November 1999

In this issue

3  Establishing or removing SDSF RACF security
10  Auditing your security with SAS/CPE from SMF
17  An ISPF dialog to manage catalog aliases – part 2
28  Maintaining RACF authorizations
52  Identifying orphan entries in RACF access lists
60  RACF news

© Xephon plc 1999
Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year’s subscription to RACF Update, comprising four quarterly issues, costs £190.00 in the UK; $290.00 in the USA and Canada; £196.00 in Europe; £202.00 in Australasia and Japan; and £200.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the August 1995 issue, are available separately to subscribers for £50.50 ($77.50) each including postage.

RACF Update on-line
Code from RACF Update can be downloaded from our Web site at http://www.xephon.com/racfupdate.html; you will need the user-id shown on your address label.

© Xephon plc 1999. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Establishing or removing SDSF RACF security

This REXX EXEC is designed to establish or remove RACF access to operators, systems programmers, or other users of SDSF. It defines entities to class OPERCMDS.

A RACLIST refresh is required to enable any changes made to OPERCMDS class. The assumption here is that the user who executes this EXEC has either the SPECIAL attribute or has CLAUTH to OPERCMDS class.

SOURCE CODE

```rexx
/*****************************  REXX  ********************************/
/*                                                                   */
/*  EXEC Name   : SDSFSECR                                           */
/*                                                                   */
/*                                                                   */
/*  Function    : This EXEC either establishes or removes SDSF       */
/*                RACF security                                      */
/*                                                                   */
/*  Description : This EXEC will either establish or remove SDSF     */
/*                RACF security. You will be prompted for one        */
/*                or the other.                                      */
/*                                                                   */
/*                Be aware that if you define a generic profile      */
/*                and a specific (or more specific) profile, the     */
/*                more specific profile will be checked before the   */
/*                generic one. For example, take the two following   */
/*                permit statements:                                */
/*                                                                   */
/*                PE JES2.CANCEL.BAT CL(OPERCMDS) ID(JOHN) ACC(READ) */
/*                PE JES2.**         CL(OPERCMDS) ID(BILL) ACC(READ) */
/*                                                                   */
/*                In the above case, whenever user BILL goes to      */
/*                access the resource JES2.CANCEL.BAT, he will not    */
/*                be successful. This is because the JES2.**          */
/*                resource will be checked AFTER                    */
/*                JES2.CANCEL.BAT                                  */
/*                                                                   */
/*                Be aware of this when setting these up.            */
/*                                                                   */
/*$                                                                  */
/*****************************  REXX  ********************************/
PARSE UPPER ARG helpreq
IF helpreq = '?' THEN
```

DO
  CALL HELPDISP helpreq
EXIT
END /* Set up basic stuff */

CALL GET_PROCESS /* (R)emove or (E)stablish */
CALL POSITIVE_NEGATIVE /* Prepare to turn on or off */
IF type = E THEN /* Establish security */
  DO
    CALL SETUP /* Set up basic stuff */
    CALL GET_PROCESS /* (R)emove or (E)stablish */
    CALL POSITIVE_NEGATIVE /* Prepare to turn on or off */
    IF type = E THEN
      DO
        CALL SETROPTS
        CALL RDEFINE
        CALL PERMIT_SYSTEMS /* Systems Programming access */
        CALL PERMIT_EMERGENCY /* Emergency user-id access */
        CALL PERMIT_NCC /* NCC access */
        CALL PERMIT_SYSOPER /* Operations access */
        CALL PERMIT_ALLELSE /* Leftovers */
        CALL SETROPTS_REFRESH /* Refresh */
      END
      ELSE
        DO
          CALL PERMIT_SYSTEMS /* Systems Programming */
          CALL PERMIT_EMERGENCY /* Emergency user-id access */
          CALL PERMIT_NCC /* NCC */
          CALL PERMIT_SYSOPER /* Operations */
          CALL PERMIT_ALLELSE /* Leftovers */
          CALL RDEFINE
          CALL SETROPTS
        END
      END
    END
  END
END /* All done */

ELSE
  DO
    CALL PERMIT_SYSTEMS /* Systems Programming */
    CALL PERMIT_EMERGENCY /* Emergency user-id access */
    CALL PERMIT_NCC /* NCC */
    CALL PERMIT_SYSOPER /* Operations */
    CALL PERMIT_ALLELSE /* Leftovers */
  END
  CALL RDEFINE
  CALL SETROPTS
END

EXIT /* All done */
/* */
/* Basic Set-up */
/* */

SETUP:
PARSE SOURCE . . execname . /* For error messages */
ERRORMSG = '*** '||execname||'-' /* Setup error prefix */
error_leng = LENGTH(errormsg) + 3 /* For multiple lines */
spacer = ' ' DO i = 1 to error_leng
  spacer = spacer||' '
END
RETURN

GET_PROCESS:
m1 = errormsg||'ØØI: Are you Removing (R) or' m1 = m1||' Establishing (E) security? (R|E)'
SAY m1
done = N DO UNTIL done = Y
  PARSE UPPER PULL type
SELECT
WHEN type = '' THEN DO
  SAY errmsg||'Ø1E: Invalid answer, retry'
  SAY ''
  ITERATE
END
WHEN type ^= 'E' & type ^= 'R' THEN DO
  SAY errmsg||'Ø1E: Invalid answer, retry'
  ITERATE
END
OTHERWISE
  done = Y
END
END
RETURN

