**conditional access list**

The portion of a resource profile that specifies the users and groups that might access the resource at a specified level when a specified condition is true. For example, with program access to data sets, the condition is that the user must be executing the program specified in the access list. Contrast with *standard access list*.

**coordinator system**

In a RACF data sharing group, the system on which the system operator or administrator enters a RACF command that is propagated throughout the group. Contrast with *peer system*.

**coupling facility**

The hardware element that provides high-speed caching, list processing, and locking functions in a sysplex.

**CPF**

See *command prefix facility*.

**CPI-C**

See *common programming interface*.

**current connect group**

The group specified by a user when logging on to the system, or the user's default group if the user did not specify a group when logging on. With SETROPTS NOGRPLIST in effect, RACF uses the user's authority and this group's authority during access checking. With SETR GRPLIST in effect, RACF includes the authority of the user's other groups, if any, but the user still has only one "current connect group". You can use the &RACGPID variable in members of GLOBAL profiles to refer to the user's current connect group.

**current security label**

The security label that RACF uses in RACF authorization checking if the SECLABEL class is active. For interactive users, this is the security label specified when the user logged on, or (if no security label was specified) the default security label in the user's user profile. For batch jobs, this is the security label specified in the SECLABEL operand of the JOB statement, or (if no security label was specified) the user's current security label in the user profile associated with the job.

**custom field**

A field in a user or group profile that can be used to store installation data, and for which the installation can customize the keyword name and attributes. A custom field is defined in the CFDEF segment of a general resource profile in the CFIELD class. Installation data contained in a custom field is stored in the CSDATA segment of the user or group profile.

**Customer Information Control System (CICS)**

A program licensed by IBM that provides online transaction processing services and management for critical business applications. CICS runs on many platforms (from the desktop to the mainframe) and is used in various types of networks that range in size from a few terminals to many thousands of terminals. The CICS application programming interface (API) enables programmers to port applications among the hardware and software platforms on which CICS is available. Each product in the CICS family can interface with the other products in the CICS family, thus enabling interoperability.

**DASDVOL authority**

A preferred alternative to assigning the OPERATIONS or group-OPERATIONS attribute, DASDVOL authority allows you to authorize operations personnel to access only those volumes that they must maintain. Using DASDVOL authority is also more efficient for functions such as volume dumping, because only one authorization check for the volume needs to be issued, instead of individual requests for each data set on the volume. Note that modern data management software (such as DFSMSdss) does not require DASDVOL authority. Contrast with *OPERATIONS attribute*, and *group-OPERATIONS attribute*.

**data lookaside facility (DLF)**

A facility that processes DLF objects. A DLF object contains data from a single data set managed by Hiperbatch. The user (an application program) is connected to the DLF object, and the connected user can then access the data in the object through normal QSAM or VSAM macro instructions.

**data security**

The protection of data from intentional or unintentional unauthorized disclosure, modification, or destruction.

**data security monitor (DSMON)**

A RACF auditing tool that produces reports enabling an installation to verify its basic system integrity and data security controls.

**data set profile**

A profile that provides RACF protection for one or more data sets. The information in the profile can include the data set profile name, profile owner, universal access authority, access list, and other data. See *discrete profile* and *generic profile*.

**data sharing group, RACF**

A collection of one or more instances of RACF in a sysplex that have been identified to XCF and assigned to the group defined for RACF sysplex data sharing. RACF joins group IRRXCF00 when enabled for sysplex communication.

**data sharing mode**

An operational RACF mode that is available when RACF is enabled for sysplex communication. Data sharing mode requires installation of coupling facility hardware.

**Db2z/OS administrative authority**

A set of privileges, often covering a related set of objects, and often including privileges that are not explicit, have no name, and cannot be specifically granted. For example, the ability to terminate any utility job is included in the SYSOPR authority.

**Db2z/OS explicit privilege**

A privilege that has a name, and is held as the result of an SQL GRANT statement.

**default ACL**

An ACL that is specifically associated with a directory, and which gets inherited by an object created within the directory.

**default group**

The group specified in a user profile that provides a default current connect group for the user. See *current connect group*.

**DEFINE request**

The issuing of the RACROUTE macro with REQUEST=DEFINE specified or using a RACF command to add or delete a resource profile causes a DEFINE request. The DEFINE request replaces the RACDEF function.

**delegated resource**

A general resource that is eligible to be accessed by specially programmed applications that request RACF to check the daemon or application's authority for a resource when the client's authority is insufficient. Applications programmed in this way, such as the FTP daemon, are said to contain support for *nested* ACEEs because the identity of the daemon is said to be nested beneath the identity of the client for authorization purposes. See *nested ACEE*.

**delegation**

The act of giving users or groups the necessary authority to perform RACF operations.



**DER**

This term represents the Distinguished Encoding Rules, which are a subset of the Basic Encoding Rules. See also *BER*.

**digital certificate**

A digital document that binds a public key to the identity of the certificate owner, thereby enabling the certificate owner to be authenticated. A certificate is issued by a certificate authority.

RACF can manage three types of digital certificates:

- **certificate-authority certificate.** A certificate associated with a certificate authority and is used to verify signatures in other certificates.
- **site certificate.** A certificate associated with a server, or network entity other than a user or certificate authority.
- **user certificate.** A certificate associated with a RACF user ID that is used to authenticate the user's identity, and might also be used to represent a server.

**DIRAUTH request**

The issuing of the RACROUTE macro with REQUEST=DIRAUTH specified. A DIRAUTH request works on behalf of the message-transmission managers to ensure that the receiver of a message meets authorization requirements based on the security label.

**directed command**

A RACF command that is issued from a user ID on an RRSF node. It runs in the RACF subsystem address space on the same or a different RRSF node under the authority of the same or a different user ID. A directed command is one that specifies AT or ONLYAT. See *command direction* and *automatic command direction*.

**directory default ACL**

A model ACL that gets inherited by subdirectories that are created within the parent directory.

**directory model ACL**

See directory default ACL.

**discrete profile**

A resource profile that provides RACF protection for a single resource. Contrast with *generic profile* and *fully qualified generic profile*.

**discretionary access control**

An access control environment in which the resource owner determines who can access the resource. Contrast with *mandatory access control*.

**disjoint**

Pertaining to security labels, when the set of security categories that defines the first does not include the set of security categories that defines the second, and the set of security categories that defines the second does not include the set of security categories that defines the first. This also means that the first does not dominate the second and the second does not dominate the first. See *dominate*.

**distributed identity filter**

A mapping association between a RACF user ID and one or more distributed user identities which is stored in a general resource profile in the IDIDMAP class and administered using the RACMAP command. A distributed identity filter consists of one or more components of a distributed user's name and the name of the registry where the user is defined.

**DLF object**

When data lookaside facility (DLF) is active, the first attempt to access a QSAM or VSAM data set defined to DLF creates a DLF object. A DLF object contains data from a single data set managed by Hiperbatch. The user (an application program) is connected to the DLF object, and the connected user can then access the data in the object through normal QSAM or VSAM macro instructions.

**dominate**

One security label dominates a second security label when the security level that defines the first is equal to or greater than the security level that defines the second, and the set of security categories that defines the first includes the set of security categories that defines the second. A security label dominates itself since comparison of a security label with itself meets this definition.

## **DSMON**

See *data security monitor*.

## **dynamic CDT**

An optional portion of the class descriptor table that contains RACF classes built from profiles in the CDT general resource class. It does not include the required classes that comprise the supplied CDT (module ICHRRCDX) nor the optional classes that comprise the installation-defined CDT (module ICHRRCDE), if it exists. The dynamic CDT is processed as a logical extension of the static CDT. See also *static CDT*.

## **effective group identifier (effective GID)**

When the user connects to the system (for example, logs on to a TSO/E session), one group is selected as the user's current group. When a user becomes a z/OS UNIX user, the GID of the user's current group becomes the effective GID of the user's process. The user can access resources available to members of the user's effective GID. See *group identifier (GID)* and contrast with *real GID*.

## **effective user identifier (effective UID)**

When a user becomes a z/OS UNIX user, the UID from the user's RACF user profile becomes the effective UID of the user's process. The system uses the effective UID to determine if the user is a file owner. See *user identifier (UID)* and contrast with *real UID*.

## **EIM**

See *Enterprise identity mapping*.

## **EIM domain**

An LDAP name space that contains the enterprise identifiers, registry users, and relationships or associations between them.

## **Enterprise identity mapping (EIM)**

An infrastructure that user administration applications, servers, operating systems, and auditing tools can use to store identity mappings in a centralized, distributed registry (LDAP). The information is stored in LDAP to allow one user ID to be mapped to another (as long as the identities belong to the same application) using this support.

## **entity**

A user, group, or resource (for example, a DASD data set) that is defined to RACF.

## **envelope**

A container stored in the user's profile that contains the user's encrypted password or password phrase so that it can be retrieved and decrypted by authorized users of the R\_Admin callable service (IRRSEQ00) as part of a password synchronization solution, such as IBM Tivoli Directory Integrator.

## **ENVR object**

A transportable form of the ACEE that can be used within a single system to create the original ACEE without accessing the RACF database. It can be used, with limits, elsewhere in a single sysplex to recreate the original ACEE without accessing the RACF database.

## **equivalence**

Two security labels that contain the same security level and the same set of categories are considered equivalent, with each being dominated by and dominating the other.

## **erase-on-scratch**

The physical overwriting of data on a DASD data set when the data set is deleted (scratched).

## **extended ACL entry**

An ACL entry for an individual user or group.

## **EXTRACT request**

The issuing of the RACROUTE macro with REQUEST=EXTRACT specified. An EXTRACT request retrieves or replaces certain specified fields from a RACF profile or encodes certain clear-text (readable) data. The EXTRACT request replaces the RACXTRT function.

## **failsoft processing**

Processing that occurs when no data sets in the primary RACF database are available (RACF is installed but inactive). RACF cannot make decisions to grant or deny access. The operator is prompted frequently to grant or deny access to data sets. The resource manager decides on the action for general resource classes with a return code of 4.

Failsoft processing can also occur as the result of RVAR Y INACTIVE (temporary failsoft) or as the result of a serious system error requiring a re-IPL (permanent failsoft).

### **FASTAUTH request**

The issuing of the RACROUTE macro with REQUEST=FASTAUTH specified. The primary function of a FASTAUTH request is to check a user's authorization to a RACF-protected resource or function. A FASTAUTH request uses only in-storage profiles (brought into storage using RACF functions such as RACROUTE REQUEST=LIST) for faster performance than an AUTH request. The FASTAUTH request replaces the FRACHECK function. See *authorization checking*.

### **field-level access checking**

The RACF facility by which a security administrator can control access to segments, other than the base segment, in a RACF profile and fields in those segments.

### **file default ACL**

A model ACL that is inherited by files that are created within the parent directory.

### **file model ACL**

See file default ACL.

### **file permission bits**

In z/OS UNIX, information about a file that is used, along with other information, to determine if a process has read, write, or execute/search permission to a file or directory. The bits are divided into three parts, which are owner, group, and other.

### **file security packet (FSP)**

In z/OS UNIX, a control block containing the security data (file's owner user identifier (UID), owner group identifier (GID), and the permission bits) associated with the file. This data is stored with the file in the file system.

### **file system object**

Used to generically refer to either a file or directory.

### **file transfer protocol (FTP)**

In the Internet suite of TCP/IP-related protocols, an application layer protocol that transfers bulk data files between machines or hosts.

### **FMID**

See *function modification identifier*.

### **FRACHECK request**

RACROUTE REQUEST=FASTAUTH replaces the FRACHECK function. See *FASTAUTH request*.

### **FSP**

See *file security packet*.

### **FTP**

See *File Transfer Protocol*.

### **fully qualified generic profile**

A DATASET profile that was defined using the GENERIC operand and has a name that contains no generic characters. A fully qualified generic profile protects only resources whose names exactly match the name of the profile. Contrast with *discrete profile* and *generic profile*.

### **function modification identifier (FMID)**

A 7-character identifier that is used in elements associated with z/OS to identify the release of the element.

### **GDG**

See *generation data group*.

### **general resource**

Any resource, other than an MVS data set, that is defined in the class descriptor table (CDT). General resources include DASD volumes, tape volumes, load modules, terminals, IMS and CICS transactions, and installation-defined resource classes.

**general resource profile**

A profile that provides RACF protection for one or more general resources. The information in the profile can include the general resource profile name, profile owner, universal access authority, access list, and other data.

**general user**

A user who has limited RACF privileges, such as logging on, accessing resources, and creating data sets. General users typically use and create RACF-protected resources, but have no authority to administer resources other than their own.

**generation data group (GDG)**

A collection of data sets with the same base name, such as PAYROLL, that are kept in chronological order. Each data set in the GDG is called a generation data set, and has a name such as PAYROLL.G0001V00, PAYROLL.G0002V00, and so forth.

**generic profile**

A resource profile that can provide RACF protection for zero or more resources. The resources protected by a generic profile have similar names and identical security requirements, though with RACFVARS, a generic profile can protect resources with dissimilar names, too. For example, a generic data set profile can protect one or more data sets. Contrast with *discrete profile*.

**global access checking**

The ability to allow an installation to establish an in-storage table of default values for authorization levels for selected resources. RACF refers to this table before performing normal RACROUTE REQUEST=AUTH processing and grants the request without performing an AUTH request if the requested access authority does not exceed the global value. RACF uses this table to process AUTH requests faster and with less overhead (no checking of access lists, no auditing) when you have resources for which you decide to grant access to all users, except those with restricted user IDs. If the requested access does not exceed the access granted by the table, RACF bypasses most of its normal AUTH processing. Global access checking can grant the user access to the resource, but it cannot deny access.

**global resource serialization**

A mechanism using ENQ with the SYSTEMS option (or, in some older programs, the RESERVE option) to serialize resources across multiple z/OS images. It is used by RACF to serialize access to its database and to in-storage tables and buffers.

**globally RACLISTed profiles**

In-storage profiles for RACF-defined resources that are created by RACROUTE REQUEST=LIST and that are anchored from an ACEE. Globally RACLISTed in-storage profiles are shared across a system, such as the way that in-storage profiles created by SETROPTS RACLIST are shared. Contrast with *locally RACLISTed profiles*.

**group**

A collection of RACF-defined users who can share access authorities for protected resources.

**group-ADSP attribute**

A group-related user attribute similar to the ADSP attribute for a user, but assigned by using the CONNECT command to restrict its effect to those cases where the user creates data sets with that group as the high level qualifier of the data set name (or as determined by the naming convention table or exit).

**group-AUDITOR attribute**

A group-related user attribute similar to the AUDITOR attribute for a user, but assigned by using the CONNECT command to restrict the user's authority to resources that are within the scope of the group. Contrast with *AUDITOR attribute*.

**group authority**

An authority specifying which functions a user can perform in a group. The group authorities are USE, CREATE, CONNECT, and JOIN.

**group data set**

A RACF-protected data set in which either the high-level qualifier of the data set name or the qualifier supplied by an installation-naming convention table or exit routine is a RACF group name.

**group-GRPACC attribute**

A group-related user attribute similar to the GRPACC attribute for a user, but assigned by using the CONNECT command to restrict its effect to the specific group. Contrast with *GRPACC attribute*.

**group ID**

Obsolete term for *group name*.

**group identifier (GID)**

A number between 0 and 2147483647 that identifies a group of users to z/OS UNIX. The GID is associated with a RACF group name when it is specified in the OMVS segment of the group profile. See *real GID*. Contrast with *effective group identifier (effective GID)*.

**group name**

A string of 1 - 8 characters that identifies a group to RACF. The first character must be A - Z, # (X'7B'), \$ (X'5B'), or @ (X'7C'). The rest can be A - Z, #, \$, @, or 0 - 9.

**group-OPERATIONS attribute**

A group-related user attribute similar to the OPERATIONS attribute for a user, but assigned by using the CONNECT command to restrict its effect to those resources that are within the scope of the group.

If a person needs to perform maintenance activities on DASD volumes, it is more efficient (for RACF processing) and better (for limiting the resources the person can access) to give the person authority to those volumes using the PERMIT command than to assign the person the OPERATIONS or group-OPERATIONS attribute. Contrast with *DASDVOL authority* and *OPERATIONS attribute*.

**group profile**

A profile that defines a group. The information in the profile includes the group name, profile owner, and users in the group.

**grouping profile**

A profile in a resource group class.

**group-related user attribute**

A user attribute, assigned at the group level, that enables the user to control the resource, group, and user profiles associated with the group and its subgroups. Group-related user attributes include group-SPECIAL attribute, group-AUDITOR attribute, and group-OPERATIONS attribute. Contrast with *user attribute*.

**group-REVOKE attribute**

Assigned through the CONNECT command that prevents the user from using that group as the current connect group. Also prevents RACF from considering that group during authorization checking.

**group-SPECIAL attribute**

A group-related user attribute similar to the SPECIAL user attribute, but it is assigned by the CONNECT command to restrict the user's authority to users, groups, and resources within the scope of the group. Within this scope, it gives the user full control over everything except auditing options. However, it does not give the user authority to change global RACF options that will affect processing outside the group's scope. Contrast with *SPECIAL attribute*.

**GRPACC attribute**

With this attribute, any group data sets that the user defines to RACF (through the ADSP attribute, the PROTECT operand on the DD statement, or the ADDSD command) are automatically made accessible to other users in the group at the UPDATE level of access authority if the user defining the profile is a member of the group. Contrast with *group-GRPACC attribute*.

**ICB**

See *inventory control block*.

**ICL**

See *issued certificate list (ICL)*.

**ICHRIN03**

See *started procedures table*.

**inheritance**

The act of automatically associating an ACL with a newly created object without requiring administrative action.

**issued certificate list (ICL)**

PKI Services database containing the history of issued certificates.

**interprocess communication facilities (IPC)**

IPC facilities are services that allow different processes to communicate. Message passing (using message queues), semaphore sets, and shared memory services are forms of interprocess communication facilities.

**inventory control block (ICB)**

The first block in a RACF database. The ICB contains a general description of the database and, for the master primary data set, holds the RACF global options specified by SETROPTS.

**IPC**

See *interprocess communication facilities*.

**installation-defined CDT**

An optional portion of the CDT (module ICHRRCDE) that is installed by the installation. The function provided by this module can be replaced with the dynamic CDT function.

**issuer's distinguished name (IDN)**

The X.509 name that is associated with a certificate authority.

**kernel**

The part of z/OS UNIX that provides support for such services as UNIX I/O, process management, and general UNIX functionality.

**kernel address space**

The address space in which the z/OS UNIX kernel runs. See *kernel*.

**key**

In cryptography, a sequence of symbols that is used with a cryptographic algorithm for encrypting or decrypting data. See *private key* and *public key*.

**key ring**

A named collection of certificates for a specific user or server application used to determine the trustworthiness of a client or peer entity. Contrast to *virtual key ring*.

**label**

A usable "handle" for a certificate.

**LDAP**

See *lightweight directory access protocol*.

**lightweight access directory protocol (LDAP)**

Similar to directory access protocol (DAP), but simpler to use and has a programming interface; LDAP is composed of entries identified by their distinguished names.

**link pack area (LPA)**

An area of virtual storage containing reenterable routines from system libraries that are loaded at IPL time and can be used concurrently by all tasks in the system. The LPA presence in main storage saves loading time.

**LIST request**

The issuing of the RACROUTE macro with REQUEST=LIST specified. A LIST request builds in-storage profiles for a RACF general resource class. The LIST request replaces the RACLIST function.

**list-of-groups checking**

A RACF option (SETROPTS GRPLIST) that enables a user to access all resources available to all groups of which the user is a nonrevoked member, regardless of the user's current connect group. For any particular resource, RACF allows access based on the highest access among the groups in which the user is a member.

**local logical unit (local LU)**

A logical unit that resides on the local system. Contrast with *partner logical unit (partner LU)*, or *remote logical unit (remote LU)*, which typically resides on a remote system. When both the local and partner LUs reside on the same system, the LU through which communication is initiated is the local LU, and the LU through which communication is received is the partner LU.

**local mode**

An RRSF node is operating in local mode when it has no RRSF logical node connection with any other RRSF node.

**local transaction program (local TP)**

A transaction program that resides on the local system. Contrast with *partner transaction program* (*partner TP*), which typically resides on a remote system.

**locally RACLISTed profiles**

In-storage profiles for RACF-defined resources that are created by RACROUTE REQUEST=LIST and that are anchored from an ACEE. Locally RACLISTed in-storage profiles are not shared across a system, the way that in-storage profiles created by SETROPTS RACLIST are shared. Contrast with *globally RACLISTed profiles*.

**logging**

The recording of audit data about specific events.

**logical connection**

See *RRSF logical node connection*.

**logical unit (LU)**

A type of network accessible unit that enables users to gain access to network resources and communicate with each other.

**logical unit type 6.2 (LU type 6.2)**

The SNA logical unit type that supports general communication between programs in a cooperative processing environment. Also, the SNA logical unit type on which CPI-C and APPC/MVS TP conversation services are built.

**LPA**

See *link pack area*.

**LU**

See *logical unit*.

**LU type 6.2**

See *logical unit type 6.2*.

**MAC**

See *mandatory access control*.

**main system**

The system on a multisystem RRSF node that is designated to receive most of the RRSF communications sent to the node.

**managed user ID association**

A user ID association in which one of the associated user IDs is a managing user ID, and the other is a managed user ID. The managing user ID can run allowed RACF commands under the authority of the managed user ID. The managed user ID cannot run commands under the authority of the managing user ID. A managed user ID association does not allow password synchronization between the associated user IDs. Contrast with *peer user ID association*.

**mandatory access control (MAC)**

A means of restricting access to objects on the basis of the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (clearance) of subjects to access information of such sensitivity.

**mask**

A technique to provide protection against casual viewing of a password that has been defined or altered, when an encryption function is not available.

**master primary data set**

The first data set activated in the primary RACF database.

**MCS**

See *multiple console support*.

**MCS console**

A non-SNA device defined to MVS that is locally attached to an MVS system and is used to enter commands and receive messages.

**member**

A user belonging to a group.

**member profile**

A profile that defines a member and security level for that member.

**member system**

Any one of the MVS system images in a multisystem RRSF node.

**model ACL**

See default ACL.

**modeling**

See *profile modeling*.

**multilevel security**

A security policy that allows the classification of data and users based on a system of hierarchical security levels (for example: unclassified, secret, top secret) combined with a system of non-hierarchical security categories (for example: Project A, Project B, Project C). The system imposes mandatory access controls restricting which users can access data based on a comparison of the classification of the users and the data.

**multiple console support (MCS)**

The operator interface in an MVS system.

**multiple-subsystem scope**

A RACF classification model used in conjunction with the Db2z/OS access control module, or RACF external security module, to construct Db2z/OS resource names with the subsystem ID as part of the class name. Contrast with *single-subsystem scope*.

**multisystem node**

See *multisystem RRSF node*.

**multisystem RRSF node**

An RRSF node consisting of multiple MVS system images that share the same RACF database. One of the systems is designated to be the main system, and it receives the unsolicited RRSF communications sent to the node.

**MVS**

(multiple virtual storage) The mainframe operating system that allows multiple users to work simultaneously using the full amount of virtual storage.

**NCSC**

National Computer Security Center. The part of the U.S. Department of Defense that determines defense and security criteria.

**nested ACEE**

An ACEE that contains the security environment (ENVR object) of a daemon nested beneath the security environment of the client to support daemon access to delegated resources. See *ACEE* and *delegated resource*.

**network-qualified name**

An identifier for a partner LU in the form *netid.luname*, where *netid* is a 1 - 8 character network identifier and *luname* is a 1 - 8 character LU name.

**node**

See *RRSF node*.

**nonautomatic profile**

A tape volume profile that RACF creates when an RDEFINE command is issued or when tape data set protection is not active. A tape volume profile created in this manner is called a nonautomatic profile because RACF never deletes the profile except in response to the RDELETE command. Contrast with *automatic profile*.



**non-data sharing mode**

One of two normal modes of operation when RACF is enabled for sysplex communication and is the mode in which RACF communicates information using sysplex facilities to other instances of RACF, but does not make use of the coupling facility in doing so.

**OPERATIONS attribute**

A user attribute that grants the equivalent of ALTER access to all data sets unless the user or one of the user's connect groups appears explicitly in the access list of a data set's profile. If a user needs to perform maintenance activities on DASD volumes, granting DASDVOL authority to those volumes using the PERMIT command is preferred over assigning the OPERATIONS or group-OPERATIONS attribute. Note that most modern DASD maintenance programs do not require the OPERATIONS attribute. Contrast with *DASDVOL authority* and *group-OPERATIONS attribute*.

**operator identification card (OIDCARD)**

A small card with a magnetic stripe encoded with unique characters and used to verify the identity of a terminal operator to RACF.

**owner**

The user or group that creates a profile, or is specified as the owner of a profile. The owner can modify, list, or delete the profile.

**PADS**

See *program access to data sets (PADS)*.

**partner logical unit (partner LU)**

A logical unit that typically resides on a remote system. Often synonymous with *remote logical unit (remote LU)*. Contrast with *local logical unit (local LU)*, which resides on the local system. When both the local and partner LUs reside on the same system, the LU through which communication is initiated is the local LU, and the LU through which communication is received is the partner LU.

**partner transaction program (partner TP)**

A transaction program that resides on a remote system. Contrast with *local transaction program (local TP)*, which typically resides on the local system.

**PassTicket**

An alternative to the RACF password that permits workstations and client machines to communicate with the host. It allows a user to gain access to the host system without sending the RACF password across the network.

**password**

A string of characters known to a user who must specify it to gain full or limited access to a system and to the data stored within it. RACF uses a password to verify the identity of the user.

**password envelope**

See *envelope*.

**password synchronization**

An option that can be specified when a peer user ID association is defined between two user IDs. If password synchronization is specified for a user ID association, then whenever the password for one of the associated user IDs is changed, the password for the other user ID is automatically changed to the newly defined password. See *automatic password direction*.

**password phrase**

A longer string of mixed characters known to a user who must specify it to gain full or limited access to a system and to the data stored within it. RACF uses a password phrase to verify the identity of the user. Used as a more secure alternative to the password.

**password phrase envelope**

See *envelope*.

**peer system**

In a RACF data sharing group, any system to which RACF propagates a command entered by the system operator or administrator. Contrast with *coordinator system*.

**peer user ID association**

A user ID association that allows either user ID to run allowed RACF commands under the authority of the other user ID using command direction. A peer user ID association can also establish password synchronization between the associated user IDs. Contrast with *managed user ID association*.

**permission bits**

In z/OS UNIX, part of security controls for directories and files stored in the z/OS UNIX file system. Used to grant read, write, search (just directories), or execute (just files) access to owner, file or directory owning group, or all others.

**persistent verification (PV)**

A VTAM security option for conversation-level security between two logical units (LUs) that provides a way of reducing the number of password transmissions by eliminating the need to provide a user ID and password on each attach (allocate) during multiple conversations between a user and a partner LU. The user is verified during the signon process and remains verified until the user has been signed off the partner LU.

**PKCS**

See *public key cryptographic standards*.

**PKI**

See *public key infrastructure*.

**PKIX**

See *public key infrastructure standards*.

**POSIT**

A number specified for each class in the class descriptor table that identifies a set of flags that control RACF processing options.

**POSIX**

(Portable Operating System Interface For Computer Environments) An IEEE standard for computer operating systems.

**primary data set**

A data set in the primary RACF database. See *master primary data set*.

**primary RACF database**

The RACF database designated in the data set name table, or specified at IPL time, that contains the RACF profiles used for authorization checking. The primary RACF database might consist of as many as 90 data sets. See *backup RACF database*.

**private key**

In public key cryptography, a key that is known only to its owner. Contrast with *public key*.

**problem state**

A state during which a processing unit cannot execute input/output and other privileged instructions. Contrast with *supervisor state*.

**process**

In z/OS UNIX, a function created by a `fork()` request. See *task*.

**profile**

Data that describes the significant characteristics of a user, a group of users, or one or more computer resources. A profile contains a base segment, and optionally, a number of other segments. See *data set profile*, *discrete profile*, *general resource profile*, *generic profile*, *group profile*, and *user profile*.

**profile list**

A list of profiles indexed by class (for general resources) or by the high-level qualifier (for data set profiles) and built in storage by the RACF routines.

**profile modeling**

The ability for a user or an installation to copy information (such as universal access authority or access lists) from an existing resource profile when defining a new resource profile. This might occur automatically when using ADDSD based on the MODEL specification in a USER or group PROFILE, or manually with the FROM keyword of the ADDSD and RDEFINE commands, or with keywords on RACROUTE REQUEST=DEFINE.

**program access to data sets (PADS)**

A RACF function that enables an authorized user or group of users to access one or more data sets at a specified access authority only while running a specified RACF-controlled program. See *program control*.

**program control**

A RACF function that enables an installation to control who can run RACF-controlled programs. See *program access to data sets*.

**protected resource**

A resource defined to RACF for the purpose of controlling access to the resource. Some of the resources that can be protected by RACF are DASD volumes, tape volumes, load modules, terminals, IMS and CICS transactions, and installation-defined resource classes.

**protected user ID**

A user ID that cannot enter the system by any means that requires a password or password phrase, and cannot be revoked by incorrect password and password phrase attempts. Assigning a protected user ID to z/OS UNIX, a UNIX daemon, or another important started task or subsystem assures that the ID cannot be used for other purposes, and that functions will not fail because the ID has been revoked.

**public key**

In public key cryptography, a key that is made available to everyone. Contrast with *private key*.

**public key cryptography**

Cryptography in which public keys and private keys are used for encryption and decryption. One party uses a common public key and the other party uses secret private key. The keys are complementary in that if one is used to encrypt data, the other can be used to decrypt it.

**public key cryptographic standards (PKCS)**

Set of standards developed by RSA Corporation to facilitate interoperability for cryptographic protocols.

**public key infrastructure (PKI)**

The set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke public key certificates based on public key cryptography.

**public key infrastructure Standards (PKIX)**

Set of standards needed to support an X.509-based PKI.

**PV**

See *persistent verification*.

**RACDEF request**

The DEFINE function replaces the RACDEF function. See *DEFINE request*.

**RACF**

See *Resource Access Control Facility*.

**RACF access control module**

A Db2z/OS module that receives control from the Db2z/OS access control authorization exit point (DSNX@XAC) to handle Db2z/OS authorization checks. This term applies only to the Db2z/OS module available beginning with Db2z/OS Version 8.

**RACF Db2z/OS external security module**

A RACF module that receives control from the Db2z/OS access control authorization exit point (DSNX@XAC) to handle Db2z/OS authorization checks. This term applies to the RACF module available for Db2z/OS Version 7 and earlier.

**RACF database**

The repository for the security information that RACF maintains.

**RACF data set**

One of the data sets comprising the RACF database.

**RACF-indicated**

Pertaining to a data set for which the RACF indicator is set on. If a data set is RACF-indicated, a user can access the data set only if a RACF profile or an entry in the global access checking table exists

for that data set. On a system without RACF, a user cannot access a RACF-indicated data set until the indicator is turned off. For VSAM data sets, the indicator is in the catalog entry. For non-VSAM data sets, the indicator is in the data set control block (DSCB). For data sets on tape, the indicator is in the RACF tape volume profile of the volume that contains the data set.

**RACF manager**

The routines within RACF that provide access to the RACF database. Contrast with *RACF storage manager*.

**RACF-protected**

Pertaining to a resource that has either a discrete profile or an applicable generic profile. A data set that is RACF-protected by a discrete profile must also be RACF-indicated.

**RACF remote sharing facility (RRSF)**

A set of RACF functions that links together multiple RACF databases, allowing remote RACF administration and password synchronization.

**RACF remove ID utility**

A RACF utility that identifies references to user IDs and group names in the RACF database. The utility can be used to find references to residual user IDs and group names or specified user IDs and group names. The output from this utility is a set of RACF commands that can be used to remove the references from the RACF database after review and possible modification. See *residual user ID*.

**RACF report writer**

A RACF function that produces reports on system use and resource use from information found in the RACF SMF records. However, the preferred method for producing RACF SMF reports is the RACF SMF data unload utility (IRRADU00).

**RACF segment**

Obsolete term for *base segment*.

**RACF SMF data unload utility (IRRADU00)**

A RACF utility that enables installations to create a sequential file from the security-relevant audit data. The sequential file can be viewed directly, used as input for installation-written programs, and manipulated with sort/merge utilities. It can also be uploaded to a database manager (such as Db2z/OS) to process complex inquiries and create installation-tailored reports. See *SMF records*.

**RACF storage manager**

Manages the allocation of storage for the RACF programs running on a system.

**RACHECK request**

The AUTH request replaces the RACHECK function. See *AUTH request*.

**RACINIT request**

The VERIFY request replaces the RACINIT function. See *VERIFY request*.

**RACLIST request**

The LIST request replaces the RACLIST function. See *LIST request*.

**RACLISTed profiles**

See *locally RACLISTed profiles* and *globally RACLISTed profiles*.

**RACROUTE macro**

An assembler macro that provides a means of calling RACF to provide security functions, including the *AUDIT request*, *AUTH request*, *DEFINE request*, *DIRAUTH request*, *EXTRACT request*, *FASTAUTH request*, *LIST request*, *SIGNON request*, *STAT request*, *TOKENBLD request*, *TOKENMAP request*, *TOKENXTR request*, *VERIFY request*, and *VERIFYX request*.

**RACSTAT request**

The STAT request replaces the RACSTAT function. See *STAT request*.

**RACXTRT request**

The EXTRACT request replaces the RACXTRT function. See *EXTRACT request*.

**RBA**

See *relative byte address*.

**read-only auditor authority**

Authority granted to a user with the ROAUDIT attribute.

**read-only mode**

A recovery mode of operation when RACF is enabled for sysplex communication. Read-only mode does not allow updates to be made to the RACF database except for statistics generated during logon and job initiation.

**real GID**

The attribute of a process that, at the time of process creation, identifies the group of the user who created the process. See *group identifier (GID)*. Contrast with *effective group identifier (effective GID)*.

**real UID**

The attribute of a process that, at the time of process creation, identifies the user who created the process. See *user identifier (UID)*. Contrast with *effective user identifier (effective UID)*.

**relative byte address (RBA)**

The address in the RACF database.

**relative distinguished name (RDN)**

One component of a distinguished name.

**remote logical unit (remote LU)**

A logical unit that resides on a remote system. Often synonymous with *partner logical unit (partner LU)*. Contrast with *local logical unit (local LU)*, which typically resides on the local system.

**residual authority**

References in the RACF database to group names and user IDs that have been deleted.

**residual group name**

References in the RACF database to a group name that has been deleted.

**residual user ID**

References in the RACF database to a user ID that has been deleted.

**Resource Access Control Facility (RACF)**

A component of z/OS Security Server that provides access control by identifying and verifying the users to the system, authorizing access to protected resources, logging detected unauthorized attempts to enter the system, logging unauthorized attempts to enter the system, and logging detected accesses to protected resources. RACF for z/VM is available as a feature of z/VM.

**resource group profile**

A general resource profile in a resource grouping class. A resource group profile can provide RACF protection for one or more resources with *unlike* names. See *resource grouping class*.

**resource grouping class**

A RACF class in which resource group profiles can be defined. A resource grouping class is related to another class, sometimes called a *member class*. For example, the resource grouping class GTERMINL is related to the class TERMINAL. See *resource group profile*.

**resource profile**

A profile that provides RACF protection for one or more resources. USER, GROUP, and CONNECT profiles are not resource profiles. The information in a resource profile can include the profile name, profile owner, universal access authority, access list, and other data. Resource profiles can be discrete profiles or generic profiles. See *discrete profile* and *generic profile*.

**RESTRICTED attribute**

A user attribute that can be assigned to a shared user ID, such as PUBLIC or ANONYMOS, or a user ID used with a certificate name filter, to prevent the user ID from being used to access protected resources it is not specifically authorized to access. Restricted users cannot gain access to protected resources through global access checking, UACC, or an ID(\*) entry on the access list, and optionally, to a z/OS UNIX file system object through the 'other' bits.

**reverse mandatory access check**

A mandatory access check in which the security label of the resource must dominate the security label of the user in order for the user to be granted access to the resource.

**REVOKE attribute**

A user attribute that prevents a RACF-defined user from entering the system.

**ROAUDIT attribute**

A user attribute that allows the user to list any profile (including its auditing options) using the RACF commands. Contrast with AUDITOR attribute and group-AUDITOR attribute.

**role**

In Tivoli products, a functional grouping of user authorizations. A ROLE profile represents a role and identifies the authorizations associated with that role.

**RRSF**

See *RACF remote sharing facility*.

**RRSF logical node connection**

Two RRSF nodes are logically connected when they are properly configured to communicate through APPC/MVS or TCP/IP, and they have each been configured by the TARGET command to have an OPERATIVE connection to the other.

**RRSF network**

Two or more RRSF nodes that have established RRSF logical node connections to each other.

**RRSF node**

An MVS system image or a group of MVS system images sharing a RACF database, which has been defined as an RRSF node, single-system RRSF node, or multisystem RRSF node to RACF by a TARGET command. See *RRSF logical node connection*.

**RTOKEN**

The RACF resource security token. An RTOKEN is an encapsulation or representation of the security characteristics of a resource. Resource managers, for example JES, can assign RTOKENs to the resources they manage; for example, JES spool files. See *UTOKEN* and *STOKEN*.

**SAF**

See *System Authorization Facility*.

**security**

See *data security*.

**security category**

A non-hierarchical grouping of sensitive information used to control access to data.

**security classification**

The use of security categories, a security level, or both, to impose additional access controls on sensitive resources. An alternative way to provide security classifications is to use security labels.

**security label**

An installation-defined name that corresponds to a specific RACF security level with a set of zero or more security categories. This is equivalent to the NCSC term *sensitivity label*.

**security level**

An installation-defined name that corresponds to a numerical security level; the higher the number, the higher the security level.

**security token**

A collection of identifying and security information that represents data to be accessed, a user, or a job. This contains a user ID, group name, security label, node of origin, and other information.

**segment**

A portion of a profile. The format of each segment is defined by a template.

**SETROPTS RACLISTed profiles**

See *globally RACLISTed profiles*.

**SFS**

See *Shared File System*.

**shared GID**

An OMVS GID value that has been assigned to more than one group.

**shared UID**

An OMVS UID value that has been assigned to more than one user.

**shared file system (SFS)**

On z/VM, a part of CMS that lets users organize their files into groups known as directories and selectively share those files and directories with other users.

**signed-on-from list**

A list of user entries identifying those users who have been signed on from a partner LU to a local LU and is associated with persistent verification.

**SIGNON request**

The issuing of the RACROUTE macro with REQUEST=SIGNON specified. A SIGNON request is used to manage the signed-on-from lists associated with persistent verification.

**single-subsystem scope**

A classification model used in conjunction with the Db2z/OS access control module, or RACF external security module, to construct Db2z/OS classes with the subsystem ID as part of the class name. Contrast with *multiple-subsystem scope*.

**single-system node**

See *single-system RRSF node*.

**single-system RRSF node**

An RRSF node consisting of one MVS system image.

**site certificate**

A type of certificate managed by RACF. See *digital certificate*.

**SMF**

See *System Management Facility*.

**SMF data unload utility**

See *RACF SMF data unload utility*.

**SMF records**

Records and system or job-related information collected by the System Management Facility (SMF) and used in billing users, reporting reliability, analyzing the configuration, scheduling jobs, summarizing direct access volume activity, evaluating data set activity, profiling system resource use, and maintaining system security.

Variable-length process or status records from the SMF data set that are written to the SMF log data set. These records vary in layout based on the type of system information they contain. See *RACF SMF data unload utility*.

**SMS**

See *Storage Management Subsystem*.

**SNA**

See *System Network Architecture (SNA)*.

**source user ID**

The source half of a source user ID and target user ID pair that has an established user ID association between them. For command direction the source user ID is the user ID that issued the command that is being directed. For password synchronization the source user ID is the user ID whose password changed, causing a change to the password of the target user ID. Contrast with *target user ID*.

**SPECIAL attribute**

A user attribute that gives the user full control over all of the RACF profiles in the RACF database and allows the user to issue all RACF commands, except for commands and operands related to auditing. Contrast with *group-SPECIAL attribute*.

**split database**

A RACF database that has been divided among multiple data sets.

**standard access list**

The portion of a resource profile that specifies the users and groups that might access the resource and the level of access granted to each. Synonymous with *access list*. Contrast with *conditional access list*.

**started procedures table (ICHRIN03)**

Associates the names of started procedures with specific RACF user IDs and group names. It can also contain a generic entry that assigns a user ID or group name to any started task that does not have a matching entry in the table. However, it is recommended that you use the STARTED class for most cases rather than the started procedures table.

**static CDT**

The non-dynamic portion of the class descriptor table that is contained in the supplied CDT (module ICHRRCDX) and the optional installation-defined CDT (module ICHRRCDE). See also *dynamic CDT*.

**STAT request**

The issuing of the RACROUTE macro with REQUEST=STAT specified. A STAT request determines if RACF is active and (optionally) if a given resource class is defined to RACF and active. The STAT request replaces the RACSTAT function.

**STOKEN**

A UTOKEN associated with a user who has submitted work. See *UTOKEN* and *RTOKEN*.

**Storage Management Subsystem (SMS)**

A DFSMS facility used to automate and centralize storage management by providing the storage administrator with control over data class, storage class, management class, storage group, and automatic class selection routine definitions.

**structure**

See *cache structure*.

**stub**

A function that connects with the specified library, but remains outside the specified library.

A protocol extension procedure.

**subject's distinguished name (SDN)**

The X.509 name in a digital certificate that is associated with the name of the subject.

**superuser**

In z/OS UNIX, a system user who operates with the special privileges needed to perform a specified administrative task.

**superuser authority**

In z/OS UNIX, the unrestricted authority to access and modify any part of the operating system, usually associated with the user who manages the system.

**supervisor**

The part of a control program that coordinates the use of resources and maintains the flow of processing unit operations. Synonym for *supervisory routine*.

**supervisor state**

A state during which a processing unit can execute input/output and other privileged instructions. Contrast with *problem state*.

**supervisory routine**

A routine, usually part of an operating system, that controls the execution of other routines and regulates the flow of work in a data processing system. Synonymous with *supervisor*.

**supplied CDT**

The required portion of the CDT (module ICHRRCDX) that is supplied by IBM and shipped with RACF. Classes defined in the supplied CDT must not be modified by the installation.

**syscall**

See *callable service*.

**sysplex (system complex)**

Multiple systems communicating and cooperating with each other through multisystem hardware elements and software services to process the installation's workloads.

**sysplex communication**

An optional RACF function that allows the system to use XCF services and communicate with other systems that are also enabled for sysplex communication.



**system authorization facility (SAF)**

An interface defined by MVS that enables programs to use system authorization services in order to control access to resources, such as data sets and MVS commands. SAF either processes security authorization requests directly or works with RACF, or other security product, to process them.

**system call**

In z/OS UNIX, a synonym for *callable service*.

**system complex**

See *sysplex*.

**System Management Facility (SMF)**

The part of the operating system that collects and records system and job-related information used in billing users, reporting reliability, analyzing the configuration, scheduling jobs, summarizing direct access volume activity, evaluating data set activity, profiling system resource use, and maintaining system security. The information is recorded in the SMF log data set.

**Systems Network Architecture (SNA)**

The IBM architecture that defines the logical structure, formats, protocols, and operational sequences for transmitting information units through, and controlling the configuration and operation of, networks. The layered structure of SNA allows the ultimate origins and destinations of information, that is, the users, to be independent of and unaffected by the specific SNA network services and facilities used for information exchange.

**tape volume set**

The collection of tape volumes on which a multivolume data set resides. A volume set is represented in one RACF profile.

**tape volume table of contents (TVTOC)**

Information about a tape data set that RACF stores in the tape volume profile for the volume on which the data set resides. The TVTOC includes the data set name, data set sequence number, creation date, and an indicator as to whether a discrete tape data set profile exists.

**target node**

An RRSF node that a given RRSF node is logically connected to, as a result of a TARGET command. See *local node* and *remote node*.

**target user ID**

The target half of a source user ID and target user ID pair that has an established user ID association between them. For command direction, the target user ID is the user ID specified on the AT or ONLYAT keyword, and is the user ID under whose authority the command is run on the specified node. For password synchronization, the target user ID is the user ID whose password RACF automatically updates when the password for the source user ID is changed. Contrast with *source user ID*.

**task**

A basic unit of work to be performed or a process and the procedures that run the process.

**template**

Contains mappings of the profiles on the RACF database.

**token**

A real or virtual device that stores cryptographic data objects such as keys and digital certificates.

**TOKENBLD request**

The issuing of the RACROUTE macro with REQUEST=TOKENBLD specified. A TOKENBLD request builds a UTOKEN.

**TOKENMAP request**

The issuing of the RACROUTE macro with REQUEST=TOKENMAP specified. A TOKENMAP request maps a token in either internal or external format, allowing a caller to access individual fields within the UTOKEN.

**TOKENXTR request**

The issuing of the RACROUTE macro with REQUEST=TOKENXTR specified. A TOKENXTR request extracts a UTOKEN from the current address space, task or a caller-specified ACEE.

**TP**

See *transaction program*.

**tranquility**

Keeping the security classification of a resource constant while it is in use; keeping the security classification of a user constant while active.

**transaction program (TP)**

A program that processes transactions in an SNA network.

**TVTOC**

See *tape volume table of contents*.

**UACC**

See *universal access authority*.

**UADS**

See *user attribute data set*.

**universal access authority (UACC)**

The default access authority that applies to a resource if the user or group is not specifically permitted access to the resource, unless the user is restricted. The universal access authority can be any of the access authorities.

**universal group**

A user group defined using the UNIVERSAL operand of the ADDGROUP command. Universal groups are expected to have a large number of members and are unlikely to be deleted. Group profiles for universal groups do not contain complete membership information, and the LISTGRP command is not recommended to list members. Using the output of the database unload utility (IRRDBU00) is the best way to list members of a universal group.

**user**

A person who requires the services of a computing system.

**user attribute**

The extraordinary privileges, restrictions, and processing environments assigned to a user. The user attributes are SPECIAL, AUDITOR, ROAUDIT, CLAUTH, OPERATIONS, GRPACC, ADSP, and REVOKE.

**user attribute data set (UADS)**

In TSO, a partitioned data set with a member for each authorized user. Each member contains the appropriate passwords, user identifications, account numbers, LOGON procedure names, and user characteristics that define the user.

**user certificate**

A type of certificate managed by RACF. See *digital certificate*.

**user data set**

A data set defined to RACF in which either the high-level qualifier of the data set name or the qualifier supplied by an installation exit routine is a RACF user ID.

**user ID**

A RACF user ID. A string of 1 - 8 alphanumeric characters that uniquely identifies a RACF user, procedure, or batch job to the system. For TSO users, the user ID cannot exceed 7 characters and must begin with an alphabetic, #, \$, or @ character. The user ID is defined by a user profile in the RACF database and is used as the name of the profile.

**user ID association**

A relationship between two user IDs, established through the RACLINK command, which is required for command direction and password synchronization between the user IDs. See *peer user ID association* and *managed user ID association*.

**user identification**

See *user ID*.

**user identification and verification**

The acts of identifying and verifying a RACF-defined user to the system during logon or batch job processing. RACF identifies the user by the user ID and verifies the user by the password, password phrase, PassTicket, verified digital certificate, or operator identification card supplied during logon processing or the password supplied on a batch JOB statement.

**user identifier (UID)**

A number between 0 and 2147483647 that identifies a user to z/OS UNIX. The UID is associated with a RACF user ID when it is specified in the OMVS segment of the user profile. It can be contained in an object of type `uid_t`, that is used to identify a system user. When the identity of the user is associated with a process, a UID value is referred to as a real UID, an effective UID, or an (optional) saved set UID. See *real UID*. Contrast with *effective user identifier (effective UID)*.

**user name**

In RACF, 1 - 20 alphanumeric characters that represent a RACF-defined user. Contrast with *user ID*.

**user profile**

A description of a RACF-defined user that includes the user ID, user name, default group name, password, password phrase, profile owner, user attributes, and other information. A user profile can include information for subsystems such as TSO and DFP.

**UTOKEN**

The RACF user security token. A UTOKEN is an encapsulation or representation of the security characteristics of a user. RACF assigns a UTOKEN to each user in the system. See *STOKEN* and *RTOKEN*.

**verification**

See *user identification and verification*.

**VERIFY request**

The issuing of the RACROUTE macro with REQUEST=VERIFY specified. A VERIFY request is used to verify the authority of a user to enter work into the system. The VERIFY request replaces the RACINIT function.

**VERIFYX request**

The issuing of the RACROUTE macro with REQUEST=VERIFYX specified. A VERIFYX request verifies a user and builds a UTOKEN, and handles the propagation of submitter ID.

**virtual key ring**

The set of certificates owned by a user ID and used by a user or server application to determine the trustworthiness of a client or peer entity. Each RACF user ID is associated with a virtual key ring. In contrast to a real key ring, a virtual key ring is not added to RACF. In addition, the private key cannot be retrieved from a CERTAUTH or SITE virtual key ring, as it can be from a real key ring. The most common type is the CERTAUTH virtual key ring which is used when an application uses a key ring to validate the certificates of others but has no need for its own certificate and private key.

**virtual machine (VM)**

An operating system that appears to be at the exclusive disposal of the particular user, but whose functions are accomplished by sharing the resources of a real data processing system.

In z/VM, the operating system that represents the virtual processors, virtual storage, virtual devices, and virtual channel subsystem allocated to a single user. A virtual machine also includes any expanded storage dedicated to it.

**VM**

See *virtual machine*.

**workspace data sets**

VSAM data sets used by RACF for queuing requests sent to and received from target nodes in an RRSF environment.

**writedown mode**

The setting of an address space at which it can create output data at a lower security label than the current security label of the address space on a system where writedown is normally disallowed because the RACF MLS(FAILURES) option is in effect.

**writedown privilege**

The ability of users to set their address spaces to *writedown mode* in which they are able to write data to an object with a lower security label than the user's current security label on a system where writedown is normally disallowed because the RACF MLS(FAILURES) option is in effect.

### **X.500**

ITU/ISO 9594 standard for an open system directory information tree; includes protocols for access and security.

### **z/OS**

A program licensed by IBM that not only includes and integrates functions previously provided by many IBM software products, including the MVS operating system, but also:

1. Is an open, secure operating system for IBM enterprise servers
2. Complies with industry standards
3. Is based on the new 64-bit z/Architecture®
4. Supports technology advances in networking server capability, parallel processing, and object-oriented programming.

### **z/OS UNIX group identifier (GID)**

See *group identifier (GID)*.

### **z/OS UNIX System Services (z/OS UNIX)**

The set of functions provided by the shells, utilities, kernel, file system, debugger, Language Environment, and other elements of the z/OS operating system that allows users to write and run application programs that conform to UNIX standards.

### **z/OS UNIX user identifier (UID)**

See *user identifier (UID)*.