POSITIVE_NEGATIVE:
  no = ''
  delete = ''
  rdefine = 'RDEFINE'
  acc_none = 'UACC(NONE)'
  acc_read = 'UACC(READ)'
  acc_update = 'UACC(UPDATE)'
  acc_control = 'UACC(CONTROL)'

IF type = R THEN DO
  no = 'NO'
  delete = 'DELETE'
  rdefine = 'RDELETE'
  acc_none = ''
  acc_read = ''
  acc_update = ''
  acc_control = ''
END

  classact = no||'CLASSACT'
  generic = no||'GENERIC'
  gencmd = no||'GENCMD'
  raclist = no||'RACLIST'
RETURN

SETROPTS:
/* */
/* SETROPTS Processing */
/* */
c.1 = "SETROPTS "||classact||"(SDSF WRITER JESSPOOL OPERCMDS)"
c.2 = "SETROPTS "||generic||"(SDSF WRITER JESSPOOL OPERCMDS)"
c.3 = "SETROPTS "||gencmd||"(SDSF WRITER JESSPOOL OPERCMDS)"
c.4 = "SETROPTS ||classact||"(CONSOLE)"
c.5 = "SETROPTS ||raclist||"(OPERCMD)

DO i=1 TO 5
   c.i
END
RETURN

RDEFINE:
c.1  = rdefine||" SDSF ISFCMD.** "||acc_read
c.2  = rdefine||" SDSF ISFOPER.SYSTEM "||acc_none
c.3  = rdefine||" SDSF ISFATTR.** "||acc_none
c.4  = rdefine||" SDSF ISFINIT.** "||acc_none
c.5  = rdefine||" SDSF ISFOPER ANYDEST.JES2 "||acc_read
c.6  = rdefine||" SDSF ISFCMD.FILTER.ACTION "||acc_none
c.7  = rdefine||" SDSF ISFCMD.FILTER.FINDLIM "||acc_none
c.8  = rdefine||" SDSF ISFCMD.FILTER.OWNER "||acc_none
c.9  = rdefine||" SDSF ISFCMD.FILTER.ABEND "||acc_none
c.10 = rdefine||" SDSF ISFCMD.FILTER.TRACE "||acc_none
c.11 = rdefine||" SDSF ISFCMD.FILTER.INITIATOR.JES2 "||acc_none
c.12 = rdefine||" SDSF ISFCMD.FILTER.PREFIX "||acc_none
c.13 = rdefine||" SDSF ISFATTR.OUTPUT.CLASS "||acc_update
c.14 = rdefine||" SDSF ISFATTR.OUTPUT.DEST "||acc_none
c.15 = rdefine||" WRITER JES2.LOCAL.** "||acc_none
c.16 = rdefine||" WRITER JES2.RJE.** "||acc_none
c.17 = rdefine||" WRITER JES2.NJE.KAISER.** "||acc_read
c.18 = rdefine||" JESSPOOL ** "||acc_none
c.19 = rdefine||" JESSPOOL KPM.***CMMM204.*.D.*.* "||acc_none
c.20 = rdefine||" JESSPOOL KPM.***CMMM204.*.GROUP.* "||acc_none
c.21 = rdefine||" OPERCMD JES2.** "||acc_none
c.22 = rdefine||" OPERCMD MVS.** "||acc_none
c.23 = rdefine||" OPERCMD JES2.MODIFY.BATOUT "||acc_none
c.24 = rdefine||" OPERCMD JES2.RELEASE.BATOUT "||acc_none
c.25 = rdefine||" OPERCMD JES2.CANCEL.BAT "||acc_none
c.26 = rdefine||" OPERCMD JES2.CANCEL.STC "||acc_none
c.27 = rdefine||" OPERCMD JES2.MODIFY.STCOUT "||acc_none

DO i=1 TO 27
   c.i
END
RETURN

PERMIT_SYSTEMS:
/*
/* Establish access for Systems Programming */
/*
c.1 = "PE ISFCMD.** CLASS(SDSF) ID(SYSTEMS) ACCESS(READ) "||delete
c.2 = "PE ISFOPER.SYSTEM CLASS(SDSF) ID(SYSTEMS) "
c.2 = c.2||"ACCESS(READ) "||delete
/*
c.3 = "PE ISFATTR.** CLASS(SDSF) ID(SYSTEMS) ACCESS(UPDATE) "||delete
c.4 = "PE ISFINIT.** CLASS(SDSF) ID(SYSTEMS) ACCESS(CONTROL) "||delete
c.5  = "PE ISFOPER.ANYDEST.JES2 CLASS(SDSF)"
c.5  = c.5||" ID(SYSTEMS) ACCESS(READ) "||delete
c.6  = "PE ISFATTR.OUTPUT.DEST CLASS(SDSF) ID(SYSTEMS) "
c.6  = c.6||"ACCESS(UPDATE) "||delete
c.7  = "PE JES2.NJE.KAISER.** CLASS(WRITER) ACCESS(READ) "
c.7  = c.7||"ID(SYSTEMS) "||delete
c.8  = "PE JES2.LOCAL.** CLASS(WRITER) "
c.8  = c.8||"ID(SYSTEMS) ACCESS(ALTER) "||delete
c.9  = "PE JES2.RJE.** CLASS(WRITER) ID(SYSTEMS) "
c.9  = c.9||"ACCESS(ALTER) "||delete
c.10 = "PE ** CLASS(JESSPOOL) ID(SYSTEMS) ACCESS(ALTER) "||delete
c.11 = "PE KPM0.*.CMMM2O4.*.D*. CLASS(JESSPOOL) ACCESS(READ) "
c.11 = c.11||" ID(SYSTEMS) "||delete
c.12 = "PE KPM0.*.CMMM2O4.*.GROUP.* CLASS(JESSPOOL) ACCESS(ALTER) "
c.12 = c.12||" ID(SYSTEMS) "||delete
c.13 = "PE JES2.** CLASS(OPERCMDS) ID(SYSTEMS) "
c.13 = c.13||"ACCESS(CONTROL) "||delete
c.14 = "PE MVS.** CLASS(OPERCMDS) ID(SYSTEMS) ACCESS(CONTROL) "
c.14 = c.14||"ACCESS(CONTROL) "||delete
c.15 = "PE JES2.CANCEL.BAT CLASS(OPERCMDS) ID(SYSTEMS) "
c.15 = c.15||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete
c.16 = "PE JES2.MODIFY.STCOUT CLASS(OPERCMDS) ID(SYSTEMS) "
c.16 = c.16||"ACCESS(CONTROL) WHEN(CONSOLE(SDSF)) "||delete
c.17 = "PE JES2.CANCEL.STC CLASS(OPERCMDS) ID(SYSTEMS) "
c.17 = c.17||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete

DO i=1 TO 17
   c.i
END
RETURN

PERMIT_NCC:
/*                                                                   */
/*  Establish access for NCC                                         */
/*                                                                   */
c.1  = "PE ISFCMD.** CLASS(SDSF) ID(NCC) ACCESS(READ) "||delete
c.2  = "PE ISFOPER.SYSTEM CLASS(SDSF) ID(NCC) "
c.2  = c.2||"ACCESS(READ) "||delete
c.3  = "PE ISFATTR.** CLASS(SDSF) ID(NCC) ACCESS(UPDATE) "||delete
c.4  = "PE ISFINIT.** CLASS(SDSF) ID(NCC) ACCESS(CONTROL) "||delete
c.5  = "PE ISFOPER.ANYDEST.JES2 CLASS(SDSF)"
c.5  = c.5||" ID(NCC) ACCESS(READ) "||delete
c.6  = "PE ISFATTR.OUTPUT.DEST CLASS(SDSF) ID(NCC) "
c.6  = c.6||"ACCESS(UPDATE) "||delete
c.7  = "PE JES2.NJE.KAISER.** CLASS(WRITER) ACCESS(READ) "
c.7  = c.7||"ID(NCC) "||delete
c.8  = "PE JES2.LOCAL.** CLASS(WRITER) "
c.8  = c.8||"ID(NCC) ACCESS(ALTER) "||delete
c.9  = "PE JES2.RJE.** CLASS(WRITER) ID(NCC) "
c.9  = c.9||"ACCESS(ALTER) "||delete
c.10 = "PE ** CLASS(JESSPOOL) ID(NCC) ACCESS(ALTER) "||delete

c.11 = "PE KPMØ.*.CMMM2Ø4.*.D*. CLASS(JESSPOOL) ACCESS(READ)"

c.11 = c.11||" ID(NCC) "||delete

c.12 = "PE KPMØ.*.CMMM2Ø4.*.GROUP.* CLASS(JESSPOOL) ACCESS(ALTER)"

c.12 = c.12||" ID(NCC) "||delete

c.13 = "PE JES2.** CLASS(OPERCMDS) ID(NCC)"

c.13 = c.13||"ACCESS(CONTROL) "||delete

c.14 = "PE MVS.** CLASS(OPERCMDS) ID(NCC) ACCESS(CONTROL)"

c.14 = c.14||"ACCESS(CONTROL) "||delete

c.15 = "PE JES2.CANCEL.BAT CLASS(OPERCMDS) ID(NCC)"

c.15 = c.15||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete

c.16 = "PE JES2.MODIFY.STCOUT CLASS(OPERCMDS) ID(NCC)"

c.16 = c.16||"ACCESS(CONTROL) WHEN(CONSOLE(SDSF)) "||delete

c.17 = "PE JES2.CANCEL.STC CLASS(OPERCMDS) ID(NCC)"

c.17 = c.17||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete

DO i=1 TO 17
  c.i
END

RETURN

PERMIT_SYSOPER:
/
/*
/ * Establish access for Operations
/ */
c.1 = "PE ISFCMD.** CLASS(SDSF) ID(SYSOPER) ACCESS(READ)"

c.2 = "PE ISFOPER.SYSTEM CLASS(SDSF) ID(SYSOPER)"

c.2 = c.2||"ACCESS(READ) "||delete

c.3 = "PE ISFATTR.** CLASS(SDSF) ID(SYSOPER) ACCESS(UPDATE)"

c.4 = "PE ISFINIT.** CLASS(SDSF) ID(SYSOPER) ACCESS(CONTROL)"

c.5 = "PE ISFOPER.ANYDEST.JES2 CLASS(SDSF)"

c.5 = c.5||"ID(SYSOPER) ACCESS(READ) "||delete

c.6 = "PE ISFATTR.OUTPUT.DEST CLASS(SDSF) ID(SYSOPER)"

c.6 = c.6||"ACCESS(UPDATE) "||delete

c.7 = "PE JES2.LOCAL.** CLASS(WRITER)"

c.7 = c.7||"ID(SYSOPER) ACCESS(ALTER) "||delete

c.8 = "PE JES2.RJE.** CLASS(WRITER) ID(SYSOPER)"

c.8 = c.8||"ACCESS(ALTER) "||delete

c.9 = "PE JES2.NJE.KAISER.** CLASS(WRITER) ACCESS(READ)"