# Index

## Special Characters

`_POSIX_CHOWN_RESTRICTED` constant [527](#)  
`?????? UACC` [555](#)  
`?????? user ID` [481](#)  
`*` (asterisk)  
    generic character in profile names  
        specifying [152](#), [189](#)  
    on the ID operand of the PERMIT command  
        authorization checking [722](#), [723](#)  
        contrasted with UACC [6](#), [158](#), [366](#)  
        specifying [6](#), [158](#)  
`**` (double asterisk)  
    generic character in profile names  
        specifying [152](#), [189](#)  
    suggested replacement for %\* in general resource  
    profile names [189](#)  
`**SYSUT` data set  
    reserved name [152](#)  
`**SYSUT` high-level qualifier  
    PROTECTALL [120](#)  
`&`  
    generic character in general resource profile names  
        specifying [188](#)  
`&&TEMP` data set  
    reserved name [152](#)  
`&RACGPID`  
    comparison with GRPACC attribute [198](#)  
    global access checking [197](#)  
`&RACLNDE`  
    and PROPCNTL profiles [457](#)  
    creating [466](#)  
    recommended for JESJOBS profiles [462](#)  
    recommended for NODES profiles [469](#)  
`&RACLNDE` profile  
    checked when JES2 reloads offloaded data [489](#)  
    creating [482](#)  
    description [218](#), [482](#)  
`&RACLNDE` variable  
    recommended for JESSPOOL profiles [484](#)  
`&RACUID`  
    field-level access checking [202](#)  
    global access checking [197](#)  
    specifying in the HOME directory path [521](#)  
    using with USER.OMVS profiles [52](#)  
`&SUSER` value  
    on ADDMEM operand in NODES class  
        check of submitting user ID and node [477](#)  
        description [459](#)  
        example showing simple NJE user translation [479](#)  
`%` (percent sign)  
    generic character in profile names  
        specifying [152](#), [189](#)  
`%*`  
    in general resource profile names [189](#)  
`+++++++` user ID [481](#)  
`$SUBMIT.userid` profile (FACILITY class)

migrating to SURROGAT profiles [456](#)

## Numerics

3480 tape drives with IDRC feature  
    and IEC.TAPERING profile in FACILITY class [181](#)  
3490 tape drives  
    and IEC.TAPERING profile in FACILITY class [181](#)  
4-digit device  
    using [233](#)

## A

ACBs  
    controlling who can open [246](#)  
access attempts  
    logging [2](#)  
access authority  
    description [6](#)  
    for applications [733](#)  
    for consoles [229](#)  
    for DASD data sets [163](#)  
    for tape volume profiles [173](#)  
    granting or denying for general resource class [192](#)  
    required by IBM support personnel [367](#)  
    required for terminals [730](#)  
    responsibility for assigning [2](#)  
    summary of authorities and commands [703](#)  
    TSO resources [508](#)  
    using started procedures [141](#)  
access control lists (ACLs)  
    for z/OS UNIX data  
        [530](#)  
access list  
    assigning access authorities for consoles [229](#)  
    assigning access authorities for DASD data sets [163](#)  
    authority of a user not in for data set [157](#)  
    authority of a user not in for general resource [192](#)  
    conditional  
        data set profiles [157](#)  
        general resource profiles [193](#)  
    controlling access to data sets on tape volume [174](#)  
    creating for DFP field-level access checking [504](#)  
    creating for field-level access checking [202](#)  
    example of command to create entry for tape volume  
        [174](#)  
    example of command to delete from tape volume profile  
        [174](#)  
    for discrete data set profile [151](#)  
    in GDG base name profile [161](#)  
    limiting the size [193](#)  
    listing invalid user IDs in [382](#)  
    reducing effort of maintaining [89](#)  
    refreshing for global access checking [130](#)  
    sharing for a multivolume tape data set [181](#)  
access to resources

- access to resources (*continued*)
  - authorizing only for RACF-defined users [366](#)
  - RACF authorization checking for [718](#)
- access to system
  - limiting [73](#)
  - terminals [227](#)
- accessibility
  - contact IBM [741](#)
- account number for TSO
  - protecting [507](#)
- ACCTNUM class
  - activating [508](#)
  - authorization checking for [510](#)
  - considerations [509](#)
  - description [691](#)
  - protecting TSO account numbers [507](#)
  - RACF variable example [219](#)
  - SETOPTS RACLIST processing [509](#)
  - UACC authorities [508](#)
- ACEECHK
  - activity checking [297](#)
  - exception list processing [297](#)
- ACEECHK class
  - description [683](#)
- achieving system security after the first IPL with RACF installed [356](#)
- ACICSPCT class
  - description [686](#)
- activating
  - program control [132](#)
  - SETOPTS RACLIST processing
    - for TSO general resource classes [509](#)
  - system-wide RACF options with SETOPTS command [105](#)
  - tape data set protection [123](#)
  - tape volume protection [123](#)
  - TSO general resource classes [508](#)
- ADDCREATOR options [163](#)
- ADDMEM operand
  - RALTER command
    - for global access checking table [197](#)
  - RDEFINE command
    - protecting terminals with a GTERMINL profile [226](#)
- ADDUSER command
  - NOPASSWORD option [74](#)
  - NOPHRASE option [74](#)
  - RESTRICTED option [75](#)
- administration
  - delegating tasks [9](#)
  - RACF commands for group [14](#)
  - RACF commands for user [13](#)
  - sharing responsibilities [92](#)
  - using groups to allow flexibility [85](#)
- administration, RACF
  - classroom courses [xxvii](#)
- administrative control
  - allowed by RACF [7](#)
- administrative group
  - defining [85](#)
- ADMINISTRATOR option
  - DFSMSdss commands [168](#)
- ADSP (automatic data set protection) attribute
  - bypassing system-wide [122](#)
  - description of [16](#), [62](#)
- ADSP (automatic data set protection) attribute (*continued*)
  - limitation on assigning [66](#)
  - planning for the use of [36](#)
  - to protect data set with discrete profile [150](#)
  - when protecting tape data sets [172](#)
  - when RACF is deactivated [512](#)
- AIMS class
  - description [689](#)
- ALCSAUTH class
  - description [683](#)
- algorithm, masking
  - PassTicket key [286](#)
- ALLOCATE command (TSO)
  - to protect data set with discrete profile [151](#)
  - using the PROTECT operand on [162](#)
  - using the SECMODEL operand on [162](#)
- allocation of devices
  - controlling with DEVICES profiles [232](#)
- ALLOWCONTAIN operand [81](#)
- ALTER access authority
  - as related to RACF commands [703](#)
  - for general resources [193](#)
  - for RACF-protected tape volume labels [182](#)
  - restricting its ability to manage discrete profiles [259](#)
  - to a tape volume [179](#)
- ALTUSER command
  - ALLOWCONTAIN option [81](#)
  - CONTAIN option [81](#)
  - NEVERCONTAIN option [81](#)
  - NOCONTAIN option [81](#)
  - NOPASSWORD option [74](#)
  - NOPHRASE option [74](#)
  - RESTRICTED option [75](#)
- APPC (advanced program-to-program communication)
  - protecting with RACF [248](#)
- APPC application
  - defining PTKTDATA profiles [284](#)
- APPC transaction program
  - protecting with APPCTP profiles [249](#)
  - WORKATTR segments [55](#)
- APPCLU class
  - activating for VTAM LU 6.2 bind [222](#)
  - conversation security options [250](#)
  - defining for VTAM LU 6.2 bind [221](#)
  - description [683](#)
  - example for VTAM LU 6.2 bind [222](#)
  - granting access [222](#)
  - SESSION segment [221](#)
  - SESSION segment fields [209](#)
  - SETOPTS RACLIST not used [222](#)
  - VTAM session interval [132](#)
- APPCPORT class
  - authorization checking [731](#)
  - conditional access [157](#), [194](#)
  - description [683](#)
  - protecting the port of entry (POE) [248](#)
- APPCSERV class
  - authorizing APPC servers [250](#)
  - description [683](#)
- APPCSI class
  - description [683](#)
  - protecting CPI-C side information [249](#)
- APPCTP class
  - description [683](#)

APPCTP class (*continued*)

protecting APPC transaction programs [249](#)

APPL class

activating [223](#)

authorization checking [733](#)

controlling LU attach request [249](#)

description [683](#)

protecting applications [223](#)

protecting conversations between partner LUs [250](#)

SETROPTS RACLIST processing [223](#)

application

authorizing access to a RACF-protected [733](#)

authorizing for SAF callable services [603](#)

protecting with APPL profiles [223](#)

application identity mapping [524](#)

application name

defining PTKTDATA profiles [286](#)

application updates

controlling automatic direction of [440](#)

assigning

access authorities to DASD data sets [165](#)

group as owner of RACF profile [91](#)

group authorities [92](#)

optional user attributes [14](#)

ownership of data set profile [150](#)

ownership of RACF group [90](#)

ownership of resource profile [20](#)

user attributes [66](#)

assistive technologies [741](#)

associations

user ID

approving [410](#)

defining [409](#)

defining for other users [410](#)

defining for your user ID [410](#)

deleting [411](#)

listing [411](#)

managed [410](#)

AT operand

controlling use of [436](#)

directing commands [412](#)

attribute

ADSP (automatic data set protection)

bypassing system-wide [122](#)

description of [16](#), [62](#)

limitation on assigning [66](#)

assigning user or group [14](#)

AUDITOR

description of [16](#), [58](#)

listing users with [67](#)

suggestions for assigning [66](#)

checking user's group-related [111](#)

CLAUTH (class authority)

description of [16](#), [60](#)

definition of user and group [5](#)

group-AUDITOR

description of [16](#), [58](#)

scope of authority [63](#)

group-OPERATIONS

compared to DASDVOL authority [60](#)

compared to DFSMSdss authorization [60](#), [168](#)

description of [16](#), [60](#)

scope of authority [63](#)

group-SPECIAL

attribute (*continued*)

group-SPECIAL (*continued*)

description of [15](#), [58](#)

illustration of scope of authority [63](#)

scope of authority [63](#)

GRPACC (group access)

description of [16](#), [61](#)

OPERATIONS

compared to DASDVOL authority [60](#)

compared to DFSMSdss authorization [60](#), [168](#)

description of [16](#), [59](#)

listing users with [67](#)

suggestions for assigning [66](#)

RESTRICTED

description of [62](#)

REVOKE

description of [61](#)

listing users with [67](#)

ROAUDIT

description of [16](#), [58](#)

scope of control of group-level [14](#)

SPECIAL

description of [15](#), [57](#)

listing users with [67](#)

suggestions for assigning [66](#)

summary [699](#)

UNIVERSAL, for groups [88](#)

user

assigning at the group level [62](#)

verifying

with DSMON reports [67](#)

AUDIT operand

using for general resource profiles [186](#)

audit record

debugging your security arrangements [716](#), [717](#)

auditing

message traffic [248](#)

messages sent with TSO SEND, LISTBC, or TPUT [248](#)

unknown operator command [241](#)

auditing information

authority to display [58](#)

auditor

defining

example [359](#)

duties of [10](#)

responsibilities during implementation planning [34](#)

AUDITOR attribute

as related to RACF commands [700](#)

description of [16](#), [58](#)

listing users with [67](#)

suggestions for assigning [66](#)

authentication

of passwords

exit routines for [22](#)

authentication, user ID

brief description [738](#)

authority

checking after restarting a job [365](#)

DASDVOL

compared to DFSMSdss authorization [168](#)

DASDVOL and DFSMSdss authorization [168](#)

definition of group [5](#)

delegation to group authority [92](#)

limits at the group level [63](#)

authority (*continued*)

- of auditor [58](#)
- of CLAUTH (class authority) user [60](#)
- of OPERATIONS user [59](#)
- of profile owner to access data set [150](#)
- of read-only auditor [58](#)
- of started procedure to access resources [141](#)
- of user with group-level attribute [62](#)
- of users and groups to use terminals [227](#)
- OPERATIONS and DASDVOL [60](#)
- OPERATIONS and DFSMSdss authorization [60](#), [168](#)
- required to access terminals [730](#)
- required to create a data set [149](#)
- required to issue RACF commands [693](#)
- required to open a nonlabeled tape [183](#)
- required to perform catalog operations on a protected VSAM catalog [166](#)
- required to refresh global access checking lists [130](#)
- required to refresh in-storage lists [130](#)
- required to run DSMON program [58](#)
- requirements for tape labels [182](#)
- requirements for TAPEVOL and TAPEDSN options [179](#)
- scope of for user with group-level attributes [63](#)
- structure at group level [63](#)
- summary [693](#)
- summary of authorities and commands [699](#)
- to access a general resource [192](#)
- to access applications [733](#)
- to access consoles [229](#)
- to access DASD data sets [163](#)
- to access data set when not in access list [157](#)
- to access protected tape data sets [170](#)
- to access protected tape volumes [170](#)
- to access tape volume profiles [173](#)
- to assign or revoke the ADSP attribute [62](#)
- to assign the GRPACC attribute [62](#)
- to assign the RESTRICTED attribute [62](#)
- to create profiles [19](#)
- to modify generic profiles [156](#)
- to modify or delete profiles [20](#)
- to revoke a user from system [61](#)
- to work with profiles through attributes at group level [63](#)
- to write to a tape data set or tape volume [178](#)
- when owning a RACF group [90](#)
- when owning a user profile [57](#)

authority for TSO user

- protecting [507](#)

authorization checking

- bypassing for general resource classes [114](#)
- CICS transactions [732](#)
- description [2](#)
- for a tape data set [173](#)
- for access to protected consoles [731](#)
- for access to protected IP addresses [731](#)
- for access to protected JES input devices [731](#)
- for access to protected terminals [730](#)
- for fields in a RACF profile [200](#), [510](#)
- for multivolume tape data sets [181](#)
- for protected SMS classes [501](#)
- for RACF-protected resources [718](#)
- for security labels [733](#)
- for TSO resources [510](#)
- repeating when restarting a job [365](#)
- SAF router exits [719](#)

authorization checking (*continued*)

- specifying for general resource classes [114](#)
- using exits to performing additional [22](#)

authorized access attempts

- logging [2](#)

authorized caller table report

- from DSMON [363](#)

authorizing

- another user to submit jobs for you [455](#)
- controlling with DEVICES profiles [494](#)
- inbound work (JES) [468](#)
- outbound work [483](#)
- use of printers [494](#)

authorizing only for RACF-defined users

- access to resources [366](#)

automatic assignment of unique UNIX identities

- overview [518](#)
- through UNIX services [520](#)
- using RACF commands [518](#)

automatic direction

- of application updates
- controlling [440](#)
- of commands
- controlling [437](#)
- of password phrases
- controlling [439](#)
- of passwords
- controlling [439](#)
- order considerations [416](#)

automatic direction of commands [417](#)

automatic password direction [433](#)

automatic TVTOC tape volume profile [176](#)

**B**

backup RACF database [368](#)

base name profile

- access list in GDG [161](#)

base segment

- group profile [86](#)
- user profile [47](#)

BASIC attribute for programs

- defining [317](#)
- overview [316](#)

BASIC program security mode

- maintaining a clean environment [306](#)
- migrating to ENHANCED mode [308](#)
- more complex controls [307](#)
- program control by SMFID [305](#)
- program control for SERVAUTH [314](#)
- simple program protection [303](#)
- using execute control [307](#)
- using program access to data sets (PADS) [310](#)

batch job

- authority to access a data set [157](#)
- defining PTKTDATA profiles [284](#)
- preventing security exposures for [366](#)
- preventing unauthorized users from running [454](#)
- user identification with USER operand [367](#)

batch monitor

- running jobs under execution batch monitor [454](#)

batch user

- assigning user ID to [38](#)

BCICSPCT class



- BCICSPCT class (*continued*)
  - description [686](#)
- benefits of using RACF groups [89](#)
- bind profile, LDAP [617](#)
- bind, password on
  - RACF support for [221](#)
- BLKUPD command
  - overview [25](#)
- block update command [25](#)
- BPX.SERVER [607](#), [608](#)
- BPX.UNIQUE.USER profile [519](#), [520](#)
- BPXPRMxx
  - setting user limits [515](#)
- bypass label processing (BLP)
  - authorizing with FACILITY profile [182](#)
- bypass password protection
  - DSMON report [362](#)
  - use by callers of RACF [365](#)
- bypassing
  - ADSP attribute [122](#)
  - authorization checking for general resource classes [114](#)
  - password protection [19](#)
  - password protection of tape data sets [170](#)
  - password protection of tape volumes [170](#)
  - protection of data sets when DASDVOL class is active [167](#)
  - RACF protection of a system data set during system access [166](#)
  - write-enable protection for tapes [180](#)
- bypassing PassTicket replay protection [290](#)

## C

- CACHECLS class
  - description [683](#)
- callable services, authorizing applications that invoke [603](#)
- CANCEL command (TSO)
  - controlling job cancellation [464](#)
- cancelling jobs
  - controlling [464](#)
- capturing output
  - of RACF TSO commands [413](#)
  - produced by password synchronization [409](#)
- capturing the output of RACF commands [26](#)
- catalog
  - assigning access authority to [165](#)
  - master catalog
    - global access checking table entries [198](#)
  - password and RACF authorization requirements for catalog operations [165](#)
  - protecting [166](#)
  - protecting with FACILITY profiles [166](#)
  - user catalog
    - global access checking table entries [198](#)
- CATDSNS operand
  - SETROPTS command [120](#)
- categories
  - maintaining in an RRSF environment [97](#)
- CBIND class
  - description [683](#)
- CCA (common cryptographic architecture) [287](#)
- CCICSCMD class
  - description [686](#)
- CDT class

- CDT class (*continued*)
  - description [683](#)
- CDTINFO
  - field-level access checking [201](#)
- CDTINFO segment
  - user profile
    - FIELD profile names [205](#)
- certificate authorities [539](#)
- certificate name filters [571](#)
- CFDEF
  - field-level access checking [201](#)
- CFDEF segment
  - user and group profile
    - FIELD profile names [205](#)
- CFIELD class
  - description [683](#)
  - profile names [638](#)
- CHOWN.UNRESTRICTED profile [528](#)
- CICS
  - general resource classes [686](#)
  - using with RACF [250](#)
- CICS application
  - defining a PTKTDATA profile [284](#)
- CICS segment
  - description [18](#)
  - field-level access checking [201](#)
  - user profile
    - contents of [48](#)
    - FIELD profile names [205](#)
- CICS signon table
  - deleting user entry [78](#)
- CICS transaction
  - authorization checking [732](#)
- CIMS class
  - description [689](#)
- class descriptor table (CDT)
  - adding installation-defined classes
    - details [263](#)
    - overview [19](#)
  - dynamic CDT classes [263](#)
  - obtaining security report [363](#)
  - supplied classes for z/OS systems [683](#)
- class descriptor table report
  - from DSMON [363](#)
- CLASSACT operand
  - SETROPTS command [114](#)
- CLASSACT(\*) operand
  - SETROPTS command
    - guidelines [105](#), [114](#)
- classes
  - DCE
    - KEYSMSTR [255](#)
  - defining
    - overview [19](#)
  - KEYSMSTR
    - activating [256](#)
    - defining [256](#)
  - names of supplied general resource classes for z/OS [683](#)
- RRSFDATA
  - controlling access to RACLINK command [435](#)
  - controlling access to RRSF functions [434](#)
  - controlling automatic direction [437](#)

- classes (*continued*)
  - RRSFDATA (*continued*)
    - controlling password and password phrase synchronization [435](#)
    - controlling use of AT operand [436](#)
    - controlling use of RACLINK DEFINE operand [435](#)
    - controlling use of RACLINK PWSYNC operand [435](#)
  - STARTED
    - and STDATA segment [142](#)
    - setting up [142](#)
    - STARTED profile names [143](#)
  - UNIXMAP
    - profiles for [526](#)
    - z/OS UNIX considerations [524](#)
  - z/OS UNIX event auditing
    - DIRACC [537](#)
    - DIRSRCH [537](#)
    - FSOBJ [537](#)
    - FSSEC [537](#)
    - IPCOBJ [537](#)
    - PROCACT [537](#)
    - PROCESS [537](#)
- classifying
  - data [95](#)
  - users [95](#)
- classroom courses, RACF [xxvii](#)
- CLAUTH (class authority) attribute
  - as related to RACF commands [701](#)
  - assigning for TAPEVOL class [172](#)
  - delegating [61](#)
  - description of [16](#), [60](#)
  - example for JESSPOOL class [486](#)
  - FACILITY class [211](#)
  - with JOIN group authority [92](#)
- clean environment [306](#)
- client/server environment
  - PassTicket [281](#)
- CLIST
  - creating user IDs with [56](#)
- command authorization
  - in an MCS sysplex [240](#)
- command direction
  - AT option [412](#)
  - on local node [412](#)
  - on remote node [413](#)
  - ONLYAT option [415](#)
  - order considerations [416](#)
- commands
  - AT operand
    - controlling use of [436](#)
  - automatic direction [417](#)
  - controlling automatic direction of [437](#)
  - ONLYAT operand
    - controlling use of [437](#)
  - output capturing [413](#)
  - RACDCERT
    - controlling access to [549](#)
  - RACLINK
    - controlling access to [435](#)
    - controlling use of DEFINE operand [435](#)
    - controlling use of PWSYNC operand [435](#)
  - START [143](#)
  - summary [693](#)
- communications device
  - controlling allocation with DEVICES profiles [232](#)
- COMPATMODE operand
  - SETROPTS command [135](#)
- conditional access
  - CONSOLE class [157](#), [193](#)
  - MDSNTB class [194](#)
  - OPERCMDs class [194](#), [241](#)
- conditional access list
  - data set profiles [157](#)
  - during authorization checking [722](#)
  - general resource profiles [193](#)
- CONNECT command
  - using to specify attributes at the group level [62](#)
- connect group
  - specifying on TSO LOGON command [511](#)
  - using current connect group checking [112](#)
- CONNECT group authority
  - as related to RACF commands [702](#)
  - description [91](#)
- console
  - access authorities for [229](#)
  - conditional access [157](#), [193](#)
  - creating a RACF user profile for [241](#)
  - extended MCS
    - OPERPARM segment in user profiles [52](#)
  - protecting [228](#)
  - protecting with SECLABEL profiles [229](#)
  - RACF authorization checking for [731](#)
- CONSOLE class
  - activating [229](#)
  - and SECLABEL class [229](#)
  - authorization checking [731](#)
  - conditional access [157](#), [193](#)
  - description [683](#)
  - LOGON [491](#)
  - protecting MCS consoles [228](#)
  - SETROPTS RACLIST processing [229](#)
  - UACC authorities [229](#)
- CONSOLE command (TSO)
  - and OPERPARM segment [52](#)
- contact
  - z/OS [741](#)
- CONTAIN operand [81](#)
- CONTAINED attribute
  - defining [81](#)
- contained user IDs [81](#)
- CONTROL access authority
  - as related to RACF commands [703](#)
  - for general resources [193](#)
- controlled program
  - example of command to define [324](#)
- conversation security options
  - SESSION segment of APPCLU profile [250](#)
- conversion utility, IRRIRA00 [524](#)
- converting masked keys [288](#)
- CONVSEC field
  - APPCLU profile [250](#)
- coordinating profile updates [355](#)
- courses about RACF [xxvii](#)
- CPI-C side information
  - protecting with APPCSI profiles [249](#)
- CPSMOBJ class
  - description [687](#)

- CPSMXMP class
  - description [687](#)
- CPU-ID field
  - SMF CONTROL file [286](#)
- CREATE group authority
  - allows OPERATIONS user to define data set profiles [59](#)
  - as related to RACF commands [702](#)
  - description [91](#)
  - for protecting data sets in group [147](#)
- creating
  - a new data set
    - authority required [149](#)
  - access list for field-level access checking [202](#)
  - TVTOC (tape volume table of contents) [175](#)
  - user data sets [148](#)
- cryptographic product
  - requirements for PassTicket [287](#)
- CRYPTOZ class
  - brief description [689](#)
- CSDATA
  - field-level access checking [201](#)
- CSDATA segment
  - adding custom field data [643](#)
  - authorizing users to update data [645](#)
  - group profile
    - contents of [87](#)
  - user and group profile
    - FIELD profile names [205](#)
  - user profile
    - contents of [49](#)
    - IRRDBU00 utility [372](#)
- CSFKEYS class
  - description [689](#)
- CSFSERV class
  - description [689](#)
- CSVLLA prefix
  - FACILITY profile [234](#)
- current connect group checking
  - activating [112](#)
  - deactivating [112](#)
- custom fields
  - activating [642](#)
  - adding data to CSDATA segment [643](#)
  - authorizing users to define [644](#), [646](#)
  - authorizing users to update CSDATA [645](#)
  - changing attributes [647](#)
  - common errors [651](#)
  - defining [638](#)
  - overview [637](#)
  - profiles in the CFIELD class [638](#)
  - removing [650](#)
  - RRSF considerations [653](#)
  - task roadmap [638](#)

## D

- DADSM scratch function
  - DASDVOL authority [167](#)
- daemon access to delegated resources [257](#)
- DASD data set
  - access authorities for [163](#)
  - DFP-managed
    - preventing access to [120](#)
  - erasing of scratched [165](#)

- DASD data set (*continued*)
  - multivolume considerations [163](#)
  - protecting [18](#), [147](#), [163](#)
  - protecting with discrete profile [150](#)
  - protecting with discrete profile through ADSP attribute [62](#)
  - protecting with the PROTECT operand on TSO ALLOCATE command [162](#)
  - protecting with the SECMODEL parameter on TSO ALLOCATE command [162](#)
  - scratching when protected with discrete profile [151](#)
- DASD volume
  - authority [167](#)
  - DASDVOL and DFSMSdss authorization [168](#)
  - OPERATIONS and DASDVOL authority [60](#)
  - OPERATIONS and DFSMSdss authorization [60](#), [168](#)
  - RACF authorization checking for [719](#)
  - requirements for RACF databases [368](#)
- DASDVOL class
  - alternatives
    - DFSMSdss authorization [168](#)
  - bypassing protection of individual data sets [167](#)
  - compared to OPERATIONS attribute [60](#), [167](#)
  - description [683](#)
  - OPERATIONS attribute allows access [59](#)
  - recording statistics for [118](#)
- data
  - classifying [95](#)
- data blocks
  - number of resident [370](#)
- data control group
  - defining [85](#)
- data set
  - accessing if critical profile deleted [119](#)
  - activating or deactivating erase-on-scratch processing [124](#)
  - activating tape data set protection [123](#)
  - controlling creation of [149](#)
  - creating with PROTECTALL in effect [119](#)
  - defining automatically with ADSP attribute [16](#)
  - determining owner of SMS-managed [501](#)
  - dumping and restoring on a DASD volume [167](#)
  - for which RACF protection is bypassed during system access [166](#)
  - for which RACF protection is enforced during system access [166](#)
  - generic profile checking [153](#)
  - listing all invalid user IDs with access [382](#)
  - logging real data set names [122](#)
  - option for protecting all [119](#)
  - overview of RACF protection [35](#)
  - ownership of profile [150](#)
  - password-protected [160](#)
  - protecting data set that has single-qualifier name [148](#)
  - protecting duplicate-named residing on different volumes [161](#)
  - protecting for a group [149](#)
  - protecting GDG [161](#)
  - protecting non-VSAM with the JCL PROTECT parameter [162](#)
  - protecting non-VSAM with the JCL SECMODEL parameter [162](#)
  - protecting single-qualifier named data sets [122](#)
  - protecting with a dummy group ID [150](#)

## data set *(continued)*

- protecting with discrete profile [150](#)
- protecting with discrete profile through ADSP attribute [62](#)
- protecting with generic profile [151](#)
- protecting with security category and security level [97](#)
- RACF authorization checking for [719](#)
- RACF authorization checking for user's own [721](#)
- recovering if critical profile deleted [119](#)
- security classification of [21](#), [95](#)
- system temporary data sets
  - preventing access to [120](#)
- system-wide modeling options [121](#)
- table-driven naming conventions [147](#)
- transferring to another system [156](#)
- types that RACF can protect [18](#)
- uncataloged permanent data sets
  - preventing access to [120](#)
- using a group to control [85](#)
- using discrete profiles to protect multivolume [162](#)
- using profile modeling when protecting [36](#)
- z/OS UNIX considerations for protecting files [530](#)
- z/OS UNIX file, considerations for [530](#)

## data set profile

- authority granted through group-level attributes [63](#)
- authority of OPERATIONS user over [59](#)
- conditional access list [157](#)
- controlling access to DFP segment [503](#)
- controlling who can create [148](#)
- default UACC for [157](#)
- default UACC when connected to a group [67](#)
- defining to protect program library [324](#)
- DFP segment [500](#)
- high-level qualifier of profile name [147](#)
- ownership of [150](#)
- preventing duplicate-named [162](#)
- protecting multivolume data set with discrete [162](#)
- protecting program dumps [230](#)
- rules for defining [147](#)
- sharing a common [162](#)
- when changes take effect [717](#)

## database

- sharing data between remote systems [370](#)

## database profiles

- synchronizing [434](#)

## database, Db2

- for IRRDBU00 output [382](#)

## DATASET class

- bypassing recording of statistics [118](#)
- OPERATIONS attribute allows access [59](#)
- recording statistics for [118](#)

## days of week

- terminal can access system [73](#), [227](#)
- user can access system [73](#)

## Db2

- access control module [251](#)
- creating a Db2 table space for IRRDBU00 output [383](#)
- creating Db2 indexes for IRRDBU00 output [383](#)
- creating Db2 tables for IRRDBU00 output [383](#)
- creating optimization statistics for the Db2 database [385](#)
- database for IRRDBU00 output [382](#)
- Db2 utility statements required to delete the group records [385](#)

## Db2 *(continued)*

- deleting the IRRDBU00 data from the Db2 database [385](#)
  - general resource classes [687](#)
  - loading the Db2 tables for IRRDBU00 output [384](#)
  - reorganizing the unloaded RACF data in the Db2 database [384](#)
  - SQL utility statements
    - for creating a table for IRRDBU00 output [383](#)
    - for creating indexes for IRRDBU00 output [383](#)
    - for defining a table space [383](#)
  - table names provided in SYS1.SAMPLIB [385](#)
  - using with IRRDBU00 output [382](#)
  - using with RACF [251](#)
  - utility statements required to load the tables with IRRDBU00 output [384](#)
- ## Db2 load utility
- sample statements for IRRDBU00 output [382](#)
- ## Db2 queries
- sample statements for IRRDBU00 output [382](#)
- ## DBSYNC EXEC [434](#)
- ## DCE
- general resource class [688](#)
  - KEYSMSTR class [255](#)
- ## DCE segment
- field-level access checking [201](#)
  - user profile
    - contents of [49](#)
    - FIELD profile names [205](#)
- ## DCE.PASSWORD.KEY [607](#)
- ## DCICSDCT class
- description [687](#)
- ## DD statements (JCL)
- parameters related to RACF [364](#)
- ## ddname
- for IRRDBU00 input data sets [374](#)
- ## deactivating
- SETROPTS GENLIST processing [126](#)
  - SETROPTS RACLIST processing [128](#)
  - unused user ID [111](#)
- ## deadlocks
- avoiding [231](#)
- ## debugging
- problems with profile definitions [715](#)
- ## default group
- assigning a user to [5](#)
  - connecting a user to [62](#)
- ## default NJE user ID
- ownership of SYSOUT based on NODES profiles [476](#)
  - specifying with SETROPTS command [481](#)
- ## default user IDs
- concepts [480](#)
  - description [480](#)
- ## DEFER mode
- logging access attempts when operating in [3](#)
- ## DEFINE operands
- RACLINK command
    - controlling use of [435](#)
- ## defining
- general resources
    - summary of steps [185](#)
  - groups [13](#)
  - profiles for field-level access checking of DFP segment [502](#)
  - RACF group

- defining (*continued*)
  - RACF group (*continued*)
    - summary of steps [93](#)
  - resources with CLAUTH authority [60](#)
  - rules for defining data set profiles [147](#)
  - tape volume without a TVTOC [174](#)
  - tape volumes to RACF [170](#)
  - tape volumes with a TVTOC [172](#)
  - user IDs [56](#)
  - users
    - summary of steps [75](#)
    - users to RACF using ICF [76](#)
- delegated resources [257](#)
- delegating
  - administration [79](#)
  - help desk functions [655](#)
  - ownership of FACILITY profiles [211](#)
  - resources [257](#)
- deleting
  - groups
    - summary of steps [94](#)
- DELMEM operand
  - RALTER command
    - for global access checking table [198](#)
- DELUSER processing
  - with distributed identity filter [673](#)
- DES (data encryption standard) algorithm
  - encrypting session keys [222](#)
  - replacing [139](#)
- Device Support Facilities (ICKDSF)
  - DASDVOL authority [167](#)
- DEVICES class
  - activating [234](#), [495](#)
  - controlling access to printers [494](#)
  - controlling device allocation [232](#)
  - defining profiles [495](#)
  - description [683](#)
  - example [234](#)
  - SETROPTS RACLIST processing [234](#)
  - UACC authorities [233](#), [495](#)
- DFP segment
  - data set profile
    - FIELD profile names [206](#)
    - field-level access checking [501](#)
    - RACF processing [501](#)
    - SMS-managed data sets [500](#)
  - description [17](#)
  - field-level access checking [201](#)
  - group profile
    - contents of [87](#)
    - DFSMSdss storage administrator [93](#)
    - field-level access checking [501](#)
    - overriding default values [500](#)
    - RACF processing [501](#)
    - SMS-managed data sets [499](#)
  - user profile
    - contents of [49](#)
    - field-level access checking [501](#)
    - overriding default values [500](#)
    - RACF processing [501](#)
    - SMS-managed data sets [499](#)
- DFP-managed temporary data sets
  - protecting with TEMPDSN profiles [224](#)
- DFSMSdfp
  - DFSMSdfp (*continued*)
    - general resource classes [690](#)
    - RACF support for [497](#)
  - DFSMSdss
    - ADMINISTRATOR option [168](#)
    - and OPERATIONS attribute [59](#)
    - authorized storage administration
      - authorizing with FACILITY profiles [168](#)
      - compared to OPERATIONS attribute [60](#), [168](#)
      - description [168](#)
    - DASDVOL authority [167](#)
  - DFSMSshm
    - considerations for tape data sets [180](#)
  - DFSORT ICETOOL [375](#)
  - digital certificate name filters [571](#)
  - digital certificates
    - administering through initACEE callable service (IRRSIA00) [604](#)
    - administering through R\_datalib callable service (IRRSDL00 or IRRSDL64) [606](#)
    - administering with the RACDCERT command [548](#)
    - and WebSphere Application Server [566](#)
    - applications that administer [604](#), [606](#)
    - automatic registration using WebSphere Application Server [582](#)
    - excluding [578](#)
    - expiring [584](#)
    - exporting certificates using R\_PKIServ callable service [608](#)
    - extracting private keys [607](#)
    - generating certificates using R\_PKIServ callable service [608](#)
    - grouped in a key ring [567](#)
    - implementation scenarios [590](#)
    - irrcerta, irrsitec, and irrmulti user IDs [584](#)
    - issuer's X.509 distinguished name [572](#)
    - key rollover [584](#)
    - managing serial numbers [607](#)
    - modeling [577](#)
    - NOTRUST option [578](#)
    - overview [539](#)
    - RACF private key size restrictions [547](#)
    - rekeying [584](#)
    - renewing [584](#)
    - retrieving certificates using R\_PKIServ callable service [608](#)
    - rollover [584](#)
    - sharing with multiple servers [568](#)
    - storing in Integrated Cryptographic Service Facility (ICSF) [582](#)
    - subject's X.509 distinguished name [572](#)
    - supplied [589](#)
    - using the DIGTCERT class [566](#)
    - using the DIGTCRIT class [579](#)
    - X.500 directory information tree [571](#)
  - digital signing of load modules
    - overview [327](#), [344](#)
  - digital signing of programs
    - overview [327](#)
    - task roadmap [330](#)
  - digital verification of program signatures
    - task roadmap [338](#)
  - DIGTCERT class
    - description [683](#)

- DIGTCERT class (*continued*)
  - profile name [565](#)
  - profile ownership [566](#)
  - RACLISTing [566](#)
- DIGTCRIT class
  - activating additional criteria [581](#)
  - description [684](#)
  - RACLISTing [579](#)
  - using application criteria [579](#)
- DIGTNMAP class
  - activating certificate name filtering [573](#)
  - description [684](#)
  - profile [573](#)
- DIGTRING class
  - description [684](#)
  - profile name [568](#)
- DIMS class
  - description [689](#)
- DIRACC class
  - description [691](#)
  - z/OS UNIX event auditing [537](#)
- DIRAUTH class
  - activating [248](#), [510](#)
  - auditing all access attempts [248](#)
  - checking security labels for messages [246](#)
  - description [684](#)
- directed command
  - order considerations [416](#)
- DIRSRCH class
  - description [691](#)
  - z/OS UNIX event auditing [537](#)
- discrete profile
  - authority to create [19](#)
  - authority to modify or delete [20](#)
  - building for tape data set and tape volume [171](#)
  - creating for data set through ADSP attribute [62](#)
  - creating to control access to specific field in TSO segment [202](#)
  - creating to protect non-VSAM data set with JCL PROTECT parameter [162](#)
  - creating to protect non-VSAM data set with JCL SECMODEL parameter [162](#)
  - creating with ADSP attribute [16](#)
  - defining to protect GDG data set [161](#)
  - defining with PROTECT parameter in JCL [181](#)
  - description of [18](#)
  - disposition of when scratching a DASD data set [167](#)
  - protecting data sets with [150](#)
  - protecting duplicate-named data sets with [161](#)
  - protecting multivolume data sets with [162](#)
  - user with GRPACC attribute [16](#)
- discretionary access control (DAC)
  - definition [9](#)
- displaying
  - auditing information [58](#)
  - information from RACF profiles [26](#)
  - user IDs or group names in RACF database [24](#)
- distributed identity filter
  - DELUSER processing [673](#)
  - IDIDMAP profiles [672](#)
  - overview [671](#)
  - residual IDIDMAP profiles [673](#)
- distributed identity filter (*continued*)
  - system requirements [671](#)
  - user profile updates [672](#)
  - using the RACMAP command [671](#)
- DLF (data lookaside facility)
  - controlling access to DLF objects [235](#)
- DLF installation exit [236](#)
- DLF object
  - protecting with DLFCLASS profiles [175](#), [235](#)
- DLFCLASS class
  - activating [237](#)
  - defining profiles [236](#)
  - description [684](#)
  - DLFDATA segment [235](#)
  - example [237](#)
  - FIELD profile names [206](#)
  - granting access to profiles [237](#)
  - protecting DLF objects [235](#)
  - SETROPTS RACLIST processing [237](#)
  - UACC authorities [236](#)
- DLFDATA segment
  - DLFCLASS profile
    - contents [235](#)
    - FIELD profile names [206](#)
    - field-level access checking [201](#)
- DPAGELBL parameter
  - on OUTPUT statement in JCL [365](#)
- DSMON (data security monitor)
  - AUDITOR attribute [58](#)
  - authorized caller table report [363](#)
  - class descriptor table report [363](#)
  - global access checking table report [363](#)
  - group tree report [63](#), [362](#)
  - list of reports produced [361](#)
  - program properties table (PPT) report [362](#)
  - protecting with PROGRAM profiles [25](#), [58](#)
  - RACF exits report [363](#)
  - ROAUDIT attribute [58](#)
  - selected data sets report [364](#)
  - selected user attribute report [363](#)
  - selected user attribute summary report [364](#)
  - started procedures table report [363](#)
  - system report [362](#)
  - using [25](#), [361](#)
  - verifying
    - user attributes [67](#)
- DSNADM class
  - description [687](#)
- DSNAME parameter
  - on DD statement in JCL [365](#)
- DSNR class
  - description [687](#)
- DSNRAUTH class
  - description [687](#)
- dumps
  - non-RACF methods for controlling access to program [232](#)
  - protecting with data set profiles [230](#)
  - protecting with FACILITY profiles [230](#)
  - restriction on
    - with EXECUTE authority [230](#)
- dynamic class descriptor table (CDT)
  - attribute comparisons [277](#), [278](#)
  - CDT class profiles [264](#)



- dynamic class descriptor table (CDT) (*continued*)
  - changing a POSIT value [269](#)
  - changing CDT entries [270](#)
  - classes with GENERIC(DISALLOWED) [272](#)
  - classes with shared POSIT values [267](#)
  - classes with unique POSIT values [265](#)
  - deleting a dynamic class [273](#)
  - disabling [275](#)
  - migration [276](#)
  - overview [263](#)
  - re-enabling a dynamic class [276](#)
  - RRSF considerations [279](#)
  - shared system considerations [278](#)
  - sysplex considerations [278](#)
  - using [264](#)
- dynamic started procedures table and STARTED class [142](#)

## E

- EARLYVERIFY operand
  - SETROPTS command [455](#)
- ECICSDCT class
  - description [687](#)
- EDIT command (TSO)
  - using [367](#)
- EGN operand
  - SETROPTS command [105](#)
- EIM
  - general resource class [688](#)
- EIM segment
  - FIELD profile names [206](#)
  - field-level access checking [201](#)
- EJBROLE class
  - description [688](#)
- encoding
  - definition of [139](#)
- encrypting
  - APPCLU session key [222](#)
  - PassTicket key [287](#)
  - RACF VSAM data set [369](#)
- ENCRYPTKEY value
  - RALTER command [288](#)
- end-user function [608](#)
- enhanced generic naming
  - DATASET class
    - activating or deactivating [121](#)
    - when installing RACF for the first time [105](#)
- ENHANCED program security mode
  - maintaining a clean environment [306](#)
  - migrating from BASIC mode [308](#)
  - overview [315](#)
  - program control by SMFID [305](#)
  - program control for SERVAUTH [314](#)
  - simple program protection [303](#)
  - using execute control [316](#)
  - using program access to data sets (PADS) [315](#)
- ENHANCED-WARNING program security mode [308](#)
- Enterprise Identity Mapping
  - general resource class [688](#)
- Enterprise Java Beans
  - general resource classes [688](#)
- enveloping, passwords [623](#)
- erase-on-scratch processing

- erase-on-scratch processing (*continued*)
  - activating or deactivating system-wide [124](#)
  - controlling [165](#)
- errors
  - PassTicket
    - preventing [293](#)
- event notification for LDAP changes [618](#)
- events
  - z/OS UNIX auditing [537](#)
- EXECs
  - DBSYNC [434](#)
- EXECUTE
  - restriction on dumps [230](#)
- EXECUTE authority
  - restriction on dumps [230](#)
- execute-controlled library
  - example of setting up [324](#)
  - in ENHANCED program security mode [316](#)
  - processing [321](#)
- execution batch monitor
  - running jobs under [454](#)
- exit routine
  - list of all defined [363](#)
  - password authentication [22](#)
  - REQUEST=LIST exit
    - resource group profiles [214](#)
    - using to tailor RACF [21](#)
- EXPDT operand of JCL statement
  - to specify security retention period for data set [178](#)
- expiring digital certificate [584](#)
- EXPORT
  - accesses required [610](#)
- extended MCS console
  - OPERPARM segment in user profiles [52](#)
- extending a private key [585](#)
- external writer
  - access to JESSPOOL profiles [484](#)
  - access to spool data sets [484](#)

## F

- FACILITY class
  - activating (first profile) [211](#)
  - activating (LLA-managed data sets) [234](#)
  - activating (NJE node) [493](#)
  - activating (operator commands) [496](#)
  - activating (program dumps) [231](#)
  - activating (RJE workstation) [493](#)
  - activating (vector facility) [229](#)
  - authorizing DFSMSdss administration [168](#)
  - bypass label processing for tapes [182](#)
  - CLAUTH attribute [211](#)
  - delegating profile ownership [211](#)
  - description [684](#)
  - ICHLBP profile [182](#)
  - ICHUNCAT.data-set-name profile [120](#)
  - IEAABD.DMPAKEY profile [230](#)
  - IEAABD.DMPAUTH profile [230](#)
  - IEAVECTOR profile [229](#)
  - IEC.TAPERING profile [180](#)
  - IRR.DIGTCERT [46](#), [549](#)
  - IRR.LISTUSER [655](#)

## FACILITY class *(continued)*

- IRR.LU.EXCLUDE [659](#)
- IRR.LU.OWNER [657](#)
- IRR.LU.TREE [658](#)
- IRR.PASSWORD.RESET [662](#)
- IRR.PWRESET.EXCLUDE [666](#)
- IRR.PWRESET.OWNER [663](#)
- IRR.PWRESET.TREE [664](#)
- planning profiles [211](#)
- protecting catalogs [166](#)
- protecting LLA-managed data sets [234](#)
- protecting NJE nodes [493](#)
- protecting program dumps [230](#)
- protecting RJE workstations [492](#)
- protecting vector facilities [229](#)
- SETROPTS RACLIST processing [211](#), [231](#)
- UACC authorities
  - LLA-managed data sets [234](#)

## FACILITY class resource

- IRR.RPKISERV.PKIADMIN [611](#)

## failsoft processing [367](#)

## fast authorization checking [732](#)

## FCICSFCT class

- description [687](#)

## FIELD class

- activating [202](#)
- activating (DFP segment) [501](#)
- description [510](#), [684](#)
- example of command to permit access to profile [202](#)
- protecting DFP segment fields [501](#)
- protecting fields in RACF profiles [200](#)
- protecting OMVS segment fields [52](#), [88](#)
- SETROPTS RACLIST processing [202](#)

## FIELD profile names

- CDTINFO segment of general profile [205](#)
- CFDEF segment [205](#)
- CICS segment of user profile [205](#)
- CSDATA segment [205](#)
- DCE segment of user profile [205](#)
- DFP segment of data set profile [206](#)
- DFP segment of group profile [206](#)
- DFP segment of user profile [206](#)
- DLFDATA segment of DLFCLASS profile [206](#)
- EIM segment [206](#)
- ICSF segment [206](#)
- ICTX segment of LDAPBIND profile [206](#)
- KERB segment of REALM profile [207](#)
- KERB segment of user profile [207](#)
- LANGUAGE segment of user profile [207](#)
- LNOTES segment of user profile [207](#)
- MFA segment in MFADEF class profiles [207](#)
- NDS segment of user profile [207](#)
- NETVIEW segment of user profile [207](#)
- OMVS segment of group profile [208](#)
- OMVS segment of user profile [208](#)
- OPERPARM segment of user profile [208](#)
- OVM segment of group profile [208](#)
- OVM segment of user profile [208](#)
- PROXY segment of FACILITY profile [209](#)
- PROXY segment of user profile [209](#)
- SESSION segment of APPCLU profile [209](#)
- SIGVER segment of PROGRAM profile [209](#)
- SSIGNON segment of PTKTDATA profile [209](#)
- STDATA segment of STARTED profile [209](#)

## FIELD profile names *(continued)*

- SVFMR segment of general resource profile [209](#)
- TME segment of data set profile [209](#)
- TME segment of general resource profile [210](#)
- TME segment of group profile [209](#)
- TSO segment of user profile [210](#)
- WORKATTR segment of user profile [210](#)

## field-level access checking

- creating access list for [202](#)
- description [200](#)
- DFP segment in data set profile [501](#)
- DFP segment in group profile [501](#)
- DFP segment in user profile [501](#)
- TSO segment of user profile [510](#)
- with FIELD profiles [200](#)

## filters

- for certificate names [571](#)
- for distributed identity names [671](#)

## FIMS class

- description [689](#)

## flushing a VLF cache

- using RACF commands [356](#)

## FSACCESS class

- description [691](#)

## FSEXEC class

- description [691](#)

## FSOBJ class

- description [692](#)
- z/OS UNIX event auditing [537](#)

## FSP (file security packet)

- definition [530](#)

## FSSEC class

- description [692](#)
- z/OS UNIX event auditing [537](#)

## FTP daemon, authorizing access [257](#)

## functional group

- defining [86](#)

## fundamental elements of RACF

- users and groups [13](#)

## G

### GCICSTRN class

- description [687](#)

### GCPSMOBJ class

- description [687](#)

### GCSFKEYS class

- description [689](#)

### GDASDVOL class

- description [684](#)
- OPERATIONS attribute allows access [59](#)

### GDG (generation data group)

- defining generic resource profile
  - with enhanced generic naming active [161](#)
- protecting GDG data sets [161](#)
- security retention period processing for [179](#)

### GDSNBP class

- description [687](#)

### GDSNCL class

- description [687](#)

### GDSNDB class

- description [687](#)



- GDSNGV class
  - description [687](#)
- GDSNJR class
  - description [687](#)
- GDSNPK class
  - description [687](#)
- GDSNPN class
  - description [687](#)
- GDSNSC class
  - description [687](#)
- GDSNSG class
  - description [688](#)
- GDSNSM class
  - description [688](#)
- GDSNSP class
  - description [688](#)
- GDSNSQ class
  - description [688](#)
- GDSNTB class
  - description [688](#)
- GDSNTS class
  - description [688](#)
- GDSNUF class
  - description [688](#)
- GDSNUT class
  - description [688](#)
- GEJBROLE class
  - description [688](#)
- GENCERT
  - accesses required [610](#)
- general resource
  - overview of RACF protection [35](#)
  - protecting [185](#)
  - using profile modeling when protecting [36](#)
- general resource class
  - activating or deactivating [114](#)
  - bypassing recording of statistics [118](#)
  - defining
    - overview [19](#)
  - generic profile checking [190](#)
  - ineligible for global access checking [199](#)
  - product use of
    - CICS [686](#)
    - Db2 [687](#)
    - DCE [688](#)
    - DFSMSdftp [690](#)
    - EIM [688](#)
    - Enterprise Identity Mapping [688](#)
    - Enterprise Java Beans [688](#)
    - IBM MQ [690](#)
    - ICSF [689](#)
    - IMS [689](#)
    - Infoprint Server [689](#)
    - Information/Management [689](#)
    - LFS/ESA [689](#)
    - License Manager [689](#)
    - Lotus Notes for z/OS [690](#)
    - MFA [690](#)
    - miscellaneous [683](#)
    - NetView [690](#)
    - Novell Directory Services for OS/390 [690](#)
    - RACF [683](#)
    - SMS [690](#)
    - Tivoli [691](#)

- general resource class (*continued*)
  - product use of (*continued*)
    - Tivoli Service Desk [689](#)
    - TSO [691](#)
    - WebSphere MQ [691](#)
    - z/OS Integrated Security Services Network Authentication Service [690](#)
    - z/OS UNIX [691](#)
    - z/OSMF [691](#)
  - protecting SMS classes [497](#)
  - recording statistics for [118](#)
  - security classification of [21](#), [95](#)
  - supplied for z/OS [683](#)
  - to protect TSO resources [507](#)
- general resource profile
  - authority granted through group-level attributes [63](#)
  - authority of CLAUTH user to define [60](#)
  - authority of OPERATIONS user over [59](#)
  - conditional access list [193](#)
  - default UACC for [192](#)
  - default UACC when connected to a group [67](#)
  - defining [185](#)
  - sharing in-storage [125](#)
- generic character
  - when defining generic profiles
    - with enhanced generic naming active [152](#), [189](#)
- generic command processing
  - activating or deactivating [115](#)
- GENERIC operand
  - SETROPTS command [115](#)
  - to define generic profile for data set [151](#)
  - to find most specific profile for data set [153](#)
- generic profile
  - authority to create [19](#)
  - authority to modify [156](#)
  - authority to modify or delete [20](#)
  - choosing [188](#)
  - defining for data sets [151](#)
  - defining to protect GDG data sets [161](#)
  - description of [18](#)
  - finding the most specific for a data set [153](#)
  - fully qualified
    - protecting duplicate-named data sets with [161](#)
  - order of checking [153](#)
  - preventing in a dynamic class [272](#)
  - preventing in a static class [188](#)
  - protecting data sets using [151](#)
  - protection while transferring data sets to another system [156](#)
  - rationale for using [7](#), [36](#)
  - rationale for using for data sets [151](#)
  - refreshing in-storage profile lists [130](#)
  - renaming a multivolume data set protected with a [163](#)
  - restricting creation in general resource classes [113](#)
  - top generic profiles for general resource classes [18](#)
  - top profile in JESJOBS class [462](#)
- generic profile checking
  - activating for FIELD general resource class [202](#)
  - activating or deactivating [115](#)
  - effect on performance [192](#)
  - for data sets [153](#)
  - for general resources [190](#)
  - using during failsoft processing [367](#)
  - with ADSP attribute [62](#)

- GENERIC(DATASET) operand
  - SETROPTS command [156](#)
- GENERIC(DISALLOWED)
  - dynamic classes [272](#)
  - sharing a POSIT value [272](#)
- GENERIC=DISALLOWED
  - static classes [188](#)
- GENERICOWNER operand
  - SETROPTS command
    - related to CLAUTH attribute [16](#), [60](#), [77](#), [211](#)
- GENLIST operand
  - SETROPTS command [125](#)
- GENRENEW
  - accesses required [610](#)
- getting started with RACF [356](#)
- GID mapping
  - and VLF [524](#)
  - UNIXMAP class profiles [526](#)
- GIMS class
  - activating [213](#)
  - description [689](#)
- GINFOMAN class
  - description [689](#)
- global access checking
  - activating or deactivating [118](#)
  - creating table entries [195](#)
  - defining an entry for the vector facility [229](#)
  - during authorization checking [720](#)
  - protecting frequently accessed system data sets [166](#)
  - refreshing in-storage checking lists [130](#)
  - special considerations [199](#)
  - using during failsoft processing [367](#)
- global access checking table
  - entries for JESNEWS [488](#)
  - entries for STORCLAS and MGMTCLASS profiles [498](#)
  - listing [199](#)
  - performance benefits [187](#)
- global access checking table report
  - from DSMON [363](#)
- GLOBAL class
  - description [684](#)
- GLOBAL operand
  - SETROPTS command [118](#), [199](#)
  - use of [199](#)
- GLOBAL=YES option [238](#)
- GMBR class
  - description [684](#)
- GMQADMIN class
  - description [690](#)
- GMQNLIST class
  - description [690](#)
- GMQPROC class
  - description [690](#)
- GMQQUEUE class
  - description [690](#)
- GMXADMIN
  - description [691](#)
- GMXNLIST
  - description [691](#)
- GMXPROC
  - description [691](#)
- GMXQUEUE
  - description [691](#)
- GMXTOPIC

- GMXTOPIC (*continued*)
  - description [691](#)
- graphics device
  - controlling allocation with DEVICES profiles [232](#)
- group
  - as owner of data set profile [150](#)
  - as owner of resource profile [20](#)
  - assigning as owner of RACF profile [91](#)
  - attributes [5](#)
  - authority [5](#)
  - authority structure [63](#)
  - authority to access data set when not in access list [157](#)
  - authority to access general resource when not in access list [192](#)
  - authorizing to access resources [5](#)
  - benefits of using RACF groups [89](#)
  - defining
    - summary of steps [93](#)
  - defining for users with no common access requirements [86](#)
  - defining to RACF [13](#), [85](#)
  - definition [4](#)
  - deleting
    - summary of steps [94](#)
  - determining the owner of [362](#)
  - large [88](#)
  - listing all connections for a user [382](#)
  - maximum number of users in [85](#)
  - naming conventions for [89](#)
  - ownership of a RACF [90](#)
  - RACF commands for administration [14](#)
  - rationale for using [38](#)
  - relationships of users within [39](#)
  - scope of a [14](#)
  - specifying group terminal option [92](#)
  - summary of group-related user attributes [699](#)
  - UNIVERSAL attribute [88](#)
  - user attributes at group level [62](#)
  - using &RACGPID for global access checking [197](#)
  - using a dummy to protect data sets [150](#)
- group administrator
  - creating group for [85](#)
  - delegating responsibility to [92](#)
  - limits of authority of at the group level [63](#)
  - role of [9](#)
- group already defined in RACF
  - SYS1 (highest group in RACF structure) [13](#)
- group authority
  - assigning to user [17](#)
  - checking during list-of-groups checking [111](#)
  - during authorization checking [721](#), [722](#)
  - suggestions for assigning [92](#)
  - summary of authorities and commands [702](#)
  - types [91](#)
- group class
  - defining
    - overview [19](#)
- group data set
  - allowing access to using GRPACC attribute [61](#)
  - controlling creation of [149](#)
  - global access checking table entries using &RACGPID [198](#)
  - protecting [149](#)
  - user with GRPACC attribute [16](#)

- GROUP IDENT field
  - on TSO/E logon panel [511](#)
- group identifiers (GIDs) [513](#)
- group members
  - listing all [382](#)
- group name
  - as high-level qualifier for data set [149](#)
  - associating started procedure names with [141](#)
  - displaying from RACF database [24](#)
  - selecting [38](#)
  - specifying as prefix for data set with single-qualifier name [122](#)
- group names
  - mapping GID to [513](#)
  - mapping profiles for [526](#)
  - mapping to GIDs [524](#)
  - translating [478](#)
- group ownership of UNIX files [529](#)
- GROUP parameter
  - on JOB statement in JCL [364](#)
- group profile
  - authority granted through group-level attributes [63](#)
  - controlling access to DFP segment [502](#)
  - description of [18](#), [86](#)
  - DFP segment [499](#)
  - retrieving SMS information [501](#)
- group structure
  - establishing a RACF [39](#)
  - example [39](#)
  - mapping RACF to an existing [85](#)
- group terminal option
  - specifying on ADDGROUP or ALTGROUP command [227](#)
- group tree report
  - DSMON (data security monitor) [63](#)
  - from DSMON [362](#)
- group-AUDITOR attribute
  - as related to RACF commands [700](#)
  - description of [16](#), [58](#)
  - scope of authority [63](#)
- group-OPERATIONS attribute
  - alternatives
    - DASDVOL authority [60](#)
    - DFSMSdss authorization [60](#), [168](#)
  - description of [16](#), [60](#)
  - scope of authority [63](#)
- group-SPECIAL attribute
  - as related to RACF commands [699](#)
  - authority of user connected to group [90](#)
  - authority over data set profile owned by group [150](#)
  - description of [15](#), [58](#)
  - illustration of scope of authority [63](#)
  - scope of authority [63](#)
- GROUP.OMVS.\* profile (FIELD class) [88](#)
- GROUP.OMVS.GID profile (FIELD class) [88](#)
- GROUPJ qualifier
  - on NODES profiles [470](#)
- GROUPS qualifier
  - on NODES profiles [470](#)
- GRPACC (group access) attribute
  - comparison with &RACGPID [198](#)
  - description of [16](#), [61](#)
- GRPLIST operand
  - OMVS segment of group profile [112](#)
  - SETROPTS command [111](#)