c.9 = c.9||"ID(SYSOPER) "||delete

c.10 = "PE ** CLASS(JESSPOOL) ID(SYSOPER) ACCESS(ALTER)"

c.11 = "PE KPMØ.*.CMMM2Ø4.*.D*. CLASS(JESSPOOL) ACCESS(READ)"

c.11 = c.11||" ID(SYSOPER) "||delete

c.12 = "PE KPMØ.*.CMMM2Ø4.*.GROUP.* CLASS(JESSPOOL) ACCESS(ALTER)"

c.12 = c.12||" ID(SYSOPER) "||delete

c.13 = "PE JES2.** CLASS(OPERCMDS) ID(SYSOPER)"

c.13 = c.13||"ACCESS(CONTROL) "||delete

c.14 = "PE MVS.** CLASS(OPERCMDS) ID(SYSOPER) ACCESS(CONTROL)"

c.15 = "PE JES2.CANCEL.BAT CLASS(OPERCMDS) ID(SYSOPER)"

c.15 = c.15||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete

c.16 = "PE JES2.MODIFY.STCOUT CLASS(OPERCMDS) ID(SYSOPER)"

c.16 = c.16||"ACCESS(CONTROL) WHEN(CONSOLE(SDSF)) "||delete

c.17 = "PE JES2.CANCEL.STC CLASS(OPERCMDS) ID(SYSOPER)"
c.17 = c.17||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete

DO i=1 TO 17
  c.i
END
RETURN

PERMIT_ALLELSE:
/*                                                                    */
/* Establish access for everybody else                                 */
/*                                                                    */
c.1 = "PE ISFATTR.OUTPUT.DEST CLASS(SDSF) ID(APPJGK IBAGLM) "
c.1 = c.1||"ACCESS(UPDATE) "||delete
c.2 = "PE JES2.NJE.KAISER.** CLASS(WRITER) ACCESS(READ) "
c.2 = c.2||"ID(APPJGK IBAGLM) "||delete
c.3 = "PE KPMØ.*.CMMM2Ø4.*.D*. CLASS(JESSPOOL) ACCESS(READ) "
c.3 = c.3||" ID(APPJGK IBAGLM) "||delete
c.4 = "PE KPMØ.*.CMMM2Ø4.*.GROUP.* CLASS(JESSPOOL) ACCESS(ALTER) "
c.4 = c.4||" ID(APPJGK IBAGLM) "||delete
c.5 = "PE JES2.CANCEL.BAT CLASS(OPERCMDS) ID(APPSUP) "
c.5 = c.5||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete
c.6 = "PE JES2.MODIFY.BATOUT CLASS(OPERCMDS) ID(*) "
c.6 = c.6||"ACCESS(CONTROL) WHEN(CONSOLE(SDSF)) "||delete
c.7 = "PE JES2.RELEASE.BATOUT CLASS(OPERCMDS) ID(*) "
c.7 = c.7||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete
c.8 = "PE JES2.MODIFY.STCOUT CLASS(OPERCMDS) ID(APPJGK IBAGLM) "
c.8 = c.8||"ACCESS(CONTROL) WHEN(CONSOLE(SDSF)) "||delete
c.9 = "PE JES2.CANCEL.STC CLASS(OPERCMDS) ID(APPJGK IBAGLM) "
c.9 = c.9||"ACCESS(UPDATE) WHEN(CONSOLE(SDSF)) "||delete

DO i=1 TO 9
  c.i
END
RETURN

SETROPTS_REFRESH:
"SETROPTS REFRESH GENERIC(OPERCMDS)"
"SETROPTS REFRESH RACLST(OPERCMDS)"
RETURN

HELPDISP: PROCEDURE
/*                                                                    */
/* Description : This checks the incoming argument to see if it is     */
/* a '?'. If it is, then the top of the executing EXEC is searched     */
/* for a comment line starting with a slash-asterisk-dollar. From      */
/* the next line on the comments are displayed on the screen until    */
/* an asterisk-slash-dollar is found. The lines that are              */
/* displayed are stripped of the starting slash-asterisk and ending   */
/* asterisk-slash. Be aware that this function expects each and       */
/* every comment line (at the top of the EXEC) to begin and end       */
/* in this fashion. If they do not, unpredictable results will        */
/* occur. If no slash-asterisk-dollar is found this procedure will    */
Auditing your security with SAS/CPE from SMF

This application lists the contents of System Management Facilities (SMF) records in an easy-to-read format. SMF records reside in the SMF data file.

```plaintext
//REPORACF JOB EXP,'REPORACF',CLASS=W,MSGCLASS=O,MSGLEVEL=(1,1), // NOTIFY=DUNAND //DELDSNL EXEC PGM=IDCAMS,REGION=2048K //SYSPRINT DD SYSOUT=O //SYSIN DD * DELETE "EXPL69.REPORACF.LIST" //REPORACF EXEC SAS,REGION=8M, // WORK='200,50', // OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL' //SOURCLIB DD DSN=SAS.LOCAL.REPORTS,DISP=SHR //LIBRARY DD DSN=SAS.MXG.V1313.FORMATS,DISP=SHR //SASLIST DD DSN=EXPL69.REPORACF.LIST,DISP=(NEW,CATLG,DELETE), // UNIT=SYSDA,SPACE=(TRK,(5,9),RLSE),
```

© 1999. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (940) 455 7050, fax (940) 455 2492.
With this job, you can obtain:

- Reports that describe user and group activity.
- Reports that summarize system use and resource use.
- Reports that track the total use of a resource or that track the activity of a particular user.
- Reports to determine whether the password violation stabilizes over time, and at which terminal password violations are occurring.

The SMF input dataset consists of the following SMF record types:

- 20 – job initiation
- 30 – common address work data
- 80 – RACF processing
- 81 – RACF initialization
• 83 – RACF processing.

SMF records must firstly be dumped to a dataset that SAS/CPE can use as an input. Usually, the SMF dump program IFASMFDP is used to dump the SMF records. IFASMFDP dumps the SMF dataset (SYS1.MANX), which is a VSAM dataset, to a QSAM dataset.

We use SAS to help us analyse the data produced by SMF for RACF. This gives us the flexibility to subset the data based upon certain criteria and produce reports for the application developers to review.

SOURCE CODE

```
CREATES ONE */
/* SAS OBSERVATION FOR EACH RACF EVENT, DECODING THE VARIOUS SEGMENTS*/
/* IN A PARTICULAR RECORD INTO VARIABLES WITHIN THAT OBSERVATION, AND*/
/* SHOULD THEREBY OBLIVATE THE NEED TO ANALRACF. "0 ="Ø "' */
  1 ="1:OLD DATASET NAME" ' 
  2 ="2:NEW DATASET NAME" ' 
  3 ="3:ACCESS AUTH REQUESTED" ' 
  4 ="4:ACCESS AUTH ALLOWED" ' 
  5 ="5:DATASET LEVEL NUMBER" ' 
  6 ="6:RACF COMMAND-RELATED DATA" ' 
  7 ="7:DATA INSTALLATION-DEFINED DATA" ' 
  8 ="8:NAME USER-NAME" ' 
  9 ="9:RESOURCE NAME" ' 
 10 ="10:VOLUME SERIAL" ' 
 11 ="11:VOLUME SERIAL" ' 
 12 ="12:ID NAMES (PERMIT)" ' 
 13 ="13:FROM_RESOURCE NAME" ' 
 14 ="14:VOLUME VOLUME SERIAL +FVOLUME" ' 
 15 ="15:VOLSER VOLUME SERIAL" ' 
 16 ="16:OLDVOL VOLUME SERIAL" ' 
 17 ="17:CLASS NAME" ' 
 18 ="18:MENTITY MODEL RESOURCE NAME" ' 
 19 ="19:VOLUME SERIAL OF MODEL RESOURCES" ' 
 20 ="20:APPLICATION NAME" ' 
 21 ="21:CURRENT CLASS OPTIONS" ' 
 22 ="22:CLASS NAME FROM STATISTICS" ' 
 23 ="23:CLASS NAME FROM AUDIT" ' 
 24 ="24:RESOURCE NAME FROM ADDMEM" ' 
 25 ="25:RESOURCE NAME FROM DELMEM" ' 
 26 ="26:CLASS NAME FROM FCLASS" ' 
 27 ="27:CLASS NAME FROM CLCLASSACT" ' 
 28 ="28:CLASS NAME FROM CLAUTH" ' 
 29 ="29:APPLICATION DATA" ' 
 30 ="30:RACF DATASET STATUS" ' 
```
31 = '31:DATASET NAME FROM DATASET
32 = '32:PASSWORD INFORMATION (ENCODED)
33 = '33:FLAG BITS
34 = '34:CLASS NAME FROM GENERIC
35 = '35:CLASS NAME FROM GENCMD
36 = '36:CLASS NAME FROM GLOBAL
37 = '37:MODEL NAME
38 = '38:USERID/GROUP OWNING PROFILE
39 = '39:PROGRAM NAMES (PERMIT)
40 = '40:CATEGORY NAME ADDED(ADDSD,ALTSD,...)
41 = '41:CATEGORY NAME DELETED(ALTSD,ALTUSER)
42 = '42:CLASS NAME FROM RACLST
43 = '43:CLASS NAME FROM GENLIST
44 = '44:SEGMENT DATA, EXCEPT BASE
45 = '45:CLASS/LOGGING OPTIONS FROM SETROPTS
46 = '46:DATA SPECIFIED ON LOGSTR= ON RACROUTE
47 = '47
48 = '48:USERID TO WHOM DATA IS DIRECTED
49 = '49:USER NAME FROM ACEE
50 = '50:SECLABEL NAME TO BE ADDED TO PROFILE
51 = '51:SECLABEL NAME TO BE DELTD (PROFILE)
52 = '52:DSN AFFECTED BY A SECLABEL CHANGE
53 = '53:USER SECURITY TOKEN ICHRUTKN MAPPING
54 = '54:RESOURCE SECURITY TOKEN(RACHECK)
55 = '55
56 = '56
57 = '57
58 = '58
59 = '59
60 = '60
61 = '61
62 = '62
63 = '63:LINK VALUE TO CONNECT DS AFFECTED
64 = '64
65 = '65
66 = '66
67 = '67
68 = '68
69 = '69
70 = '70
71 = '71
72 = '72
73 = '73
74 = '74
75 = '75
76 = '76
77 = '77
78 = '78
79 = '79
80 = '80
options set=vqdllname "ehlapi32";
/*options set=vqtrace "0000ffff"; */
options comamid=ehlapi remote=a;
signon "c:\tab612\script\tabord.scr";
rsrIBUTE;
libname sasracf "psy69.racf.sasdata" disp=shr;
proc download inlib=sasracf outlib=sasracf;
run;
endrsrIBUTE;
signoff;