- GRPLIST operand (*continued*)
  - with z/OS UNIX files [112](#)
- GSDSF class
  - description [684](#)
- GTERMINL class
  - activating [213](#)
  - description [684](#)
  - example [212](#)
  - protecting several terminals [226](#)
- GXCSEKEY class
  - description [689](#)
- GXFACILI class
  - description [684](#)
- GZMFAPLA class
  - description [691](#)

## H

- hardware configuration definition (HCD) program
  - device numbers [233](#)
  - unit names for devices [233](#)
- Hardware Configuration Definition (HCD) program
  - JES printer definitions [494](#)
  - unit names for devices [233](#)
- HBRADMIN class
  - description [684](#)
- HBRCMD class
  - description [684](#)
- HBRCONN class
  - description [684](#)
- HCICSECT class
  - description [687](#)
- help desk
  - delegating functions [655](#)
- high-level qualifier
  - during authorization checking
    - for data sets [721](#)
  - of data set profile name [147](#)
  - requirements for prefix [122](#)
- HIGHTRUST option for certificates [606](#)
- HIMS class
  - description [689](#)
- Hiperbatch
  - creating DLFCCLASS profiles for [235](#)
- HISTORY suboperand
  - PASSWORD operand
    - SETROPTS command [109](#)
- holding group
  - defining [85](#)
  - limit on number of users [85](#)
- hostIdMappings extension [605](#)

## I

- IBM MFA [70](#)
- IBM MQ
  - general resource classes [690](#)
- IBMUUSER user ID
  - description [357](#)
  - logging on as [357](#)
  - revoking [358](#)
- ICETOOL, DFSORT [375](#)
- ICF (Information Center Facility)

- ICF (Information Center Facility) (*continued*)
  - using to define users to RACF [76](#)
- ICHLBP profile (FACILITY class)
  - authorization checking if TAPEVOL class is active [182](#)
  - defining [182](#)
- ICHCNX00 exit routine
  - largely replaced by ICHNCV00 module [148](#)
- ICHDEX01 exit routine
  - description of [22](#)
  - using to encrypt passwords [139](#)
- ICHDEX11 exit routine
  - description of [22](#)
- ICHEINTY macro
  - and field-level access checking [200](#)
- ICHNCV00 module
  - description [147](#)
- ICHPWX01 exit routine
  - description of [22](#)
- ICHPWX11 exit routine
  - description of [22](#)
- ICHRTX00 exit
  - authorization checking [719](#)
  - migrating \$SUBMIT.userid profiles to SURROGAT profiles [456](#)
- ICHRTX01 exit
  - authorization checking [719](#)
- ICHSECOP module
  - preventing duplicate-named data set profiles [162](#)
- ICHUNCAT.data-set-name
  - protecting with FACILITY profile [120](#)
- ICKDSF (Device Support Facilities) system utility
  - DASDVOL authority [167](#)
- ICSF (Integrated Cryptographic Service Facility)
  - and digital certificates [582](#)
  - brief description [251](#)
  - defining delegated resources for FTPD use [257](#)
  - general resource classes [689](#)
- ICSF segment
  - FIELD profile names [206](#)
  - field-level access checking [201](#)
- ICTX segment
  - field-level access checking [201](#)
  - LDAPBIND profile
    - FIELD profile names [206](#)
- ID(\*) access list entry
  - contrasted with UACC [6](#), [158](#), [366](#)
  - for system data sets [711](#)
  - specifying [6](#), [158](#), [366](#)
- identification label
  - printed on output [98](#)
- identity filter, distributed
  - overview [671](#)
- IDIDMAP profile
  - description [672](#)
  - IRRRID00 utility processing [393](#)
- IDTPARMS segment
  - field-level access checking [201](#)
- IEAABD.DMPAKEY profile (FACILITY class)
  - defining to protect program dumps [231](#)
- IEAABD.DMPAUTH profile (FACILITY class)
  - defining to protect program dumps [230](#)
- IEAVECTOR profile (FACILITY class)
  - defining [229](#)
- IEC.TAPERING profile in the FACILITY class

- IEC.TAPERING profile in the FACILITY class (*continued*)
  - defining [180](#)
- IEHMOVE system utility
  - data sets with reserved names [152](#)
  - duplicate-named data sets [161](#)
- IIMS class
  - description [689](#)
- IKJEFF53 installation exit
  - and JESJOBS class [462](#), [464](#)
- ILMADMIN class
  - description [690](#)
- implementing RACF
  - checklist for team [42](#)
  - preparing plan [34](#)
- IMS
  - using with RACF [251](#)
- IMS (Information Management System)
  - general resource classes [689](#)
- IMS application
  - defining a PTKTDATA profile [284](#)
- in-storage profile
  - activating [125](#)
  - RACGLIST class [238](#)
  - refreshing generic profile lists [130](#)
  - saving [238](#)
  - SETROPTS GENLIST processing for [126](#)
  - SETROPTS options [125](#)
  - SETROPTS RACLIST processing for [127](#)
  - using during failsoft processing [367](#)
- INACTIVE operand
  - SETROPTS command [111](#)
- inbound work
  - authorizing with NODES profiles [468](#)
- INFOMAN class
  - description [689](#)
- Infoprint Server
  - general resource class [689](#)
- Information/Management
  - general resource classes [689](#)
- initACEE callable service
  - controlling the use [604](#)
  - description [546](#)
  - passing additional criteria [578](#)
- initialization statements
  - protecting JES resources [453](#)
- INITSTATS operand
  - SETROPTS command [116](#)
- input sources
  - defining nodes as local sources [482](#)
- installation exit
  - IKJEFF53 [462](#)
  - using to tailor RACF [21](#)
- installation exits
  - password authentication [22](#)
- installation security plan [453](#)
- installation-defined class
  - defining
    - overview [19](#)
- interprocess communication objects, restricting access [137](#)
- interval
  - for changing password and password phrases [108](#)
- INTERVAL suboperand MINCHANGE suboperand
  - PASSWORD operand
    - SETROPTS command [108](#)

- introduction
  - security administration [1](#)
- invalid user IDs
  - listing in data set access lists [382](#)
- IODEVICE statement (MVSCP) [233](#)
- IP address
  - conditional access to SERVAUTH class [194](#)
  - RACF authorization checking for [731](#)
- IP address control with SERVAUTH resources [314](#)
- IPCOBJ class
  - description [692](#)
  - z/OS UNIX event auditing [537](#)
- IRR.DIGTCERT.ADD [594](#), [604](#), [610](#)
- IRR.DIGTCERT.ADDRING [595](#)
- IRR.DIGTCERT.ALTER [564](#)
- IRR.DIGTCERT.CONNECT [595](#)
- IRR.DIGTCERT.DELETE [565](#), [604](#)
- IRR.DIGTCERT.EXPORT [610](#)
- IRR.DIGTCERT.function [549](#)
- IRR.DIGTCERT.GENCERT [610](#)
- IRR.DIGTCERT.GENRENEW [610](#)
- IRR.DIGTCERT.LIST [550](#), [554](#)
- IRR.DIGTCERT.QRECOVER [610](#)
- IRR.DIGTCERT.REQCERT [610](#)
- IRR.DIGTCERT.REQRENEW [610](#)
- IRR.DIGTCERT.RESPOND [610](#)
- IRR.DIGTCERT.REVOKE [610](#)
- IRR.DIGTCERT.SCEPREQ [610](#)
- IRR.DIGTCERT.VERIFY [610](#)
- IRR.HOST.host-name [605](#)
- IRR.LISTUSER [655](#)
- IRR.LU.EXCLUDE [659](#)
- IRR.LU.OWNER [657](#)
- IRR.LU.TREE [658](#)
- IRR.PASSWORD.RESET [662](#)
- IRR.PROXY.DEFAULTS [607](#)
- IRR.PWRESET.EXCLUDE [666](#)
- IRR.PWRESET.OWNER [663](#)
- IRR.PWRESET.TREE [664](#)
- IRR.RAUDITX [606](#)
- IRR.RCACHESERV.cachename [606](#)
- IRR.RDCEKEY [607](#)
- IRR.RDCERUID [608](#)
- IRR.RGETINFO.EIM [608](#)
- IRR.RPKISERV [609](#)
- IRR.RPKISERV.PKIADMIN [611](#)
- IRR.RPROXYSERV [614](#)
- IRR.RTICKETSERV [615](#)
- IRRACEE class [356](#)
- IRRADU00 utility
  - brief overview [24](#)
  - logging [3](#)
- irrcerta user ID [584](#)
- IRRDBU00 utility
  - allowable parameters
    - LOCKINPUT [374](#)
    - NOLOCKINPUT [374](#)
    - UNLOCKINPUT [375](#)
  - and RACGLIST profiles [372](#)
  - brief overview [23](#)
  - checking category profiles in the SECDATA class [97](#)
  - creating a Db2 database [382](#)
  - creating a Db2 table space [383](#)
- IRRDBU00 utility (*continued*)
  - creating optimization statistics for the Db2 database [385](#)
  - creating the Db2 indexes [383](#)
  - creating the Db2 tables [383](#)
  - CSDATA segment [372](#)
  - Db2 table names [385](#)
  - Db2 utility statements
    - for deleting group records [385](#)
    - for loading tables [384](#)
  - deleting data from the Db2 database [385](#)
  - detailed description [371](#)
  - diagnostic capability [371](#)
  - example [374](#)
  - executing
    - input data set specification [374](#)
  - listing universal group members [372](#)
  - loading the Db2 tables [384](#)
  - OMVS segment of user profile [372](#)
  - operational considerations [372](#)
  - OVMS segment of user profile [372](#)
  - performance considerations [371](#)
  - QMF form [390](#)
  - record types [385](#)
  - reorganizing the unloaded RACF data in the Db2 database [384](#)
  - report output [391](#)
  - reports using RACFICE [378](#)
  - sample report [391](#)
  - sample SQL query [390](#)
  - samples using the IRRDBU00 utility output [390](#)
  - SQL query [390](#)
  - SQL utility statements
    - for creating indexes [383](#)
    - for defining a table space [383](#)
  - SQL utility statements creating a table [383](#)
  - steps for using IRRDBU00 output with Db2 [382](#)
  - universal groups [372](#)
  - usage of the class descriptor table (CDT) [372](#)
  - using output with Db2 [382](#)
  - using output with DFSORT ICETOOL [375](#)
- IRRDP100 command [642](#)
- IRRIRA00 conversion utility [524](#)
- IRRMIN00 utility
  - brief overview [23](#)
- irrmulti user ID [584](#)
- IRRID00 utility
  - brief overview [24](#)
  - deleting universal groups [403](#)
  - detailed description [391](#)
  - examples [395](#)
  - general resource processing [403](#)
  - IDIDMAP profiles [393](#)
  - member processing [403](#)
  - NOTELINK and NDSLINK profiles [393](#)
  - output [398](#)
  - removing universal group members [403](#)
  - return codes [396](#)
  - time requirements [404](#)
  - Tivoli considerations [403](#)
  - universal groups [403](#)
- IRRSAX00 callable service
  - controlling the use [606](#)
- IRRSCH00 callable service

- IRRSCH00 callable service (*continued*)
  - controlling the use [606](#)
- IRRSKD00 callable service
  - controlling the use [607](#)
- IRRSDL00 or IRRSDL64 callable service
  - controlling the use [606](#)
  - description [546](#)
- IRRSEQ00 callable service
  - controlling the use [606](#)
- IRRSGL00 callable service
  - controlling the use [608](#)
- IRRSIA00 callable service
  - controlling the use [604](#)
  - description [546](#)
- irrsitec user ID [584](#)
- IRRSPK00 callable service
  - controlling the use [614](#)
- IRRSPX00 callable service
  - controlling the use [608](#)
- IRRSPLY00 callable service
  - controlling the use [614](#)
- IRRSUD00 callable service
  - controlling the use [608](#)
- IRRUT100 utility
  - brief overview [24](#)
- IRRUT200 utility
  - brief overview [24](#)
- IRRUT400 utility
  - brief overview [23](#)
- ISPF panels
  - advantages [12](#)
  - authorization, additional [13](#)
  - authorizing users to update custom field data [646](#)
  - sample for password rules [12](#)
  - using instead of commands [12](#)
- issuer's name filters [574](#)
- issuer's X.509 distinguished name [572](#)

## J

- JAVA class
  - description [688](#)
- JCICSJCT class
  - description [687](#)
- JCL (job control language)
  - parameters related to RACF [364](#)
  - PROTECT parameter in JCL [180](#)
  - PROTECT parameter on DD statement [162](#)
  - replacing password with PassTicket [284](#)
  - restricting the use of //DD DATA statement [366](#)
  - SECMODEL parameter on DD statement [162](#)
  - tape data sets [180](#)
- JES (job entry subsystem)
  - activating or deactivating options for [120](#)
  - password print suppression [364](#)
  - RACF support for [453](#)
  - restricting use of JES3 operator commands [365](#)
  - security [453](#)
  - STARTED class [453](#)
  - started procedures table (ICHRIN03) [453](#)
  - working with RACF [453](#)
- JES commands
  - operator
    - protecting with OPERCMDS profiles [239](#)

- JES input device
  - allowing access depending on JES input device [157](#), [193](#)
  - protecting with JESINPUT profiles [467](#)
  - RACF authorization checking for [731](#)
- JES segment
  - field-level access checking [201](#)
- JESINPUT class
  - activating [468](#)
  - authorization checking [731](#)
  - description [684](#)
  - protecting JES input devices [467](#)
  - protecting NJE nodes [493](#)
  - protecting RJE workstations [493](#)
  - specifying with WHEN operand on PERMIT command [157](#), [193](#)
  - spool reload [489](#)
  - UACC authorities [467](#)
- JESJOBS class
  - activating [463](#), [464](#)
  - and &RACLNDE profile [218](#)
  - controlling job class usage [465](#)
  - controlling who can modify job attributes [465](#)
  - controlling who can use Job Modify SSI 85 [465](#)
  - description [684](#)
  - finding profiles whose names include a particular user ID [78](#)
  - protecting job cancellation [464](#)
  - protecting job names [461](#)
  - protecting job submission [462](#)
  - protecting register job [463](#)
  - protecting spool reload [489](#)
  - RACF variable example [218](#)
- JESNEWS data set
  - and SECLABEL class [488](#)
  - protecting with JESSPOOL profile [487](#)
  - protecting with OPERCMDS profiles [487](#)
- JESSPOOL class
  - activating [483](#)
  - and &RACLNDE profile [218](#)
  - and SETROPTS GENERICOWNER [113](#)
  - and TSO OUTPUT command [511](#)
  - and TSO RECEIVE command [511](#)
  - authorizing users to create profiles [486](#)
  - description [684](#)
  - external writer access to profiles [484](#)
  - finding profiles whose names include a particular user ID [78](#)
  - protecting JESNEWS data set [487](#)
  - protecting SYSIN [483](#)
  - protecting SYSOUT [483](#)
  - protecting trace data sets [488](#)
- JIMS class
  - description [689](#)
- job cancellation
  - protecting with JESJOBS profiles [464](#)
- job class
  - controlling use of [465](#)
- job data sets
  - protecting [483](#)
- job execution
  - refreshing global access checking lists [130](#)
  - refreshing in-storage generic profile lists [130](#)
- job initialization
  - brief description [738](#)



- job names
  - protecting with JESJOBS classes [461](#)
- job spool reload
  - JESINPUT profiles [489](#)
- JOB statement (JCL)
  - ensuring the security of [364](#)
  - for started procedures [141](#)
  - parameters related to RACF [364](#)
  - specifying password for deferred restart on [365](#)
  - started procedure system-generated [141](#)
  - verifying
    - user ID data [455](#)
- job submission
  - protecting with JESJOBS profiles [462](#)
- jobnames
  - controlling job names [464](#)
  - controlling with RACF [464](#)
- JOBNAMEs field
  - DLFCLASS profile [236](#)
- jobs
  - authorizing NJE [473](#)
  - restarting [365](#)
  - starting [143](#)
- JOIN group authority
  - as related to RACF commands [702](#)
  - description [92](#)

## K

- KCICSJCT class
  - description [687](#)
- KERB segment
  - field-level access checking [201](#)
  - REALM profile
    - FIELD profile names [207](#)
  - RRSF consideration [434](#)
  - user profile
    - contents of [50](#)
    - FIELD profile names [207](#)
- KERBLINK class
  - description [690](#)
  - RRSF consideration [434](#)
- key ring
  - overview [567](#)
  - sharing a certificate (private key) with multiple servers [568](#)
- key, PassTicket
  - and RACF database [287](#)
  - defining [282](#)
  - protecting [286](#)
- keyboard
  - navigation [741](#)
  - PF keys [741](#)
  - shortcut keys [741](#)
- KEYENCRYPTED value
  - RALTER command [287](#)
  - RDEFINE command [287](#)
- KEYMASKED value
  - RALTER command [287](#)
  - RDEFINE command [287](#)
- KEYSMSTR class [255](#)

## L

- LABEL parameter (JCL DD statement)
  - parameters related to RACF [365](#)
- LAN (local area network)
  - PassTicket [281](#)
- LAN File Services/ESA (LFS/ESA) [689](#)
- LANGUAGE operand
  - SETROPTS command [125](#)
- LANGUAGE segment
  - field-level access checking [201](#)
  - user profile
    - contents of [50](#)
    - FIELD profile names [207](#)
- large groups [88](#)
- large profile size considerations [193](#)
- LDAP class
  - description [685](#)
- LDAP server
  - BIND password [255](#)
  - defining an LDAP bind profile [617](#)
  - event notification [618](#)
  - profile change notification [618](#)
- LDAPBIND profile [617](#)
- LFS/ESA (LAN File Services/ESA)
  - general resource class [689](#)
  - protecting file services with LFSCLASS profiles [224](#)
- LFSCLASS class
  - activating [224](#)
  - defining profiles [224](#)
  - description [689](#)
  - protecting LFS/ESA file services [224](#)
- library lookaside (LLA) security [234](#)
- License Manager
  - general resource class [689](#)
- limited use of %\* in general resource profile names [189](#)
- limiting
  - access to system for terminal [227](#)
  - OPERATIONS user's authority [60](#)
  - size of access lists [193](#)
  - when a user can log on to system [73](#)
- limits for z/OS UNIX users [515](#)
- LIMS class
  - description [689](#)
- line drop facility
  - on TSO [227](#)
- list-of-groups checking
  - activating [111](#)
  - deactivating [111](#)
  - during authorization checking [731](#), [732](#)
  - effect on user with group-level attribute [62](#)
  - OMVS segment of group profile [112](#)
- LISTBC command (TSO)
  - auditing when used [248](#)
- LISTDSD command
  - finding out which profile protects a data set [155](#)
- listing user information
  - delegating authority for [655](#)
- LISTUSER command
  - PROTECTED attribute [74](#)
  - RESTRICTED attribute [75](#)
- LLA (library lookaside) security [234](#)
- LLA-managed data sets
  - protecting with FACILITY profiles [234](#)

- LNOTES segment
  - field-level access checking [201](#)
  - user profile
    - contents of [51](#)
    - FIELD profile names [207](#)
- LNOTES segment, USER profile [252](#)
- local mode for an RRSF node
  - description [407](#)
- local node
  - command direction [412](#)
- local nodes
  - treating some network nodes as local nodes [482](#)
- log string
  - using to debug your security [716](#), [717](#)
- logging
  - access attempts to resources [2](#)
  - nonstandard data set names in SMF [148](#)
  - real data set names [122](#)
- logging on
  - preventing user IDs [74](#)
  - to TSO from another terminal [227](#)
- LOGON command with RECONNECT operand on TSO [227](#)
- logon initialization
  - brief description [738](#)
- logon procedure for TSO
  - protecting [507](#)
- Lotus Notes for z/OS
  - general resource class [690](#)
- LU 6.2 bind
  - RACF support for [221](#)
- LU security capabilities
  - with APPC [250](#)

## M

- MAIN attribute for programs
  - defining [317](#)
  - overview [316](#)
- managed
  - user ID associations
    - defining [410](#)
- mandatory access control (MAC)
  - definition [8](#)
- mapping
  - GIDs and VLF [524](#)
  - UIDs and VLF [524](#)
- mapping profiles
  - for certificate names [571](#)
  - for distributed identities [671](#)
  - for UIDs and GIDs [526](#)
- masked keys [288](#)
- masking
  - PassTicket key [286](#)
- master scheduler initialization (MSI) [719](#)
- maximum field length unloaded by IRRDBU00 [372](#)
- maximum number of users per group [85](#)
- maximum security for PassTicket keys [287](#)
- maximum VTAM session interval length [132](#)
- MCICSPPT class
  - description [687](#)
- MCS [240](#)
- MCS console
  - protecting [228](#)
  - protecting with CONSOLE profiles [228](#)

- MCSOPER macro
  - and OPERPARM segment [52](#)
- MDSNBP class
  - description [688](#)
- MDSNCL class
  - description [688](#)
- MDSNDB class
  - description [688](#)
- MDSNGV class
  - description [688](#)
- MDSNJR class
  - description [688](#)
- MDSNPK class
  - description [688](#)
- MDSNPN class
  - description [688](#)
- MDSNSC class
  - description [688](#)
- MDSNSG class
  - description [688](#)
- MDSNSM class
  - description [688](#)
- MDSNSP class
  - description [688](#)
- MDSNSQ class
  - description [688](#)
- MDSNTB class
  - conditional access [194](#)
  - description [688](#)
- MDSNTS class
  - description [688](#)
- MDSNUF class
  - description [688](#)
- MDSNUT class
  - description [688](#)
- member class
  - defining
    - overview [19](#)
    - SETROPTS RACLIST processing [215](#)
- merged profiles, size considerations [193](#)
- message
  - password expiration [109](#)
  - putting real data set names into [122](#)
  - unauthorized access attempt [3](#)
  - warning
    - rationale for using [37](#)
- message processing
  - password synchronization [408](#)
- message traffic
  - auditing [248](#)
  - controlling [245](#)
- MFA
  - application bypass [72](#)
  - general resource classes [690](#)
  - policy [72](#)
- MFA application bypass [72](#)
- MFA policy [72](#)
- MFA segment
  - MFA class profile
    - FIELD profile names [207](#)
- MFADEF class
  - description [690](#)
- MFPOLICY segment
  - MFPOLICY class profile



- MFPOLICY segment (*continued*)
  - MFPOLICY class profile (*continued*)
    - FIELD profile names [207](#)
- MGMTCLAS class
  - description [690](#)
  - protecting SMS management classes [497](#)
  - SETOPTS RACLIST processing [498](#)
- MGMTCLAS parameter (JCL DD statement)
  - parameters related to RACF [365](#)
- migration
  - converting from LEVEL to SECLEVEL [97](#)
  - defining JES
    - as a started procedure [453](#)
    - with the trusted attribute [453](#)
  - of existing user IDs to RACF [56](#)
  - protecting existing data [35](#)
  - surrogate users [456](#)
- MIMS class
  - description [689](#)
- MLACTIVE operand
  - SETOPTS command
    - relationship to SECLABEL class [737](#)
    - security labels and tapes [175](#)
- MLFSOBJ operand
  - SETOPTS command [137](#)
- MLIPCOBJ operand
  - SETOPTS command [137](#)
- MLNAMES operand
  - SETOPTS command [138](#)
- MLQUIET operand
  - SETOPTS command
    - relationship to SECLABEL class [737](#)
- MLS operand
  - SETOPTS command
    - relationship to SECLABEL class [737](#)
- MLS(FAILURES) option
  - SETOPTS command
    - security labels and tapes [175](#)
- MLSTABLE operand
  - SETOPTS command [133](#)
- mode
  - local [407](#)
  - remote [407](#)
- model data set profile
  - for GDG data sets [161](#)
  - system-wide processing options [121](#)
  - ways to use [36](#), [158](#)
- model general resource profile
  - ways to use [36](#)
- MODEL operand
  - ADDGROUP command [159](#)
  - ADDSD command
    - for group data sets [159](#)
  - ADDUSER command [158](#)
  - ALTGROUP command [159](#)
  - ALTUSER command [158](#)
  - SETOPTS command
    - for GDG data sets [161](#)
    - for group data sets [160](#)
    - for user data sets [158](#)
- modeling certificate name filters [577](#)
- MODIFY LLA command
  - controlling the use of [234](#)
- MQADMIN class
  - description [690](#)
- MQCMDS class
  - description [690](#)
- MQCONN class
  - description [690](#)
- MQNLIST class
  - description [690](#)
- MQPROC class
  - description [690](#)
- MQQUEUE class
  - description [690](#)
- Multi-factor Authentication [70](#)
- MULTIID option of RACDCERT MAP command [578](#)
- multilevel security
  - and SMESAGE class [246](#)
  - enforcing fully [135](#)
  - enforcing partially (compatibility mode) [135](#)
- multiple console support
  - sysplex
    - command authorization in [240](#)
- multiple profiles
  - for resource groups
    - resolving conflicts [214](#)
- multisystem node [406](#)
- multivolume data set
  - non-VSAM DASD data set
    - considerations for protecting [163](#)
  - protecting [162](#)
  - tape data set
    - considerations for protecting [163](#)
  - VSAM DASD data set
    - considerations for protecting [163](#)
- multivolume tape data set
  - protecting [171](#)
  - RACF processing for [181](#)
  - scratching a [167](#)
- MVS message service
  - and SETROPTS LANGUAGE command [125](#)
- MVS system commands
  - MODIFY LLA command
    - controlling the use of [234](#)
  - operator
    - example of protecting [242](#)
    - protecting with OPERCMDS profiles [239](#)
  - START LLA command
    - controlling the use of [234](#)
- MXADMIN
  - description [691](#)
- MXNLIST
  - description [691](#)
- MXPROC
  - description [691](#)
- MXQUEUE
  - description [691](#)
- MXTOPIC
  - description [691](#)

## N

- name
  - defining for group [89](#)
  - defining for user [56](#)
- name filters