//SASJRACF JOB COM,'SASDIV',CLASS=W,MSGCLASS=0
//*
//DELDEB EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=* 
//SYsin DD *
DElLETE EXPL69.CPERACF
IF MAXCC <= 8 THEN SET MAXCC=Ø
/*
//SMFECLA EXEC PGM=IFASMFDP,REGION=4096K
//DUMPIN DD DISP=SHR,DSN=EXPL69.CPESMF
//RACF DD DSN=EXPL69.CPERACF,DISP=(,CATLG,DELETE),UNIT=SYSDA,
//DCB=(RECFM=VBS,LRECL=3276Ø,BLKSIZE=4096),
//SPACE=(CYL,(50,99),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
INDD(DUMPIN,OPTIONS(DUMP))
SID(LYO1)
OUTDD(RACF,TYPE(80,81,82,83))

//SASRACF EXEC SAS,REGION=8M.
//   OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SMF DD DSN=EXPL69.CPERACF,DISP=SHR
//REPORT DD DSN=SAS.LOCAL.REPORTS,DISP=SHR
//SASLIST DD SYSOUT=O
//SYSIN DD *

OPTIONS PAGESIZE=60 LINESIZE=132 ;
%CPSTART(MODE=BATCH,
   SYSTEM=MVS,
   ROOT=SAS.SAS69.TS450.CPE.,
   PDB=SAS.LOCAL.CPE.RACF.,
   DISP=OLD,
   ROOTSERV=,
   SHARE=N/A,
   MXGSRC=(:SAS.LOCAL.SOURCLIB "SAS.MXG.V1313.SOURCLIB"),
   MXGLIB=SAS.MXG.V1313.FORMATS
);

/* %INCLUDE SOURCLIB(TYPE37); */
/* %INCLUDE SOURCLIB(TYPE50); */
RUN;

%LET CMPRRC=.;
%CMPROCES(,
   COLLECTR=SMF,
   TOOLNM=MXG,
   _RC=CMPRRC
);

%PUT CMPROCES RETURN CODE IS &CMPRRC;
%LET CPREDRC=.;
%CPREDUCE(_RC=CPREDRC);
%PUT CPREDUCE RETURN CODE IS &CPREDRC;

/**** DAILY REPORTS *****/

%INCLUDE REPORT(OPTIONS);
%INCLUDE REPORT(HIER);

/*
/*
//DELFIN EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
DELETE EXPL69.CPERACF
IF MAXCC <= 8 THEN SET MAXCC=0

/*
//REPORTACF JOB EXP,'REPORTACF',CLASS=W,MSGCLASS=O,MSGLEVEL=(1,1),
//NOTIFY=DUNAND
//DELDSNL EXEC PGM=IDCAMS,REGION=2048K
//SYSPRINT DD SYSOUT=O
//SYSIN DD *
DELETE "EXPL69.REPORTACF.LIST"
*/
//REPORTACF EXEC SAS,REGION=8M,
//       WORK='200,50'.
//       OPTIONS='MEMSIZE=16M DMSBATCH BATCH TERMINAL'
//SOURCLIB DD DSN=SAS.LOCAL.REPORTS,DISP=SHR
//LIBRARY DD DSN=SAS.MXG.V1313.FORMATS,DISP=SHR
//SASLIST DD DSN=EXPL69.REPORTACF.LIST,DISP=(NEW,CATLG,DELETE),
//       UNIT=SYSDA,SPACE=(TRK,(5,9),RLSE),
//       DCB=(RECFM=F,LRECL=133,BLKSIZE=0),MGMTCLAS=DEL32
//SYSIN DD *
//OPTIONS PAGESIZE=60 LINESIZE=132 ;
%LET RETCODE=.;
%INCLUDE SOURCLIB(HIER);
LIBNAME DETAIL "SAS.LOCAL.CPE.RACF.DETAIL" DISP=SHR;
TITLE «REPORT ON THE ACTIVITY DEFINE DSN DATE AND YESTERDAY»;
FOOTNOTE «REDEVANCE LYON SERVICE PSE          «;
OPTIONS LINESIZE=133 PAGESIZE=68;
OPTIONS NOCENTER;
PROC SORT DATA=DETAIL.XTY8006 OUT=COMPR;
   BY DATETIME HOUR JOB;
DATA COMPR;
SET COMPR;
KEEP DATETIME HOUR JOB CLSNAME DSEVLNR NAEUSER RAFTERM RAFUSER OLDDSN;
PROC PRINT DATA=COMPR ;
   ID DATETIME HOUR JOB;
VAR CLSNAME DSEVLNR NAEUSER OLDDSN RAFTERM RAFUSER;
RUN;
LIBNAME DETAIL "SAS.LOCAL.CPE.RACF.DETAIL" DISP=SHR;
TITLE «REPORT ON ACCESS                    DATE      &HIER»;
FOOTNOTE «REDEVANCE LYON SERVICE PSE          «;
OPTIONS LINESIZE=133 PAGESIZE=68;
OPTIONS NOCENTER;
PROC SORT DATA=DETAIL.XTY8002 OUT=COMP;
   BY DATETIME HOUR JOB;
DATA COMP;
SET COMP;
KEEP DATETIME HOUR JOB ACEUSER ALLOW APLNAME CLSNAME DATETIME HOUR INTEN
   INT JOB OLDDSN OLDVOL RAFUSER RAFTERM RESNAME MACHINE VOLSER;
PROC PRINT DATA=COMP ;
   ID DATETIME HOUR JOB;
VAR ACEUSER ALLOW APLNAME CLSNAME INTEN RAFTERM RESNAME RAFUSER;
RUN;

Claude Dunand (France) © Xephon 1999
An ISPF dialog to manage catalog aliases – part 2

This month we complete the code for an ISPF dialog to manage catalog aliases with RACF profiles.