- name filters (*continued*)
  - for certificates [571](#)
  - for distributed identity names [671](#)
- name qualifier
  - &RACUID and &RACGPID for global access checking [197](#)
- name-hiding, with security labels [138](#)
- naming convention table
  - erasing sensitive temporary data sets [165](#)
  - protecting temporary data sets with PROTECTALL [120](#)
- naming conventions
  - conforming or not conforming to [148](#)
  - for data set profiles [147](#)
  - for group [89](#)
  - for user [56](#)
  - making use of existing [38](#)
  - table-driven for data sets [147](#)
  - ways to enforce for data sets [147](#)
- national language
  - user's preferred primary and secondary languages [50](#)
- navigation
  - keyboard [741](#)
- NCICSPPT class
  - description [687](#)
- NDS segment
  - field-level access checking [201](#)
  - user profile
    - contents of [51](#)
    - FIELD profile names [207](#)
- NDS segment, USER profile [252](#)
- NDSLINK class
  - and IRRRID00 utility processing [393](#)
  - description [690](#)
- nested ACEEs [257](#)
- NETCMDS class
  - description [690](#)
- NETSPAN class
  - description [690](#)
- NetView
  - general resource classes [690](#)
- NETVIEW segment
  - field-level access checking [201](#)
  - user profile
    - contents of [51](#)
    - FIELD profile names [207](#)
- network
  - NJE
    - propagation in [481](#)
- network security
  - NJE network [468](#)
  - PassTicket [281](#)
- network security zone, controlling [314](#)
- networking profiles
  - description [469](#)
- NEVERCONTAIN attribute
  - defining [81](#)
- NEVERCONTAIN operand [81](#)
- NEW PASSWORD field
  - on TSO/E logon panel [511](#)
- new-password exit
  - description of [22](#)
- new-password-phrase exit
  - description of [22](#)
- NJE

- NJE (*continued*)
  - authorizing jobs [473](#)
  - authorizing SYSOUT [476](#)
  - controlling user ID propagation [475](#)
  - header information [459](#)
  - network security [468](#)
  - propagation across nodes [481](#)
  - protecting nodes with FACILITY profiles [493](#)
  - protecting nodes with JESINPUT profiles [493](#)
  - validating SYSOUT [477](#)
  - verifying jobs [458](#)
- NJEUSERID operand
  - SETROPTS command [481](#)
- NOADDCREATOR options [163](#)
- NOADSP operand
  - revoking ADSP attribute [62](#)
  - SETROPTS command [122](#)
- NOCONTAIN operand [81](#)
- NODES class
  - activating [482](#), [496](#)
  - and RACFVARS class [475](#)
  - and RACFVARS profiles [471](#)
  - authorizing inbound work (JES) [468](#)
  - checking [459](#)
  - controlling commands from NJE nodes [496](#)
  - description [685](#)
  - finding profiles whose names include a particular user ID [78](#)
  - profile overview [469](#)
  - profiles with RUSER as second qualifier [496](#)
  - protecting spool reload [489](#)
  - SETROPTS RACLIST processing [482](#)
  - treating some network nodes as local nodes [482](#)
- nodes, RRSF
  - directing commands to a local node [412](#)
  - directing commands to a remote node [413](#)
  - local [406](#)
  - local mode [407](#)
  - multisystem [406](#)
  - remote [406](#)
  - remote mode [407](#)
  - single-system [406](#)
- NODMBR class
  - description [685](#)
- NOEXPIRED operand [661](#)
- NOGLOBAL operand
  - SETROPTS command [199](#)
- NOHISTORY suboperand
  - PASSWORD operand
    - SETROPTS command [109](#)
- non-VSAM data set
  - multivolume considerations [163](#)
- nonautomatic TAPEVOL profile [177](#)
- NONE access authority
  - as related to RACF commands [703](#)
  - for general resources [193](#)
- nonstandard naming conventions
  - changing with the naming convention table [147](#)
- NOOIDCARD operand [74](#)
- NOPADCHK option, with program access to data sets (PADS) [314](#)
- NOPASSWORD operand [74](#)
- NOPHRASE operand [74](#)
- NORESTRICTED operand [75](#)

- NOSTATISTICS operand
  - SETROPTS command [118](#)
- NOTELINK class
  - and IRRRID00 utility processing [393](#)
  - description [690](#)
- NOTERMUACC operand
  - for group terminal option [92](#), [227](#)
- notification, LDAP event changes [618](#)
- NOTRUST option of the RACDCERT command [564](#)
- Novell Directory Services for OS/390
  - general resource class [690](#)
- NVASAPDT class
  - description [690](#)

## O

- offloading spool (JES2 only) [488](#)
- OIDCARD (operator identification card)
  - storing information in the RACF database [139](#)
  - using as a password [1](#)
- OIMS class
  - description [689](#)
- OMVS segment
  - automatic assignment of unique UNIX identities [518](#)
  - description [18](#)
  - field-level access checking [201](#)
  - group profile
    - contents of [87](#)
    - FIELD profile names [208](#)
    - list-of-groups checking [112](#)
  - in group profiles
    - assigning unique GIDs [518](#)
  - in user profiles
    - assigning unique UIDs [518](#)
  - protecting with FIELD profiles [52](#), [88](#)
  - user profile
    - contents of [52](#)
    - FIELD profile names [208](#)
    - IRRDBU00 utility [372](#)
- ONLYAT operand
  - controlling use of [437](#)
  - using [415](#)
- operating RACF
  - considerations for administrators [355](#)
- OPERATIONS attribute
  - alternatives
    - DASDVOL authority [60](#)
    - DFSMSdss authorization [60](#), [168](#)
  - as related to RACF commands [701](#)
  - comparison to DASDVOL authority [167](#)
  - delegating [60](#)
  - description of [16](#), [59](#)
  - during authorization checking [722](#)
  - listing users with [67](#)
  - scratching temporary data sets when TEMPDSN class is active [224](#)
  - suggestions for assigning [66](#)
- operator command
  - JES3
    - restricting [365](#)
  - MVS
    - example of protecting [242](#)
    - protecting with OPERCMDS profiles [239](#)
  - unknown

- operator command (*continued*)
  - unknown (*continued*)
    - auditing [241](#)
    - protecting with OPERCMDS profiles [241](#)
- operators
  - assigning to RACF groups [454](#)
  - creating user IDs [57](#)
  - defining to RACF [454](#)
- OPERAUDIT operand
  - SETROPTS command [60](#)
- OPERCMD class
  - activating [242](#), [495](#), [496](#)
  - auditing for password security [366](#)
  - conditional access [194](#), [241](#)
  - description [685](#)
  - example [242](#)
  - protecting JESNEWS data sets [487](#)
  - protecting operator commands [239](#)
  - protecting SDSF panels [239](#)
  - protecting unknown operator commands [241](#)
  - SETROPTS RACLIST processing [240](#)
- OPERPARM segment
  - description [17](#)
  - field-level access checking [201](#)
  - user profile
    - contents of [52](#)
    - FIELD profile names [208](#)
- OPTAUDIT class
  - description [685](#)
- options
  - controlling RACF [105](#)
  - of commands
    - AT [412](#)
    - ONLYAT [415](#)
  - selecting RACF [21](#)
  - selecting system-wide with SETROPTS command [105](#)
- organization chart as pattern
  - for group-user structure with RACF [13](#)
- origin LU authorization
  - APPL class [250](#)
- outbound work
  - authorizing with WRITER profiles [483](#)
  - using security labels [483](#)
- output capturing
  - for password synchronization [409](#)
  - for RACF TSO commands [413](#)
- OUTPUT command (TSO)
  - controlling use [511](#)
- output destination
  - controlling with WRITER profiles [493](#)
- output of RACF commands, capturing [26](#)
- OUTPUT statement (JCL)
  - parameters related to RACF [365](#)
- overriding
  - default UACC for a data set profile [157](#)
  - default UACC for general resource [192](#)
  - SUPERUSER.FILESYS authority [532](#)
- overview, digital certificates [539](#)
- overwriting
  - a tape data set or tape volume [178](#)
  - authority to overwrite a tape data set [179](#)
- OVM segment
  - field-level access checking [201](#)

- OVM segment (*continued*)
  - group profile
    - FIELD profile names [208](#)
  - user profile
    - FIELD profile names [208](#)
    - IRRDBU00 utility [372](#)
- owner
  - GENERICOWNER operand on SETROPTS command [113](#)
- ownerless profile [355](#)
- ownership
  - of data set profile [150](#)
  - of RACF group [90](#)
  - of RACF profile [6](#)
  - of RACF user profile [57](#)
  - resource profile [20](#)
  - scope of authority [63](#)
  - various concepts of [20](#)
- ownership structure
  - establishing for your installation [38](#)

## P

- PADCHK option, with program access to data sets (PADS) [314](#)
- partner LU
  - protecting conversations with APPL profiles [250](#)
- partner LU name device
  - conditional access to APPCPORT class [157](#), [194](#)
  - conditional access to SERVAUTH class [157](#)
- PassTicket
  - activating the PTKTDATA class [282](#)
  - applications that generate [289](#)
  - bypassing PassTicket replay protection [290](#)
  - defining profiles [282](#)
  - definition [281](#)
  - enabling [291](#)
  - generating [289](#)
  - in a shared environment [287](#)
  - overview [281](#)
  - problem prevention [293](#)
  - protecting keys [286](#)
  - protecting with PTKTDATA profiles [282](#)
  - time range [289](#)
  - validating [289](#)
  - verifying the environment [292](#)
- PassTicket key
  - and RACF database [287](#)
  - defining [282](#)
  - encrypting [287](#)
  - masking [286](#)
  - protecting [286](#)
- password
  - activating or deactivating monitoring options [109](#)
  - alternative to [281](#)
  - authentication exits [22](#)
  - automatic direction [433](#)
  - bypassing protection [362](#), [365](#)
  - case options [106](#)
  - change interval [108](#)
  - checking after restarting a job [365](#)
  - consecutive incorrect attempts to revoke user ID [109](#)
  - controlling access to RACF passwords [365](#)
  - controlling automatic direction of [439](#)
  - delegating authority to reset [660](#)

- password (*continued*)
  - encryption of RACF user [139](#)
  - enveloping [623](#)
  - exit routines [22](#)
  - for RVARY command processing [112](#)
  - maintaining for password-protected data sets [181](#)
  - mixed-case options [106](#)
  - NOPASSWORD option [74](#)
  - NOPHRASE option [74](#)
  - number of previous values to be saved [109](#)
  - preventing revocation of user IDs [74](#)
  - print suppression by JES [364](#)
  - processing exit [22](#)
  - rationale for using [1](#)
  - replacing with PassTicket in JCL [284](#)
  - resetting
    - using IRR.PASSWORD.RESET authority [660](#)
  - special characters options [106](#)
  - specifying on TSO LOGON command [511](#)
  - syntax rules [108](#)
  - validating [289](#)
  - versus RACF authorization requirements for VSAM data sets [165](#)
  - warning message for password expiration [109](#)
- password enveloping [623](#)
- PASSWORD field
  - on TSO/E logon panel [511](#)
- password on bind
  - RACF support for [221](#)
- PASSWORD operand
  - SETROPTS command [106](#), [108](#)
- PASSWORD parameter
  - on JOB statement in JCL
    - consideration for inbound NJE jobs [471](#)
    - consideration for surrogate job submission [455](#)
- password phrase
  - activating or deactivating monitoring options [109](#)
  - assigning [69](#)
  - change interval [108](#)
  - consecutive incorrect attempts to revoke user ID [109](#)
  - controlling automatic direction of [439](#)
  - delegating authority to change [660](#)
  - new-password-phrase exit (ICHPWX11) [70](#)
  - number of previous values to be saved [109](#)
  - rationale for using [1](#)
  - resetting
    - using IRR.PASSWORD.RESET authority [660](#)
  - synchronization, controlling [435](#)
  - syntax rules [70](#)
- password phrases [68](#)
- password protection
  - bypassing [19](#), [362](#)
  - utilizing or bypassing for tape data sets [170](#)
  - utilizing or bypassing for tape volumes [170](#)
- password rules
  - ISPF panel for [12](#)
- password synchronization
  - controlling [435](#)
  - defining
    - for other users [410](#)
    - for your user ID [410](#)
  - message processing [408](#)
  - output capturing [409](#)
- password-protected data set

password-protected data set (*continued*)  
     providing protection for [181](#)

passwords [68](#)

PCICSPSB class  
     description [687](#)

PERFGRP class  
     activating [508](#)  
     authorization checking for [510](#)  
     considerations [509](#)  
     description [691](#)  
     protecting TSO performance groups [507](#)  
     SETROPTS RACLIST processing [509](#)  
     UACC authorities [508](#)

performance  
     and DASDVOL authority [167](#)  
     and UNIXMAP class [524](#)  
     and VLF [524](#)  
     for general resource profiles [187](#)  
     generic profiles [192](#)  
     global access checking [187](#)  
     SETROPTS RACLIST processing [187](#)

performance group for TSO  
     protecting [507](#)

permission bits  
     for z/OS UNIX data  
     [530](#)

PERMIT command  
     to limit OPERATIONS user's authority [60](#)

PHRASESYNC resource  
     in RRSFDATA class [435](#)

PIMS class  
     description [689](#)

PKA key data set, storing private keys in [582](#)

PKDS, storing private keys in [582](#)

PKISERV class  
     protecting R\_PKIServ administrative functions with [612](#)

PKISERVclass  
     description [685](#)

planning  
     JES system programmer [453](#)  
     security [453](#)

PMBR class  
     description [685](#)

POSIT value in class descriptor table (CDT)  
     changing a value [269](#)  
     extends CLAUTH authority [60](#)  
     shared value [267](#)  
     unique value [265](#)

POSIX\_CHOWN\_RESTRICTED constant [527](#)

PPT (program properties table)  
     bypass password protection setting [362](#), [365](#)  
     DSMON report [362](#)  
     system key setting [362](#)

prefix  
     for single-qualifier data set name [153](#)

PREFIX operand  
     SETROPTS command  
         single-qualifier data set names [148](#)

preventing  
     accesses to resources with REQUEST=AUTH  
     preprocessing routine [367](#)  
     changes to security labels [133](#)  
     duplicate-named data set profiles [162](#)  
     security exposure due to //DD DATA statement [366](#)

preventing (*continued*)  
     user from accessing system [61](#)

preventing generic profile names [188](#)

primary language  
     installation defaults [125](#)

primary RACF database [368](#)

Print Services Facility (PSF) for z/OS  
     identification label [98](#)  
     protecting services with PSFMPL profiles [247](#)  
     PSFMPL, general resource class [685](#)

printer security  
     controlling with DEVICES profiles [494](#)

PRINTSRV class  
     description [689](#)

private key  
     expiring certificates [586](#)  
     extending [585](#)  
     extracting using R\_datalib callable service [607](#)  
     RACF size restrictions [547](#)  
     rekeying [586](#)  
     renewing [585](#)  
     replacing [586](#)  
     retiring [586](#)  
     rollover [586](#)  
     sharing with multiple servers [568](#)  
     storing in Integrated Cryptographic Service Facility (ICSF) [582](#)  
     suppression of information during RRSF propagation  
     [431](#)

privileged attribute  
     authorization checking [719](#)  
     determining which started procedures have [363](#)  
     started procedure [142](#)

problems  
     jobs cannot be initiated [739](#)  
     started procedures fail [739](#)  
     users cannot log on [739](#)

PROACT class  
     description [692](#)  
     z/OS UNIX event auditing  
     [537](#)

PROCESS class  
     description [692](#)  
     z/OS UNIX event auditing  
     [537](#)

products supported by RACF  
     DFSMSdfp [497](#)  
     JES [453](#)  
     SMS (Storage Management Subsystem) [497](#)  
     TSO [507](#)

products, IBM  
     using with RACF  
         CICS [250](#)  
         IMS [251](#)

profile  
     coordinating updates [355](#)  
     description of discrete and generic [18](#)  
     description of group [86](#)  
     description of user [45](#)  
     group  
         contents of [18](#)  
     group ownership of a [91](#)  
     IRR.DIGTCERT.ADD [610](#)  
     IRR.DIGTCERT.EXPORT [610](#)

profile (*continued*)

- IRR.DIGTCERT.GENCERT [610](#)
- IRR.DIGTCERT.GENRENEW [610](#)
- IRR.DIGTCERT.QRECOVER [610](#)
- IRR.DIGTCERT.REQCERT [610](#)
- IRR.DIGTCERT.REQRENEW [610](#)
- IRR.DIGTCERT.RESPOND [610](#)
- IRR.DIGTCERT.REVOKE [610](#)
- IRR.DIGTCERT.SCEPREQ [610](#)
- IRR.DIGTCERT.VERIFY [610](#)
- IRR.RPKISERV.PKIADMIN [611](#)
- listing information from RACF [26](#)
- owning a data set [150](#)
- owning a group [90](#)
- recording statistics in [26](#)
- rules for defining data set [147](#)
- searching for [28](#)
- secured signon [282](#)
- size considerations [193](#)
- types of RACF [6](#)
- user

- contents of [17](#)

profile modeling

- automatic [158](#)
- ways to use [36](#)

profile name

- for PTKTDATA class [283](#)
- high-level qualifier of data set [147](#)
- rules for specifying
  - with enhanced generic naming active [152](#), [189](#)

profile ownership

- delegating for FACILITY class [211](#)

profiles

- database
  - synchronizing [434](#)
- in RRSFDATA class
  - controlling automatic direction [437](#)
- mapping
  - for UIDs and GIDs [526](#)
- PHRASESYNC
  - controlling password and password phrase synchronization [435](#)
- PWSYNC
  - controlling password and password phrase synchronization [435](#)
- UNIXMAP class
  - for UIDs and GIDs [526](#)

program

- conditional access to SERVAUTH class [194](#)

program access to data sets (PADS)

- choosing PADCHK or NOPADCHK option [314](#)
- examples [322](#)
- in BASIC program security mode [310](#)
- in ENHANCED program security mode [315](#)

PROGRAM class

- description [685](#)
- examples [322](#)
- protecting DSMON [25](#), [58](#)
- SIGVER segment fields [209](#)
- specifying with WHEN operand on PERMIT command [194](#)

program control

- activating or deactivating [132](#)
- by system identifier (SMFID)

program control (*continued*)

- by system identifier (SMFID) (*continued*)
  - example of setting up [325](#)
  - overview [305](#)

- during authorization checking [722](#)

- ENHANCED program security mode [315](#)

- example of command to activate [324](#)

- examples [322](#)

- informational messages [318](#)

- IP addresses, protecting with [314](#)

- maintaining a clean environment [306](#)

- migrating from BASIC to ENHANCED mode [308](#)

- overview [301](#)

- program security modes [302](#)

- protecting program libraries [309](#)

- SERVAUTH resources, protecting with [314](#)

- using MAIN or BASIC attributes [316](#)

program dump

- protecting with data set profiles [230](#)

- protecting with EXECUTE authority [230](#)

- protecting with FACILITY profiles [230](#)

- using non-RACF methods to control access to [232](#)

program library

- defining data set profile to protect [324](#)

- examples of protecting [322](#)

- protecting, overview [309](#)

program properties table report

- from DSMON [362](#)

program security modes

- overview [302](#)

program signing

- overview [327](#)

- task roadmap [330](#)

program verification

- task roadmap [338](#)

program-accessed data set [157](#)

programs

- protecting [301](#)

- using with RACF

- Db2 [251](#)

propagation

- across a network [461](#)

- across nodes

- in NJE network [481](#)

- controlling a user ID [457](#), [475](#)

- definition [461](#)

- in an NJE environment [475](#)

- PROPCNTL profiles [457](#)

- support for JES user identification [366](#)

PROPCNTL class

- activating [457](#)

- and &RACLNDE variable [457](#)

- and RACF variables [475](#)

- controlling user ID propagation [457](#)

- description [685](#)

- SETOPTS RACLIST processing [457](#)

PROTECT parameter (JCL DD statement)

- for a new data set [178](#)

- parameters related to RACF [365](#)

- planning for the use of [36](#)

- protecting non-VSAM data sets [162](#)

- protecting tape volumes and tape data sets [181](#)

- tape data sets [180](#)

- to protect data set with discrete profile [151](#)



- PROTECT parameter (JCL DD statement) *(continued)*
  - when protecting tape data sets [172](#)
- PROTECTALL operand
  - SETROPTS command [119](#), [156](#)
  - with generic profile checking [156](#)
- PROTECTALL processing
  - activating or deactivating [119](#)
- PROTECTED user attribute [74](#)
- protected user IDs
  - and JES batch jobs [458](#)
  - and started procedures [141](#)
  - description [74](#)
  - with z/OS UNIX [516](#)
- protecting
  - a multivolume tape data set [171](#)
  - all data sets [119](#)
  - batch user IDs [458](#)
  - cataloged tape data set [171](#)
  - catalogs [166](#)
  - consoles [228](#)
  - DASD data sets [163](#)
  - data on tape [168](#)
  - data sets [18](#), [147](#)
  - data sets that have single-qualifier data set names [148](#)
  - data sets with a dummy group name [150](#)
  - data sets with discrete profiles [150](#)
  - data sets with discrete profiles through ADSP attribute [62](#)
  - data sets with generic profiles [151](#)
  - DFP-managed temporary data sets [224](#)
  - duplicate-named data sets residing on different volumes [161](#)
  - existing data on tape [171](#)
  - file system resources in the z/OS UNIX environment [530](#)
  - GDG data sets [161](#)
  - general resources [185](#)
  - group data sets [149](#)
  - group terminals [92](#)
  - JES batch user IDs [458](#)
  - LLA-managed data sets [234](#)
  - multivolume data sets with discrete profiles [162](#)
  - new data on tape [172](#)
  - non-VSAM data sets using JCL PROTECT parameter [162](#)
  - non-VSAM data sets using the JCL SECMODEL parameter [162](#)
  - programs [301](#)
  - RACF database [369](#)
  - started procedure user IDs [141](#)
  - tape volumes [172](#)
  - terminals [225](#)
  - TSO resources [507](#)
  - uncataloged tape data set [171](#)
  - user IDs [74](#)
  - vector facility [229](#)
  - z/OS UNIX files [530](#)
  - z/OS UNIX user IDs [516](#)
- PROXY segment
  - FACILITY profile
    - FIELD profile names [209](#)
  - field-level access checking [201](#)
  - user profile
    - FIELD profile names [209](#)
- PSF.DPAGELBL resource [247](#)
- PSFMPL class

- PSFMPL class *(continued)*
  - description [685](#)
  - OPERATIONS attribute allows access [59](#)
  - protecting PSF functions [247](#)
- PTKTDATA class
  - activating [282](#)
  - APPC application profile name [284](#)
  - application names [286](#)
  - CICS application profile name [284](#)
  - defining profiles [282](#)
  - description [685](#)
  - example of defining a profile [289](#)
  - FIELD profile names [209](#)
  - IMS application profile name [284](#)
  - MVS batch job profile name [284](#)
  - profile names [283](#)
  - protecting PassTickets [282](#)
  - SETROPTS RACLIST processing [282](#)
  - TSO profile name [285](#)
  - z/OS UNIX applications [286](#)
  - z/VM profile name [286](#)
- PTKTVAL class
  - description [690](#)
- PWSYNC operand
  - RACLINK command
    - controlling use of [435](#)
- PWSYNC resource
  - in RRSFDATA class [435](#)

## Q

- QCICSPSB class
  - description [687](#)
- QIMS class
  - description [689](#)
- QMF form [390](#)
- QRECOVER
  - accesses required [610](#)
- quarantine
  - user IDs [81](#)

## R

- R\_admin callable service
  - controlling the use [606](#)
- R\_auditx callable service
  - controlling the use [606](#)
- R\_cacheserv callable service
  - controlling the use [606](#)
- R\_datalib callable service
  - controlling the use [606](#)
  - description [546](#)
- R\_dcekey callable service
  - controlling the use [607](#)
- R\_dceruid callable service
  - controlling the use [608](#)
- R\_GetInfo callable service
  - controlling the use [608](#)
- R\_PKIServ callable service
  - administrative functions [611](#)
  - controlling the use [608](#)
  - end-user functions [608](#)
  - FACILITY class resources that protect [609](#), [611](#)

R\_PKIServ callable service (*continued*)

PKISERV class resources that protect [612](#)

R\_proxyserv callable service

controlling the use [614](#)

R\_ticketserv callable service

controlling the use [614](#)

RACDBULD

member of SYS1.SAMPLIB [382](#)

RACDBUQR

member of SYS1.SAMPLIB [382](#)

RACDBUTB

member of SYS1.SAMPLIB [382](#)

RACDCERT command

administering digital certificates [548](#)

controlling access to [549](#)

LISTMAP option [573](#)

MULTIID option [578](#)

NOTRUST option [564](#)

TRUST option [564](#)

RACF

and z/OS UNIX [513](#)

database unload utility (IRRDBU00) [371](#)

remove ID (IRRRID00) utility [391](#)

using with other programs

Db2 [251](#)

RACF authorization

versus password protection for VSAM data sets [165](#)

RACF commands

attributes and authorities required [699](#)

effecting a VLF cache flush [356](#)

for group administration [14](#)

for user administration [13](#)

logging attempts to issue [2](#)

operator

protecting with OPERCMDS profiles [240](#)

summary of [693](#)

to display information from RACF profiles [26](#)

to search for RACF profile names [28](#)

using [12](#)

using exits to perform additional authorization checking [22](#)

RACF database

backup RACF database [368](#)

considerations for [368](#)

coordinating profile updates [355](#)

multiple data sets [368](#)

protecting PassTicket keys [287](#)

protection of [369](#)

sample DDL statements [382](#)

sharing [370](#)

sharing among z/OS and z/VM systems [10](#)

sharing data using RRSF [370](#)

switching to alternate [368](#)

RACF exits report

from DSMON [363](#)

RACF implementation

organizing [33](#)

RACF indication

for data sets [151](#)

RACF options

listing system-wide

example [357](#)

selecting [21](#)

RACF options (*continued*)

selecting system-wide with SETROPTS command [105](#)

using [105](#)

RACF profile

description [6](#)

listing information from [26](#)

logging attempts to modify [2](#)

recording statistics in [26](#)

searching for [28](#)

RACF protection

activating or deactivating for general resource class [114](#)

bypassing during system access of system data sets [166](#)

enforcing during system access of system data sets [166](#)

need for [1](#)

tailoring [21](#)

RACF report writer

debugging your security arrangements [716](#), [717](#)

performance [3](#)

stabilization [3](#), [25](#)

using [2](#)

RACF variable

&RACLNDE profile [218](#)

and PROPCNTL class [475](#)

defining with RACFVARS profiles [216](#)

example [218](#)

using [217](#)

using in profile names [216](#)

RACF-indicated data set

definition of [151](#)

RACFEVNT class

description [685](#)

RACFHC class

description [685](#)

RACFICE procedure

[375](#)

RACFVARS class

&RACLNDE profile [482](#)

activating [217](#), [482](#)

analyzing profiles from SEARCH [30](#)

and NODES class [475](#)

and NODES profiles [471](#)

and PROPCNTL class [475](#)

choosing [188](#)

defining RACF variables [216](#)

description [685](#)

local nodes [482](#)

SETROPTS RACLIST processing [217](#), [482](#)

UACC authorities [216](#)

using [217](#)

RACGLIST class

and IRRDBU00 [372](#)

description [685](#)

saving in-storage profiles [238](#)

RACHCMBR class

description [685](#)

RACLINK command

controlling access to [435](#)

DEFINE operand

controlling use of [435](#)

PWSYNC operand

controlling use of [435](#)

RACLIST operand

SETROPTS command [125](#), [127](#)

RACONVRT EXEC



- RACONVRT EXEC (*continued*)
  - helps to convert SYS1.UADS entries to RACF user profiles [55, 56](#)
- RACROUTE REQUEST=AUTH macro
  - for authorizing access to resources [718](#)
  - for tapes [182](#)
  - tailoring with installation exit routine [21](#)
- RACROUTE REQUEST=DEFINE macro
  - for tapes [182](#)
  - tailoring with installation exit routine [21](#)
  - to define tape volumes to RACF [170](#)
- RACROUTE REQUEST=EXTRACT macro
  - and field-level access checking [200](#)
- RACROUTE REQUEST=FASTAUTH macro
  - authorization checking [732](#)
  - tailoring with installation exit routine [22](#)
- RACROUTE REQUEST=LIST macro
  - RACLIST [238](#)
  - tailoring with installation exit routine [22](#)
- RACROUTE REQUEST=VERIFY macro
  - brief description of RACF processing [738](#)
  - tailoring with installation exit routine [21](#)
  - using with profiles in the APPL class [223](#)
- RACROUTE REQUEST=VERIFYX macro
  - brief description of RACF processing [738](#)
  - tailoring with installation exit routine [21](#)
- RACSLUNK security label [471](#)
- RAUDITX class
  - description [688](#)
- RCICSRES class
  - description [687](#)
- RDATA LIB class
  - description [685](#)
- RDEFINE command
  - for PTKTDATA profiles [282](#)
  - to define tape volumes to RACF [170](#)
- READ access authority
  - as related to RACF commands [703](#)
  - editing a data set to which you have [367](#)
  - for general resources [193](#)
  - to a tape volume [179](#)
- read-only auditor
  - defining
    - example [359](#)
- real data set names option
  - effect on single-qualifier names [148](#)
- REALDSN operand
  - SETROPTS command [122](#)
- REALM class
  - description [690](#)
  - RRSF consideration [434](#)
- RECEIVE command (TSO)
  - auditing when used [248](#)
  - controlling use [511](#)
- receiving data
  - auditing [248](#)
- reconnecting to existing TSO session from another terminal [227](#)
- recording
  - bypassing for statistics [118](#)
  - real data set names [122](#)
  - REQUEST=VERIFY processing statistics [116](#)
  - statistics for resources [118](#)
  - statistics in RACF profiles [26](#)

- reducing
  - I/O requests to the RACF database [370](#)
- REFRESH operand
  - SETROPTS command [129](#)
- refreshing
  - data set profiles [717](#)
  - global access checking lists [130](#)
  - in-storage generic profile lists [130](#)
  - in-storage profiles
    - how to avoid [89](#)
  - profiles for SETROPTS RACLIST processing [498](#)
  - SETROPTS GENLIST processing [127](#)
  - SETROPTS RACLIST processing
    - for TSO general resource classes [509](#)
- register job
  - protecting with JESJOBS profiles [463](#)
- register jobs
  - controlling [463](#)
- rekeying a private key [586](#)
- remote mode for an RRSF node
  - description [407](#)
- remote node
  - command direction [413](#)
- remote workstations
  - protecting [491](#)
- remove ID (IRRRID00) utility [391](#)
- renaming a data set
  - multivolume with generic profile [163](#)
- renewing a digital certificate [585](#)
- replacing a private key [586](#)
- replay protection for PassTickets, bypassing [290](#)
- report output from IRRDBU00 [391](#)
- report writer
  - using the RACF [25](#)
- reports
  - based on the database unload utility (IRRRID00) [378](#)
  - produced by DSMON [361](#)
- REQCERT
  - accesses required [610](#)
- REQRENEW
  - accesses required [610](#)
- REQUEST=AUTH preprocessing exit
  - during failsoft processing [367](#)
  - to prevent access to resources [367](#)
- REQUEST=DEFINE preprocessing installation exit
  - during failsoft processing [368](#)
  - overriding normal RACF authorization [149](#)
- REQUEST=LIST
  - exit description [22](#)
  - listing programs authorized to issue [363](#)
  - using the exits [214](#)
- REQUEST=VERIFY
  - bypassing collection of statistics [116](#)
  - exit problems [739](#)
  - listing programs authorized to issue [363](#)
  - processing description [738](#)
- requirements
  - encryption
    - PassTicket [287](#)
- resetting password phrases
  - delegating authority for [655, 660](#)
- resetting passwords
  - delegating authority for [660](#)

- RESGROUP operand
  - RLIST command [213](#)
- residual IDs
  - using IRRRID00 utility to find [396](#)
- resource
  - deciding what to protect [35](#)
  - protecting [18](#)
  - rules for naming profiles
    - with enhanced generic naming active [152](#), [189](#)
- resource access authority
  - summary of authorities and commands [703](#)
- resource class
  - defining
    - overview [19](#)
- resource group class
  - defining
    - overview [19](#)
  - SETROPTS RACLIST processing [215](#)
- resource group profiles
  - choosing [188](#)
  - description [212](#)
- resource groups
  - rules for multiple profiles
    - default [214](#)
    - user exits [214](#)
- resource member class
  - defining
    - overview [19](#)
- resource profile
  - ownership [20](#)
- RESOWNER field
  - compared to OWNER field [20](#)
  - DFP segment of data set profile [500](#)
- RESPOND
  - accesses required [610](#)
- RESTRICTED attribute
  - defining [75](#)
  - description of [17](#), [62](#)
- restricted user IDs [75](#)
- RESTRICTED.FILESYS.ACCESS profile [531](#)
- restricting
  - access to SVC dumps containing passwords [366](#)
  - use of //DD DATA statement [366](#)
  - user IDs [75](#)
  - user's access to resources [62](#)
- Restricting execute access
  - file system [534](#)
  - TFS [534](#)
  - zFS [534](#)
- RESUME operand
  - ALTUSER command [111](#)
  - CONNECT command [90](#)
  - to activate a previously revoked user ID [109](#)
- resuming user IDs
  - delegating authority for [660](#)
- RETAIN field
  - DLFCLASS profile [235](#)
- retiring a private key [586](#)
- RETPD operand
  - to specify security retention period for data sets [178](#)
- reverse MAC (mandatory access checking)
  - for outbound work [483](#)
- REVOKE
  - accesses required [610](#)
- REVOKE attribute
  - description of [16](#), [61](#)
  - listing users with [67](#)
- REVOKE operand
  - CONNECT command [90](#)
- REVOKE suboperand
  - PASSWORD operand
    - SETROPTS command [109](#)
- revoked user ID
  - using the RESUME operand to activate a [109](#)
- revoking
  - IBMUSER user ID [358](#)
  - preventing [74](#)
  - the ADSP attribute [62](#)
  - unused user ID [111](#)
  - user ID based on consecutive incorrect passwords and password phrases [109](#)
  - user's access to system [61](#)
  - user's access to the system [77](#)
- REXX RACVAR function package
  - determining current security label [103](#)
- RIMS class
  - description [689](#)
- RJE consoles
  - protecting [491](#)
- RJE workstation
  - protecting with FACILITY profiles [492](#)
  - protecting with JESINPUT profiles [493](#)
- RJP consoles
  - protecting [491](#)
- RJP workstation
  - protecting with FACILITY profiles [492](#)
- RLIST command
  - to list information in TVTOC of tape volume profile [175](#)
  - to list profiles in the DIGTCERT class [555](#)
- RMTOPS class
  - description [690](#)
- ROAUDIT attribute
  - as related to RACF commands [701](#)
  - description of [16](#), [58](#)
- RODMMGR class
  - description [690](#)
- ROLE class
  - description [691](#)
- rollover of a private key [586](#)
- RRSF
  - Unidirectional connections [407](#)
- RRSF (RACF remote sharing facility)
  - creating user IDs [57](#)
  - custom fields [653](#)
  - directed command
    - order considerations [416](#)
  - directing commands to incompatible systems [417](#)
  - introduction [370](#)
  - local mode [407](#)
  - local node [406](#)
  - multisystem node [406](#)
  - nodes [406](#)
  - remote mode [407](#)
  - remote node [406](#)
  - security categories
    - maintaining in an RRSF environment [97](#)
  - single-system node [406](#)
  - suppression of private-key information propagation [431](#)

- RRSFDATA class
  - controlling
    - access to RACLINK command [435](#)
    - access to RRSF functions [434](#)
    - password and password phrase synchronization [435](#)
    - use of AT operand [436](#)
    - use of RACLINK DEFINE operand [435](#)
    - use of RACLINK PWSYNC operand [435](#)
  - controlling automatic direction
    - of application updates [440](#)
    - of commands [437](#)
    - of password phrases [439](#)
    - of passwords [439](#)
  - description [685](#)
- RSCS node
  - control of inbound jobs or data [468](#)
- rules
  - establishing password case [106](#)
  - establishing password syntax
    - ISPF panel [12](#)
  - establishing password with special characters [106](#)
  - for data set qualifiers [152](#)
  - for defining data set profiles [147](#)
  - for defining generic profiles
    - with enhanced generic naming active [152](#), [189](#)
  - for generic data set profiles [152](#)
  - for generic profile checking of data sets [153](#)
  - for generic profile checking of general resources [190](#)
  - for high-level qualifiers of a data set name [153](#)
  - for naming groups [89](#)
  - for naming users [56](#)
  - for protecting group data sets [149](#)
  - for protecting user data sets [148](#)
  - resolving multiple profiles
    - default [214](#)
    - user exits [214](#)
- RUSER qualifier
  - in NODES profiles [470](#), [496](#)
- RVARSMBR class
  - description [686](#)
- RVARY command
  - protection [240](#)
  - setting password for processing [112](#)
- RVARYPW operand
  - SETROPTS command [112](#)

## S

- SAF (system authorization facility)
  - handling of JES RACROUTE macro calls [453](#)
  - ICHRTX00 router exit routine [719](#)
  - ICHRTX01 router exit routine [719](#)
  - propagation of security information
    - across a network [461](#)
    - from incoming jobs [455](#), [461](#)
  - router exits
    - description [719](#)
    - diagram of RACF router [723](#)
    - diagram of SAF router [723](#)
    - problems [739](#)
  - user token [720](#)
  - z/OS UNIX services [531](#)

- SAF callable services, authorizing applications that invoke [603](#)
- sample
  - ISPF panel [12](#)
  - QMF form [390](#)
  - report from IRRDBU00 output [391](#)
  - SQL queries for IRRDBU00 [382](#)
- SAVE command
  - requirements for issuing [367](#)
- SCDMBR class
  - description [686](#)
- SCEPREQ
  - accesses required [610](#)
- SCICSTST class
  - description [687](#)
- scope of a group
  - description [14](#)
  - resources that are within [63](#)
  - using the group tree report to show [362](#)
- scratch function
  - DASDVOL authority [167](#)
- SCRATCH function
  - to scratch data sets on a volume [167](#)
- scratch pool volume
  - definition [172](#)
  - predefining for tape data sets [177](#)
- scratching temporary data sets
  - when TEMPDSN class is active [224](#)
- SDSF (System Display and Search Facility)
  - general resource class [686](#)
  - protecting panels with OPERCMDS profiles [239](#)
  - WRITER profiles [494](#)
- SDSF class
  - description [686](#)
- SEARCH command
  - and RACFVARS profiles [30](#)
  - example to identify cataloged group data sets [94](#)
  - example to identify cataloged user data sets [78](#)
  - to find all data sets with expired security retention period [177](#)
  - to find profiles in the DIGTCERT class [555](#)
  - with CLIST option
    - to maintain security categories in an RRSF environment [97](#)
- searching
  - for RACF profile names [28](#)
- SECDATA class
  - assigning security categories to users [67](#)
  - assigning security levels to users [67](#)
  - description [686](#)
- SECLABEL class
  - and CONSOLE class [229](#)
  - and JESNEWS [488](#)
  - and PSF for z/OS [247](#)
  - and SMESAGE class [246](#)
  - and spool offload [488](#)
  - and surrogate job submission [456](#)
  - and TSO SEND command [510](#)
  - and WRITER class [483](#)
  - assigning security labels to users [67](#)
  - authorization checking [733](#)
  - blank label [471](#)
  - description [686](#)
  - planning considerations [103](#)

- SECLABEL class (*continued*)
  - protecting consoles [229](#)
  - protecting terminals [227](#)
  - RACSLUNK label [471](#)
  - relationship to SETROPTS MLS, MLACTIVE, and MLQUIET settings [737](#)
  - SETROPTS options for [133](#)
  - tape volume considerations [175](#)
  - unknown label [471](#)
- SECLABEL field
  - on TSO/E logon panel [511](#)
  - TSO segment of user profile [100](#)
- SECLABEL parameter
  - on JOB statement in JCL [364](#)
- SECLABELCONTROL operand
  - SETROPTS command [133](#)
- SECLBYSYSTEM operand
  - SETROPTS command [138](#)
- SECLJ qualifier
  - on NODES profiles [470](#)
- SECLMBR class
  - description [686](#)
- SECLS qualifier
  - on NODES profiles [470](#)
- SECMODEL parameter
  - on DD statement in JCL [365](#)
- SECMODEL parameter in JCL
  - protecting non-VSAM data sets [162](#)
- secondary language
  - installation defaults [125](#)
- securing resources
  - controlling job names [464](#)
- security
  - controlling job names [464](#)
- security administration
  - delegating [79](#)
  - introduction [1](#)
- security administrator
  - limits of authority of at the group level [63](#)
  - responsibilities during implementation planning [34](#)
  - role of [9](#)
  - tools to manage the RACF database [23](#)
  - tools to monitor and control RACF [23](#)
- security category
  - assigning to users [67](#)
  - assigning with SECDDATA profiles [67](#)
  - definition [5](#)
  - deleting UNKNOWN categories [97](#)
  - during authorization checking [721](#)
  - how RACF uses [21](#), [95](#)
  - maintaining categories in an RRSF environment [97](#)
  - tape volume [174](#)
  - understanding [96](#)
- security classification of users and data
  - activating or deactivating [131](#)
  - definition [5](#)
  - during authorization checking [721](#)
  - how RACF uses [21](#), [67](#), [95](#)
  - understanding [96](#)
- security console
  - sending warning messages to [3](#)
- security events
  - security events (*continued*)
    - z/OS UNIX auditing [537](#)
  - security implementation team
    - responsibilities [34](#)
    - selecting [33](#)
  - security information
    - sent across nodes
      - in NJE network [481](#)
  - security label
    - activating by system image [138](#)
    - assigning to users [67](#)
    - assigning with SECLABEL profiles [67](#)
    - authorization checking [733](#)
    - blank label [471](#)
    - checking for messages with DIRAUTH profiles [246](#)
    - compatibility mode [135](#)
    - console considerations [104](#)
    - consoles [229](#)
    - copying from model profile [37](#)
    - definition [5](#)
    - description [98](#)
    - enforcing multilevel security [135](#)
    - JES writer [483](#)
    - JES writer considerations [104](#)
    - planning considerations [103](#)
    - preventing copying to a lower security label [134](#)
    - protecting [133](#)
    - RACSLUNK label [471](#)
    - restricting access to file system resources [137](#)
    - restricting access to interprocess communication objects [137](#)
    - SETROPTS MLS option [37](#)
    - system data sets [711](#)
    - tape volume considerations [175](#)
    - tape volumes [174](#), [175](#)
    - terminal considerations [104](#)
    - terminals [227](#)
    - unknown label [471](#)
    - using name-hiding [138](#)
  - security level
    - assigning with SECDDATA profiles [67](#)
    - converting from LEVEL to SECLEVEL [97](#)
    - definition [5](#)
    - during authorization checking [721](#)
    - how RACF uses [21](#), [67](#), [95](#)
    - tape volume [174](#)
    - understanding [96](#)
  - security retention period
    - how RACF uses for tape data sets [178](#)
    - searching for all tape data sets with expired [177](#)
    - system-wide for tape data sets [123](#)
  - security topics for RACF
    - classroom courses [xxvii](#)
  - segment
    - APPCLU profile [209](#), [221](#)
    - CDTINFO segment
      - FIELD profile names [205](#)
    - CFDEF segment
      - FIELD profile names [205](#)
    - CICS segment
      - contents [48](#)
      - conversation security options [250](#)
      - FIELD profile names [205](#)

segment (*continued*)

- CICS segment (*continued*)
  - field-level access checking [510](#)
- CSDATA segment
  - contents [49, 87](#)
  - FIELD profile names [205](#)
  - IRRDBU00 utility [372](#)
- DCE segment
  - contents [49](#)
  - FIELD profile names [205](#)
- DFP segment
  - contents [49, 87](#)
  - DFSMSdss storage administrator [93](#)
  - FIELD profile names [206](#)
  - field-level access checking [501](#)
  - overriding default values [500](#)
  - RACF processing [501](#)
  - SMS-managed data sets [499, 500](#)
- DLFCLASS profile [206, 235](#)
- DLFDATA segment
  - contents [235](#)
  - FIELD profile names [206](#)
- EIM segment
  - FIELD profile names [206](#)
- group profile
  - contents of [86](#)
- ICSF segment
  - FIELD profile names [206](#)
- ICTX segment
  - FIELD profile names [206](#)
- KERB segment
  - contents [50](#)
  - FIELD profile names [207](#)
- LANGUAGE segment
  - contents [50](#)
  - FIELD profile names [207](#)
- LNOTES segment
  - contents [51](#)
  - FIELD profile names [207](#)
- MFA segment
  - FIELD profile names [207](#)
- MFPOLICY segment
  - FIELD profile names [207](#)
- NDS segment
  - contents [51](#)
  - FIELD profile names [207](#)
- NETVIEW segment
  - contents [51](#)
  - FIELD profile names [207](#)
- OMVS segment
  - contents [52, 87](#)
  - FIELD profile names [208](#)
  - IRRDBU00 utility [372](#)
  - list-of-groups checking [112](#)
- OPERPARM segment
  - contents [52](#)
  - FIELD profile names [208](#)
- OVMS segment
  - FIELD profile names [208](#)
- profile modeling [36](#)
- PROGRAM profile [209](#)
- PROXY segment
  - FIELD profile names [209](#)
- PTKTDATA profile [209](#)

segment (*continued*)

- SESSION segment
    - contents [221](#)
    - FIELD profile names [209](#)
  - SIGVER segment
    - FIELD profile names [209](#)
  - SSIGNON segment
    - FIELD profile names [209](#)
  - STARTED profile [209](#)
  - STDATA segment
    - FIELD profile names [209](#)
  - SVFMR segment
    - FIELD profile names [209](#)
  - TME segment
    - contents [88](#)
    - FIELD profile names [209, 210](#)
  - TSO segment
    - contents [54](#)
    - controlling access to fields in [202](#)
    - example [76](#)
    - FIELD profile names [210](#)
    - SECLABEL field [100](#)
    - user profile
      - contents [47](#)
  - WORKATTR segment
    - contents [55](#)
    - FIELD profile names [210](#)
- segments
- STDATA [142](#)
- selected data sets report
- from DSMON [364](#)
- selected user attribute report
- from DSMON [363](#)
- selected user attribute summary report
- from DSMON [364](#)
- SEND command (TSO)
- and SECLABEL class [510](#)
  - controlling with SMESAGE profiles [510](#)
- sending messages
- TSO SEND command
    - controlling with SMESAGE profiles [510](#)
- SERVAUTH class
- authorization checking [731](#)
  - conditional access [157, 194](#)
  - description [686](#)
- SERVER class
- description [686](#)
- session interval, VTAM
- maximum [132](#)
- SESSION segment
- APPCLU profile
    - contents [221](#)
    - conversation security options [250](#)
    - FIELD profile names [209](#)
  - field-level access checking [201](#)
  - maximum VTAM session interval [132](#)
  - password [132](#)
- SESSIONINTERVAL operand
- SETROPTS command [132](#)
- SESSKEY value for APPCLU class profiles [222](#)
- SETROPTS command
- EARLYVERIFY operand [455](#)
  - GENLIST operand [125](#)
  - guidelines for selected options [105](#)

- SETOPTS command (*continued*)
  - in-storage profiles [125](#)
  - RACLIST operand [125](#)
  - specifying system-wide RACF options with [105](#)
- SETOPTS GENLIST processing
  - activating or deactivating [126](#)
  - deactivating [126](#)
  - refreshing [127](#)
- SETOPTS MLS option
  - effect on copying of security label in profile modeling [37](#)
- SETOPTS RACLIST processing
  - activating for TSO general resource classes [509](#)
  - activating or deactivating [127](#)
  - APPCLU class, not used [222](#)
  - APPL class [223](#)
  - avoiding deadlocks [231](#)
  - CONSOLE class [229](#)
  - deactivating [128](#)
  - deadlocks, avoiding [231](#)
  - DEVICES class [234](#)
  - DIGTCERT class [566](#)
  - DIGTCRIT class [579](#)
  - DLFCLASS class [237](#)
  - FACILITY class (first profile) [211](#)
  - FACILITY class (program dumps) [231](#)
  - FIELD class [202](#)
  - how to avoid refreshing [89](#)
  - member classes [215](#)
  - MGMTCLAS class [498](#)
  - NODES class [482](#)
  - OPERCMDS class [240](#)
  - overview of refreshing profiles [129](#)
  - performance benefits [187](#)
  - profile size considerations [193](#)
  - PROPCNTL class [457](#)
  - PTKTDATA class [282](#)
  - RACFVARS class [217](#), [482](#)
  - refreshing for TSO general resource classes [509](#)
  - refreshing profiles for SMS classes [498](#)
  - refreshing, overview [129](#)
  - resource group classes [215](#)
  - SMESSAGE class [246](#), [510](#)
  - STORCLAS class [498](#)
  - TERMINAL class [213](#), [225](#)
  - UNIXPRIV class [527](#), [528](#)
  - VTAMAPPL class [247](#)
- SETOPTS STATISTICS option
  - maintaining two sets of statistics in a discrete resource profile [115](#)
- shared in-storage profile
  - SETOPTS GENLIST processing for [126](#)
  - SETOPTS RACLIST processing for [127](#)
- shared RACF database
  - and PassTicket [287](#)
  - sharing among z/OS and z/VM systems [10](#)
- shared user IDs
  - restricting access to resources [62](#)
- SHARED.IDS profile [516](#)
- shortcut keys [741](#)
- SID value
  - SMFPRMxx member of SYS1.PARMLIB [284](#), [285](#)
- signature verification
  - overview [327](#)
- signing, load modules
  - overview [344](#)
- signing, program
  - overview [327](#)
  - task roadmap [330](#)
- SIGVER segment
  - field-level access checking [201](#)
  - PROGRAM profile
    - FIELD profile names [209](#)
- SIMS class
  - description [689](#)
- single-qualifier data set name
  - how RACF prefixes during authorization checking [153](#)
  - protecting data set that has [122](#), [148](#)
- single-system node [406](#)
- SINGLEDSN operand
  - indicating tape volume contains one data set [177](#)
  - RDEFINE command
    - effect on tape volumes [177](#)
- size considerations for profiles [193](#)
- SMESSAGE class
  - activating [246](#), [510](#)
  - and SECLABEL class [246](#)
  - controlling the TSO SEND command [510](#)
  - defining profile [246](#)
  - defining profiles [510](#)
  - description [686](#)
  - protecting receipt of TSO messages [245](#)
  - SETOPTS RACLIST processing [246](#), [510](#)
  - UACC authorities [246](#)
- SMF (System Management Facility)
  - control of logging to data set [58](#)
  - listing RACF-generated records [25](#)
  - logging records to [2](#)
  - RACF unload utility for SMF data [24](#)
- SMF CONTROL file
  - and PTKTDATA profile [286](#)
- SMF system identifier
  - for PTKTDATA profile [284–286](#)
- SMFPRMxx member of SYS1.PARMLIB
  - and PTKTDATA profile [285](#)
- SMS (Storage Management Subsystem)
  - controlling use of additional resources [504](#)
  - controlling use of SMS classes [497](#)
  - DFP segment in RACF profiles [499](#)
  - general resource classes [497](#)
  - management classes
    - protecting with MGMTCLAS profiles [497](#)
  - RACF support for [497](#)
  - storage classes
    - protecting with STORCLAS profiles [497](#)
- SMS (Storage Management Subsystems)
  - general resource classes [690](#)
- SMS-managed data set
  - determining owner of [501](#)
  - DFP segment of data set profile [500](#)
  - DFP segment of group profile [499](#)
  - DFP segment of user profile [499](#)
  - password protection for [160](#)
- SNAME field, LNOTES segment [252](#)
- SOMDOBJs class
  - description [686](#)
- SPECIAL attribute
  - as related to RACF commands [699](#)



- SPECIAL attribute (*continued*)
  - description of [15, 57](#)
  - listing users with [67, 382](#)
  - suggestions for assigning [66](#)
- spool
  - protecting
    - job data sets [483](#)
    - SYSIN data sets [483](#)
    - SYSOUT data sets [483](#)
    - system data sets [460](#)
    - trace data sets [488](#)
  - restricting access [511](#)
- spool data sets
  - controlling access [489](#)
- spool offload
  - considerations for JES2 [488](#)
  - WRITER class [489](#)
- spool reload
  - JESINPUT profiles [489](#)
  - protecting with JESJOBS profiles [489](#)
  - protecting with NODES profiles [489](#)
- SQL query
  - sample statements for IRRDBU00 output [382, 390](#)
- SSIGNON operand
  - RALTER command [287, 288](#)
  - RDEFINE command [282, 287](#)
- SSIGNON segment
  - field-level access checking [201](#)
  - PTKTDATA profile
    - FIELD profile names [209](#)
- stage 3 of application identity mapping [524](#)
- standard access list
  - during authorization checking [721](#)
- standard naming conventions
  - when defining data set profiles [147](#)
- START command [143](#)
- START LLA command
  - controlling the use of [234](#)
- STARTED class
  - description [686](#)
  - FIELD profile names [209](#)
  - JES (job entry subsystem) entry [453](#)
  - setting up [142](#)
  - STARTED profile names [143](#)
  - STDATA segment [142](#)
- started procedure
  - bypassing security classification checking [142](#)
  - considerations [144](#)
  - defining
    - using the STARTED class [142](#)
    - using the started procedures table (ICHRIN03) [144](#)
  - effect on SURROGAT checking [457](#)
  - failsoft processing for [367](#)
  - granting access to during authorization checking [719](#)
  - identifying by user ID [3](#)
  - preventing logon and revocation of user IDs [74](#)
  - using [141](#)
- started procedures table (ICHRIN03)
  - creating [144](#)
  - dynamic, using the STARTED class [142](#)
  - JES (job entry subsystem) entry [453](#)
- started procedures table report
  - from DSMON [363](#)
- statistics
  - statistics (*continued*)
    - bypassing recording of [118](#)
    - bypassing REQUEST=VERIFY processing [116](#)
    - maintaining [115](#)
    - recording for classes [118](#)
    - recording for REQUEST=VERIFY processing [116](#)
    - recording in RACF profiles [26](#)
    - using RACF to keep [3](#)
  - statistics collection
    - using SETROPTS STATISTICS [115](#)
  - STATUS suboperand
    - RVARYPW operand (SETROPTS command) [112](#)
  - STDATA segment
    - field-level access checking [201](#)
  - STARTED profile
    - FIELD profile names [209](#)
  - STORCLAS class
    - description [691](#)
    - protecting SMS storage classes [497](#)
    - SETROPTS RACLIST processing [498](#)
  - STORCLAS parameter (JCL DD statement)
    - parameters related to RACF [365](#)
  - subject's and issuer's name filters [575](#)
  - subject's name filters [574](#)
  - subject's X.509 distinguished name [572](#)
  - SUBMIT command (TSO)
    - controlling [366](#)
    - controlling job submission [462](#)
    - controlling register job [463](#)
    - IKJEFF53 installation exit [462](#)
  - submitter information
    - for local jobs [475](#)
    - for NJE jobs [475](#)
    - from not trusted nodes [475](#)
    - from trusted nodes [475](#)
  - submitter validation [477](#)
  - submitting jobs
    - allowing another user to submit jobs for you [467](#)
    - controlling [462](#)
    - surrogate users [455, 467](#)
  - SUBSYSNM class
    - description [691](#)
  - summary
    - defining a RACF group [93](#)
    - defining general resources [185](#)
    - defining users [75](#)
    - deleting groups [94](#)
    - deleting users [77](#)
    - of RACF authorities [693](#)
    - of RACF commands [693](#)
  - summary of changes [xxix, xxx](#)
  - superuser authority [515](#)
  - SUPERUSER.FILESYS authority, overriding [532](#)
  - SUPERUSER.FILESYS.ACLOVERRIDE profile [532](#)
  - SURROGAT class
    - activating [457](#)
    - authorizing surrogate users [456](#)
    - controlling TSO SUBMIT command [366](#)
    - defining profiles [456](#)
    - description [686](#)
    - migrating from \$SUBMIT.userid FACILITY class profiles [456](#)
    - RACF variable example [219, 220](#)
    - setting up surrogate users [455, 467](#)