CLIST04

/* START OF CLIST04 */
/******************************************************************************/
/* LIB: DATASET.ISPCLIB(CLIST04) */
/* GDE: BROWSE ALIAS AND RACF INFO */
/* DOC: BROWSE THE ALIAS FROM THE CATALOG WITH PROPER RACF PROFILE */
/******************************************************************************/
PROC Ø DEBUG
IF &DEBUG = DEBUG THEN +
   CONTROL MAIN LIST CONLIST SYMLIST
ELSE +
   CONTROL MAIN NOLIST NOCONLIST NOSYMLIST NOMSG
SET SYSOUTTRAP = Ø
ISPEXEC VGET (ALSRV01,ALSMCAT1,ALSMCAT2,ALSMCAT3,ALSMCAT4,TYPE, +
   ALSJCLCK)
DO J = 1 TO 4
   SET MCAT = &&ALSMCAT&J
   ISPEXEC SETMSG MSG(MESSGØØ2)
   IF &MCAT ¬= &STR( ) THEN +
      $ LISTCAT ENT(&ALSRV01) CATALOG(&MCAT) ALL
   END
ISPEXEC SETMSG MSG(MESSGØØ7)
IF &STR(&TYPE) = &STR(U) THEN +
   $ LISTUSER &ALSRV01 TSO DFP
ELSE +
   DO
      $ LISTGRP &ALSRV01
      IF &STR(&TYPE) = &STR(A) THEN +
         DO
            ISPEXEC SETMSG MSG(MESSGØØ9)
            ISPEXEC BROWSE DATASET(&STR(&ALSJCLCK))
         END
      END
ISPEXEC SETMSG MSG(MESSGØØ5)
$ SEARCH CLASS(DATASET) MASK(&ALSRV01..)
SET ERRRC  = &LASTCC
IF &ERRRC  = 8 THEN +
   EXIT CODE(Ø)
PROFILE PREFIX(&SYSUID)
SET SYSOUTTRAP = 1ØØØ
SEARCH CLASS(DATASET) MASK(&ALSRV01..) +
CLIST('LISTDS DASET(' ') ALL DSNS GENERIC')
ALLOC DD(FILEIN) DA(EXEC.RACF.CLIST) SHR
OPENFILE FILEIN
ERROR +
   DO
      SET RC = &LASTCC
      IF (&RC = 400) THEN +
         DO
            CLOSFILE FILEIN
            FREE DD(FILEIN)
            PROFILE NOPREFIX
            SET SYSOUTLINE = Ø
            EXIT CODE(Ø)
         END
      END
   END
README: +
GETFILE FILEIN
SET CMDLEN = &LENGTH(&STR(&FILEIN))
SET RFCMD = &SUBSTR(9:CMDLEN,&FILEIN)
ISPEXEC SETMSG MSG(MESSGO06)
$ &RFCMD
GOTO README
PROFILE NOPREFIX
EXIT CODE(Ø)
/* END OF CLIST04 */

EDMAC01
/* START OF EDMAC01 */
ISREDIT MACRO
*******************************************************************/
/* LIB: DATASET.ISPCLIB(EDMAC01) */
/* GDE: INSERT A LINE AND SORT */
/* DOC: THIS EDIT MACRO IS CALLED FROM CLIST02. THIS ISPEXEC EDIT */
/* COMMAND IS THE FIRST EDIT MACRO EXECUTED BEFORE THE */
/* EDITED DATASET/MEMBER IS DISPLAYED. THIS MACRO WILL */
/* COPY A VARIABLE FROM THE SHARED POOL INTO THE FUNCTION */
/* POOL AND TRY TO FIND THE VARIABLE IN THE DATA SOMEWHERE. */
/* IF THE VARIABLE IS NOT FOUND, THIS MACRO WILL INSERT */
/* THAT VARIABLE AS THE FIRST LINE OF THE EDITED TEXT. */
/* ONCE INSERTED, THE DATA IS SORTED AND SAVED. IF FOUND, */
/* ONLY MACRO MESSAGES ARE SET, AND THE EDIT SESSION IS */
/* ENDED. THIS IS FOR THE APPLICATION HLQS. */
*******************************************************************/
ISPEXEC VGET (ALSRV01 ALSJCLCK)
ISREDIT FIND &ALSRV01 WORD
SET FINDRC = &LASTCC
IF FINDRC = Ø THEN +
   WRITE ENTRY &ALSRV01 ALREADY EXISTS IN &ALSJCLCK
ELSE +
   DO
ISREDIT LINE BEFORE .ZF = &ALSRVØ1
ISREDIT SORT
ISREDIT SAVE
WRITE ENTRY &ALSRVØ1 ADDED TO &ALSJCLCK
END
ISREDIT END
ISREDIT MACRO
/* END OF EDMACØ1 */