- SURROGAT class (*continued*)
  - started task [457](#)
  - UACC authorities [456](#)
- surrogate job submission
  - across NJE networks [471](#)
  - and SECLABEL class [456](#)
  - authorizing [455](#), [467](#)
- surrogate user
  - authorizing [467](#)
  - authorizing with SURROGAT profiles [455](#), [456](#)
- SVC dump
  - restricting access to dumps containing passwords [366](#)
- SVFMR segment
  - field-level access checking [201](#)
  - general resource profile
    - FIELD profile names [209](#)
- SWITCH suboperand
  - RVARYPW operand (SETROPTS command) [112](#)
- switching
  - to alternate RACF databases [368](#)
- synchronization
  - of database profiles
    - establishing [434](#)
- SYS1.HELP data set
  - global access checking table entries for SYS1.HELP [198](#)
- SYS1.PARMLIB data set
  - CONSOLxx member [228](#)
  - SMFPRMxx member [285](#)
- SYS1.SAMPLIB
  - RACDBULD member [382](#)
  - RACDBUQR member [382](#)
  - RACDBUTB member [382](#)
- SYS1.UADS data set
  - converting with RACONVRT EXEC [55](#), [56](#)
  - deleting user entry [78](#)
  - maintaining for certain users [54](#), [507](#)
- SYSAREA parameter
  - on OUTPUT statement in JCL [365](#)
- SYSAUTO class
  - description [686](#)
- SYSIN data sets
  - protecting [483](#), [484](#)
  - protecting with JESSPOOL profiles [483](#)
- SYSLOG data sets
  - protecting with JESSPOOL profile [488](#)
- SYSMVIEW class
  - description [686](#)
- SYSOUT data sets
  - authorizing NJE [476](#)
  - protecting [483](#), [484](#)
  - protecting with JESSPOOL profiles [483](#)
  - RACSLUNK security label [471](#)
  - undefined security label [471](#)
- SYSOUT requests
  - &USER value [459](#)
  - how verified [459](#)
- SYSOUT spool reload
  - JESINPUT profiles [489](#)
- sysplex
  - MCS
    - command authorization in [240](#)
- sysplex communication
  - data sharing option [696](#)
- sysplex data sharing option

- sysplex data sharing option (*continued*)
  - parameters [374](#)
  - performance [372](#)
- system data set
  - protecting [166](#)
  - security labels [711](#)
- System Display and Search Facility (SDSF) [686](#)
- system identifier (SMFID)
  - using for program control
    - example of setting up [325](#)
- system key
  - DSMON report [362](#)
- system operators
  - creating user IDs [57](#)
- system programmer
  - responsibilities during implementation planning [34](#)
- system report
  - from DSMON [362](#)
- system resources
  - checking the status of [361](#)
- system security
  - administering [9](#)
  - checking by using DSMON reports [361](#)
  - decentralizing administration [9](#)
- system-wide RACF options
  - activating or deactivating using SETROPTS command [105](#)

## T

- table space for Db2
  - creating [383](#)
- tables
  - dynamic started procedures
    - and STARTED class [142](#)
  - started procedures
    - creating [144](#)
    - dynamic [142](#)
- tape data set
  - activating or deactivating protection for [168](#)
  - activating protection for [123](#)
  - authority to access a protected [170](#)
  - authorization requirements for TAPEVOL and TAPEDSN options [179](#)
  - authorizing access to data set on tape volume with a TVTOC [173](#)
  - checking the security retention period [178](#)
  - command to protect an existing
    - example [171](#)
  - creating discrete tape volume profile [151](#)
  - deleting RACF protection [177](#)
  - description of TVTOC [175](#)
  - DFP-managed
    - preventing access to [120](#)
  - DFSMSshm considerations [180](#)
  - example of command to create generic profile for [172](#)
  - failsoft processing for [368](#)
  - maximum number of TVTOC entries [176](#)
  - maximum number of volumes [171](#), [176](#)
  - multivolume considerations [163](#)
  - predefining tape volume profiles for [177](#)
  - PROTECT parameter in JCL [180](#)
  - protecting [18](#), [147](#)
  - protecting a cataloged [171](#)



- tape data set (*continued*)
  - protecting an existing [171](#)
  - protecting an uncataloged [171](#)
  - protecting multivolume [171](#)
  - protecting new [172](#)
  - protecting with discrete profile through ADSP attribute [62](#)
  - protecting with PROTECT parameter in JCL [181](#)
  - protection for nonlabeled (NL) tapes [183](#)
  - protection with nonstandard labels (NSL) [182](#)
  - protection with TAPEVOL and TAPEDSN options [169](#)
  - providing password protection for [181](#)
  - RACF processing for a multivolume [181](#)
  - specifying DISP=DELETE [151](#)
  - system-wide security retention period [123](#)
- tape label
  - authorization requirements for [182](#)
  - bypassing tape label processing [182](#)
  - data set and volume protection for nonlabeled tapes [183](#)
  - data set and volume protection with nonstandard labels [182](#)
- tape volume
  - activating protection for [123](#)
  - authority to access a protected [170](#)
  - authorizing access to data set on a [173](#)
  - automatic TVTOC tape volume profile [176](#)
  - defining with a TVTOC [172](#)
  - defining without a TVTOC [174](#)
  - example of command define with a TVTOC [172](#)
  - example of command to define without a TVTOC [174](#)
  - example of command to delete access lists [174](#)
  - maximum number a data set can span [176](#)
  - maximum span [171](#)
  - nonautomatic tape volume profile [177](#)
  - OPERATIONS user's authority to create or destroy labels [59](#)
  - protecting [172](#)
  - protecting with PROTECT parameter in JCL [181](#)
  - protection and bypass label processing (BLP) [182](#)
  - protection for nonlabeled (NL) tapes [183](#)
  - protection with nonstandard labels (NSL) [182](#)
  - protection with TAPEVOL and TAPEDSN options [169](#)
  - providing password protection for data sets on [181](#)
  - RACF authorization checking for [719](#)
  - removing write-enable ring when user has READ access authority [180](#)
- tape volume profile
  - access authorities for [173](#)
  - containing a TVTOC [175](#)
  - contents of [176](#)
  - predefining for tape data sets [177](#)
- TAPEDSN operand
  - SETROPTS command [168](#)
- TAPEDSN options
  - effect on tape volume and tape data set protection [169](#)
- TAPEVOL class
  - activating [123](#), [168](#)
  - defining profiles with a TVTOC [172](#)
  - defining profiles without a TVTOC [174](#)
  - description [686](#)
  - example of using on RDEFINE command [172](#)
  - must be active for bypass label processing [182](#)
  - OPERATIONS attribute allows access [59](#)
- TAPEVOL class (*continued*)
  - RACF variable example [217](#)
  - recording statistics for [118](#)
  - UACC authorities [173](#)
- TAPEVOL options
  - effect on tape volume and tape data set protection [169](#)
- TCICSTRN class
  - description [687](#)
- technical support personnel
  - responsibilities during implementation planning [34](#)
  - role of [9](#)
- teleprocessing device
  - controlling allocation with DEVICES profiles [232](#)
- TEMPDSN class
  - activating [224](#)
  - description [686](#)
  - protecting DFP-managed temporary data sets [224](#)
- temporarily preventing significant RACF activity [134](#)
- temporary data sets
  - erasing [165](#)
  - global access checking table [120](#)
  - nonstandard names [120](#)
  - PROTECTALL [120](#)
  - protecting with TEMPDSN profiles [224](#)
- terminal
  - allowing access depending on terminal [157](#), [193](#)
  - controlling access to resources based on security levels [67](#), [96](#)
  - limiting access to system [227](#)
  - limiting when terminal can be used [73](#)
  - preventing the use of undefined [226](#)
  - protecting [225](#)
  - protecting with GTERMINL profiles [226](#)
  - protecting with SECLABEL profiles [227](#)
  - protecting with TERMINAL profiles [225](#)
  - RACF authorization checking for [730](#)
  - specifying group terminal option [92](#)
  - time and day-of-week access checking [73](#)
  - UACC authorities for undefined terminals [131](#)
- TERMINAL class
  - activating [213](#), [225](#)
  - description [686](#)
  - protecting terminals [225](#)
  - recording statistics for [118](#)
  - SETROPTS RACLIST processing [213](#), [225](#)
  - specifying with WHEN operand on PERMIT command [157](#), [193](#)
- terminal name
  - on a VTAM system [225](#)
- TERMINAL operand
  - SETROPTS command [131](#)
- TERMUACC operand
  - for universal groups [88](#)
  - specifying on ADDGROUP or ALTGROUP command [227](#)
- time of day
  - terminal can access system [73](#), [227](#)
  - user can access system [73](#)
- time range
  - PassTicket [289](#)
- timed PERMIT
  - providing [90](#)
- TIMS class
  - activating [213](#)
  - description [689](#)

- Tivoli
  - general resource class [691](#)
- Tivoli Service Desk
  - general resource classes [689](#)
- TME segment
  - data set profile
    - FIELD profile names [209](#)
  - field-level access checking [201](#)
  - general resource profile
    - FIELD profile names [210](#)
  - group profile
    - contents of [88](#)
    - FIELD profile names [209](#)
- TMEADMIN class
  - description [691](#)
- token
  - submitter information propagated from trusted nodes [458](#)
- TPUT macro
  - auditing when users receive data sent [248](#)
- trace data set
  - protecting with JESSPOOL profile [488](#)
- trademarks [746](#)
- transaction
  - authorization checking in CICS [732](#)
- translating
  - group names [478](#)
  - security labels [478](#)
  - user IDs [478](#)
- TRANSMIT command (TSO)
  - controlling use [511](#)
- TRUST option of the RACDCERT command [564](#)
- trusted attribute
  - authorization checking [719](#)
  - started procedure [142](#)
- TSO
  - defining PTKTDATA profiles [285](#)
  - using when RACF is deactivated [512](#)
  - VTAM generic resource name [285](#)
- TSO account number
  - protecting with ACCTNUM class [507](#)
  - specifying with TSO/E logon panel [55](#)
- TSO command
  - ALLOCATE
    - protecting data sets with discrete profiles [151](#)
    - using the PROTECT operand on [162](#)
    - using the SECMODEL operand on [162](#)
  - and RACF [511](#)
  - CANCEL
    - controlling job cancellation [464](#)
  - CONSOLE
    - and OPERPARM segment [52](#)
  - EDIT
    - using [367](#)
  - LISTBC
    - auditing [248](#)
  - LOGON with RECONNECT operand [227](#)
  - OUTPUT
    - controlling the use of [511](#)
  - RECEIVE
    - auditing [248](#)
    - controlling the use of [511](#)
  - SEND
    - controlling with SMESAGE profiles [510](#)
- TSO command (*continued*)
  - SUBMIT
    - controlling [366](#)
    - controlling job submission [462](#)
    - controlling register job [463](#)
    - IKJEFF53 installation exit [462](#)
  - TRANSMIT
    - controlling the use of [511](#)
- TSO line drop facility [227](#)
- TSO logon procedures
  - protecting with TSOPROC class [507](#)
- TSO messages
  - protecting with SMESAGE profiles [245](#)
- TSO performance groups
  - protecting with PERFGRP class [507](#)
- TSO resources
  - authorization checking for [510](#)
  - protecting [507](#)
- TSO segment
  - controlling access to fields in [202](#)
  - description [17](#)
  - field-level access checking [201](#)
  - overriding information in [55](#)
  - user profile
    - contents of [54](#)
    - example [76](#)
    - FIELD profile names [210](#)
    - field-level access checking [510](#)
    - SECLABEL field [100](#)
- TSO session
  - building from information in TSO segment [54](#)
  - failsoft processing for [368](#)
- TSO user attributes
  - moving from SYS1.UADS to RACF database [54](#)
- TSO user authorities
  - protecting with TSOAUTH class [507](#)
- TSO/E
  - general resource classes [691](#)
  - RACF support for [507](#)
- TSOAUTH class
  - activating [508](#)
  - authorization checking for [510](#)
  - considerations [509](#)
  - description [691](#)
  - protecting TSO user authorities [507](#)
  - SETOPTS RACLIST processing [509](#)
  - UACC authorities [508](#)
- TSOPROC class
  - activating [508](#)
  - authorization checking for [510](#)
  - considerations [509](#)
  - description [691](#)
  - protecting TSO logon procedures [507](#)
  - SETOPTS RACLIST processing [509](#)
  - UACC authorities [508](#)
- TVTOC (tape volume table of contents)
  - authorizing access to data set on tape volume with [173](#)
  - defining tape volume without a [174](#)
  - defining tape volumes with a [172](#)
  - description [175](#)
  - DFSMSHsm considerations [180](#)
  - failsoft processing for [368](#)
  - maximum number of data set entries [176](#)
- TVTOC operand

TVTOC operand (*continued*)

creating TVTOC for tape volume [177](#)

example of using on RDEFINE command [172](#)

## U

UACC (universal access authority)

ACCTNUM class [508](#)

checking what is specified for system data sets [364](#)

CONSOLE class [229](#)

contrasted with ID(\*) [6](#), [158](#), [366](#)

coverage of

batch jobs not associated with a RACF-defined user [157](#)

RACF-defined users only [366](#)

users not defined to RACF [157](#), [366](#)

DATASET class [157](#)

default for user when connected to a group [67](#)

DEVICES class [233](#)

DLFCLASS class [236](#)

during authorization checking [722](#)

during authorization checking for devices [732](#)

during authorization checking for terminals [731](#)

FACILITY class

LLA-managed data sets [234](#)

for data sets [157](#)

for general resources [192](#)

for system data sets [711](#)

for undefined terminals [131](#)

JESINPUT class [467](#)

overriding [192](#)

overriding for a data set profile [157](#)

PERFGRP class [508](#)

RACFVARS class [216](#)

SMESSAGE class [246](#)

specifying default for undefined terminals [226](#)

SURROGAT class [456](#)

TAPEVOL class [173](#)

TSOAUTH class [508](#)

TSOPROC class [508](#)

VTAMAPPL class [247](#)

WRITER class [494](#)

UCICSTST class

description [687](#)

UID mapping

and VLF [524](#)

UNIXMAP class profiles [526](#)

UIMS class

description [689](#)

UNAME field, NDS segment [252](#)

unauthorized access attempts

logging [2](#)

uncataloged data sets

preventing access to [120](#)

undefined user

capabilities on a RACF-protected system [45](#)

UNDEFINEDUSER operand

SETROPTS command [481](#)

Unidirectional or Bidirectional remote connections [407](#)

unit record device

controlling allocation with DEVICES profiles [232](#)

universal groups

defining [88](#)

deleting [403](#)

universal groups (*continued*)

listing members [372](#)

RACFICE reports [378](#)

removing members [403](#)

using database unload (IRRDBU00) [372](#)

using database unload (IRRRID00) [403](#)

UNIXMAP class

description [692](#)

performance considerations [524](#)

populating [525](#)

profiles for UIDs and GIDs [526](#)

UNIXPRIV class

CHOWN.UNRESTRICTED profile [527](#)

description [692](#)

managing z/OS UNIX privileges [527](#)

search directories [528](#)

SETROPTS RACLIST processing [527](#), [528](#)

unknown operator command

auditing [241](#)

protecting with OPERCMDS profiles [241](#)

unknown security categories

deleting [97](#)

unknown user token

for jobs submitted from a physical reader [459](#)

UPDATE access authority

as related to RACF commands [703](#)

for general resources [193](#)

to a tape volume [179](#)

USE group authority

as related to RACF commands [702](#)

description [91](#)

user

accountability of individual [3](#)

as owner of data set profile [150](#)

as owner of resource profile [20](#)

assigning user and group attributes [14](#)

attributes [5](#), [57](#)

authority required to define new user [92](#)

authority to access data set when not in access list [157](#)

authority to access general resource when not in access list [192](#)

authorizing to access resources [5](#)

classifying [95](#)

defining

summary of steps [75](#)

defining to RACF [13](#), [45](#)

defining to RACF using ICF [76](#)

delegating authority to list [655](#)

delegating authority to resume [660](#)

deleting

summary of steps [77](#)

educating system users [41](#)

encryption of RACF user passwords [139](#)

excluding from system [16](#)

forcing batch users to identify themselves to RACF [454](#)

identification with USER operand [367](#)

identifying by user ID [3](#)

limiting when user can log on [73](#)

maximum number connected to a group [85](#)

naming conventions for [56](#)

RACF commands for administration [13](#)

relationships within a group [39](#)

requirements to log on to TSO [511](#)

restricting access to resources [62](#)

- user (*continued*)
  - revoking access to system [61](#)
  - security classification of [21](#), [95](#)
  - sending warning messages to [3](#)
  - support for JES user ID propagation [366](#)
  - verification
    - checking when restarting jobs [365](#)
  - verifying
    - use of console [731](#)
    - use of IP address [731](#)
    - use of JES input device [731](#)
    - use of terminal [730](#)
- user attribute
  - description of various [57](#)
  - specifying [14](#)
- user authority for TSO
  - protecting [507](#)
- user data set
  - controlling creation of [149](#)
  - protecting [148](#)
- user filters
  - for distributed identity names [671](#)
- user ID
  - activating a previously revoked [109](#)
  - as high-level qualifier for data set [148](#)
  - assigning to batch user [38](#)
  - associating started procedure names with [141](#)
  - authentication [738](#)
  - contained [81](#)
  - controlling propagation of [457](#)
  - creating blocks of using CLIST [56](#)
  - deactivating an unused [111](#)
  - default user IDs [481](#)
  - delegating authority to list [655](#)
  - delegating authority to resume [660](#)
  - displaying from RACF database [24](#)
  - during authorization checking
    - for data sets [721](#)
  - early verification of, by JES [455](#)
  - extended password and user ID processing [109](#)
  - invalid
    - listing in data set access lists [382](#)
  - migrating to RACF [56](#)
  - preventing logon and revocation [74](#)
  - propagation
    - control in an NJE environment [475](#)
    - for jobs with no validated user ID [455](#)
    - when jobs are submitted [455](#)
  - protected [74](#)
  - rationale for using [1](#)
  - restricted [75](#)
  - revoking an unused [111](#)
  - revoking based on consecutive incorrect passwords and password phrases [109](#)
  - selecting [38](#)
  - specifying as prefix for data set with single-qualifier name [122](#)
  - translating [478](#)
  - using &RACUID for global access checking [197](#)
- user ID associations
  - approving [410](#)
  - defining
    - for other users [410](#)
    - for your user ID [410](#)
- user ID associations (*continued*)
  - deleting [411](#)
  - listing [411](#)
  - managed
    - defining [410](#)
- user identifiers (UIDs) [514](#)
- user IDs
  - ??????? [481](#)
  - +++++++ [481](#)
  - creating for RRSF users [57](#)
  - creating for system operators [57](#)
  - mapping profiles for [526](#)
  - mapping to UIDs [524](#)
  - mapping UID to [514](#)
  - security default [480](#)
  - suggestions for defining [56](#)
- user information
  - delegating authority to list [655](#)
- user interface
  - ISPF [741](#)
  - TSO/E [741](#)
- user limits for z/OS UNIX [515](#)
- USER parameter
  - on JOB statement in JCL [364](#)
- user profile
  - authority granted through group-level attributes [63](#)
  - authority of CLAUTH user to define [60](#)
  - contents [47](#)
  - controlling access to DFP segment [502](#)
  - description of [17](#), [45](#)
  - DFP segment [499](#)
  - for the IBM support personnel [367](#)
  - ownership of [57](#)
  - profiles [47](#)
  - retrieving SMS information [501](#)
  - user
    - contents [47](#)
- user structure [13](#)
- user token [720](#)
- USER.OMVS.\* profile (FIELD class) [52](#)
- USER.OMVS.HOME profile (FIELD class) [52](#)
- USER.OMVS.PROGRAM profile (FIELD class) [52](#)
- USERJ qualifier
  - on NODES profiles [470](#)
- USERS qualifier
  - on NODES profiles [470](#)
- using the NOINITSTATS option
  - for REQUEST=VERIFY processing statistics [116](#)
- utilities
  - brief overview [23](#)
  - IRRADU00 [24](#)
  - IRRDBU00 [23](#), [371](#)
  - IRRIRA00 [524](#)
  - IRRRID00 [23](#)
  - IRRRID00 [24](#), [391](#)
  - IRRUT100 [24](#)
  - IRRUT200 [24](#)
  - IRRUT400 [23](#)

## V

- Validated Boot for z/OS
  - introduction [327](#)
  - overview [344](#)

- validation
  - security [477](#)
- VCICSCMD class
  - description [687](#)
- vector facility
  - protecting with FACILITY profile [229](#)
- verification, program
  - task roadmap [338](#)
- VERIFY
  - accesses required [610](#)
- verifying
  - digital signatures of programs [327](#)
- virtual key ring [567](#)
- Virtual Lookaside Facility (VLF)
  - z/OS UNIX performance considerations [524](#)
- VLF (Virtual Lookaside Facility)
  - z/OS UNIX performance considerations [524](#)
- VLF cache
  - flushing with RACF commands [356](#)
- VMBATCH class
  - OPERATIONS attribute allows access [59](#)
- VMCMD class
  - OPERATIONS attribute allows access [59](#)
- VMMDISK class
  - OPERATIONS attribute allows access [59](#)
- VMNODE class
  - OPERATIONS attribute allows access [59](#)
- VMRDR class
  - OPERATIONS attribute allows access [59](#)
- volume authority
  - DASD volume [167](#)
- VSAM data set
  - comparison of password and RACF authorization requirements [165](#)
  - multivolume considerations [163](#)
  - protecting
    - using commands [153](#)
- VTAM (Virtual Telecommunications Access Method)
  - general resource class [686](#)
- VTAM application programs
  - protecting with VTAMAPPL profiles [246](#)
- VTAM generic resource name for TSO [285](#)
- VTAM LU 6.2 bind
  - activating APPCLU class [222](#)
  - controlling [221](#)
  - example of APPCLU class [222](#)
  - using APPCLU class [221](#), [222](#)
- VTAM password on bind
  - enforcing [221](#)
- VTAM session interval
  - maximum [132](#)
- VTAM session-level security
  - controlling [221](#)
- VTAM terminal
  - defining name for [225](#)
- VTAMAPPL class
  - activating [247](#)
  - defining profiles [247](#)
  - description [686](#)
  - protecting VTAM applications [246](#)
  - SETROPTS RACLIST processing [247](#)
  - UACC authorities [247](#)

## W

- warning message
  - number of days before password expires [109](#)
  - rationale for using [37](#)
  - unauthorized access attempt [3](#)
- WARNING suboperand
  - PASSWORD operand
    - SETROPTS command [109](#)
- WBEM class
  - description [686](#)
- WCICSRES class
  - description [687](#)
- WebSphere Application Server
  - providing automatic registration of digital certificates [582](#)
  - using digital certificates to access [566](#)
- WebSphere MQ
  - general resource classes [691](#)
- WHEN operand
  - PERMIT command
    - data set profiles [157](#)
    - general resource profiles [193](#)
    - specifying when terminal can access system [73](#)
- WHEN(APPCPORT) operand
  - of PERMIT command [157](#), [194](#)
- WHEN(CONSOLE) operand
  - on PERMIT command [157](#), [193](#)
- WHEN(CRITERIA) operand
  - on PERMIT command [194](#)
- WHEN(JESINPUT) operand
  - on PERMIT command [157](#), [193](#)
- WHEN(PROGRAM) operand
  - of PERMIT command [194](#)
- WHEN(SERVAUTH) operand
  - of PERMIT command [157](#), [194](#)
- WHEN(SYSID) operand
  - on PERMIT command [194](#)
- WHEN(TERMINAL) operand
  - on PERMIT command [157](#), [193](#)
- WIMS class
  - description [689](#)
- work entering system
  - run by a RACF-defined user [135](#)
- WORKATTR segment
  - field-level access checking [201](#)
  - user profile
    - contents of [55](#)
    - FIELD profile names [210](#)
- workstations
  - PassTicket [281](#)
- write-enable ring
  - removing when opening a tape data set for input [180](#)
  - when opening a tape data set for input [179](#)
- WRITER class
  - activating [494](#)
  - and SDSF [494](#)
  - and SECLABEL class [483](#)
  - authorizing outbound work [483](#)
  - controlling output destination [493](#)
  - defining profiles [493](#)
  - description [686](#)
  - spool offload [489](#)
  - UACC authorities [494](#)

writer, external  
access to spool data sets [484](#)

## X

X.500 directory information tree [571](#)  
X.509 issuer's distinguished name [572](#)  
X.509 subject's distinguished name [572](#)  
XCSFKEY class  
description [689](#)  
XFACILIT class  
description [686](#)

## Z

z/OS Network Authentication  
Service  
general resource classes [690](#)  
IRR.RTICKETSERV resource [615](#)  
IRRSPK00 [614](#)  
R\_ticketserv callable service [614](#)  
z/OS UNIX  
access control lists (ACLs) [530](#)  
and RACF [513](#)  
auditing security events [537](#)  
authorizing daemons [257](#)  
automatic assignment of unique UNIX identities [518](#)  
defining delegated resources [257](#)  
file system resources, restricting access [137](#)  
files  
GRPLIST option [112](#)  
protecting data [530](#)  
general resource classes [691](#)  
group identifiers (GIDs) [513](#)  
group ownership of files [529](#)  
performance considerations [524](#)  
permission bits [530](#)  
protected user IDs [516](#)  
SETROPTS MLFSOBJ in effect [137](#)  
setting user limits [515](#)  
sharing UNIX identities [516](#)  
superuser authority [515](#)  
user identifiers (UIDs) [514](#)  
Virtual Lookaside Facility (VLF) [524](#)  
z/OSMF  
general resource class [691](#)  
z/VM  
defining PTKTDATA profiles [286](#)  
finding information for RACF tasks [10](#)  
sharing a RACF database among z/OS and z/VM systems  
[10](#)  
ZMFAPLA class  
description [691](#)





Product Number: 5655-ZOS

SA23-2289-70