EDMACØ2
/* START OF EDMACØ2 */
******************************************************************************
/* LIB: DATASET.ISPCLIB(EDMACØ2) */
/* GDE: DELETE A LINE */
/* DOC: THIS EDIT MACRO IS CALLED FROM CLISTØ3. THIS ISPEXEC EDIT */
/* COMMAND IS THE FIRST EDIT MACRO EXECUTED BEFORE THE */
/* EDITED DATASET/MEMBER IS DISPLAYED. THIS MACRO WILL */
/* COPY A VARIABLE FROM THE SHARED POOL INTO THE FUNCTION */
/* POOL AND FIND THAT VARIABLE IN THE TEXT OF THE DATA. */
/* ONCE FOUND, THE LINE WILL BE DELETED, AND THE DATA WILL BE */
/* SAVED. MACRO MESSAGES ARE SET, AND THE EDIT SESSION IS */
/* ENDED. THIS IS FOR THE APPLICATION HLQS. */
******************************************************************************
ISPEXEC VGET (ALSRVØ1 ALSJCLCK)
ISREDIT FIND &ALSRVØ1 WORD
SET FINDRC = &LASTCC
IF &LASTCC = Ø THEN +
DO
    ISREDIT DELETE .ZCSR
    WRITE ENTRY &ALSRVØ1 DELETED FROM &ALSJCLCK
END
ISREDIT SAVE
ISREDIT END
/* END OF EDMACØ2 */

PANELØ1
/* START OF PANELØ1 */
******************************************************************************
/* LIB: DATASET.ISPPLIB(PANELØ1) */
/* GDE: THIS IS THE CONTROLLING PANEL. */
/* DOC: ENTER DATA ON THIS PANEL TO BE USED FOR YOUR ALIAS */
******************************************************************************
)ATTR
¬ TYPE(TEXT) INTENS(HIGH) COLOR(WHITE) SKIP(ON)
> TYPE(TEXT) INTENS(HIGH) COLOR(PINK) SKIP(ON) ATTN(ON)
' TYPE(TEXT) INTENS(LOW) COLOR(YELLOW) SKIP(ON)
~ TYPE(TEXT) INTENS(LOW) COLOR(BLUE) SKIP(ON) ATTN(ON)
+ TYPE(TEXT) INTENS(LOW) COLOR(CYAN) SKIP(ON)
ALIAS DELETE/DEFINE WITH RACF

Option ===> ZCMD

- blank'-' Browse alias  - 'A'-' Add alias  - 'D'-' Delete alias

Alias name ===> $ALSRV01

+| RACF INFORMATION

- User catalog ===> $ALSCREL

- Master catalog(s) 1. ===> $ALSMCAT1

- 2. ===> $ALSMCAT2

- 3. ===> $ALSMCAT3

- 4. ===> $ALSMCAT4

+| CATALOG INFORMATION AND JCLCHECK PARMLIB DATASET AND MEMBER

- User catalog ===> $ALSCREL

- Master catalog(s) 1. ===> $ALSMCAT1

- 2. ===> $ALSMCAT2

- 3. ===> $ALSMCAT3

- 4. ===> $ALSMCAT4

+| JCL Copy dataset(mbr) ===> $ALSJCLCK

+| Pf1'Help  > Pf3'End

)INIT

.HHELP = HLPNL01
.CURSOR = ALSRV01
.ZVARS = '(TYPE)'
&ZPRIM = YES
VGET (ALSMCAT1 ALSMCAT2 ALSMCAT3 ALSMCAT4 ALSJCLCK) PROFILE

)PROC

VER (&TYPE, LIST, A, S, U)
VER (&ALSRV01, NB)
VER (&TYPE, NB)
VER (&ALSMCAT1, NB) /* MUST HAVE AT LEAST ONE MASTER CATALOG */
IF (&ALSMCAT1, NB) /* IF ADD, MAKE SURE A USER CATALOG IS SPECIFIED */
VER (&ALSREL, NB)
IF (&TYPE = 'A') /* IF APPLICATION ENSURE A DATASET(MBR) FOR JCL */
VER (&ALSJCLCK, NB)

*******************************************************************************/
/* NOW WE HAVE THE DATA, PUT IT IN THE PROFILE POOL */
VPUT (ALSMCAT1 ALSMCAT2 ALSMCAT3 ALSMCAT4 ALSRCJCLK) PROFILE
)END
/* END OF PANEL01 */

MESSG00
/* START OF MESSG00 */
MESSG000 'PLEASE ENTER OPTION' .ALARM=_YES
'PLEASE ENTER A VALID OPTION AS LISTED BELOW '  
MESSG001 'ALREADY EXISTS - ENTER ' .ALARM=_YES
'THIS ALREADY EXISTS, PRESS ENTER OR PF3 TO CONTINUE '  
MESSG002 'CATALOG &J - PRESS ENTER' .ALARM=_YES
'ALIAS ENTRY FROM THE MASTER CATALOG '  
MESSG003 'DELETE DATASETS FIRST ' .ALARM=_YES
'YOU MUST DELETE ALL CATALOGED DATASETS BEFORE DELETING THE ALIAS '  
MESSG005 'RACF DATASET PROFILES' .ALARM=_YES
'RACF DATASET PROFILE SEARCH RESULTS FOR MASK "&ALSRV01.." '  
MESSG006 'DATASETS - PRESS ENTER' .ALARM=_YES
'DATASET DISPLAY FOR MASK "&ALSRV01.." '  
MESSG007 'OWNER - PRESS ENTER' .ALARM=_YES
'DISPLAY THE OWNER OF THE ALIAS WHETHER A USER OR A GROUP: &ALSRV01'  
MESSG008 'INVALID COMMAND REENTER' .ALARM=_YES
'PLEASE ENTER ONE OF THE FOLLOWING: <BLANK>, A, OR D (OR COMMAND)'  
MESSG009 'ENTER CMD "F &ALSRV01" ' .ALARM=_YES
'VERIFY THAT THE APPLICATION HLQ &ALSRV01 EXISTS IN THIS FILE '  
/* END OF MESSG00 */

HLPNL01
/* START OF HLPNL01 */
)ATTR DEFAULT(%+ )
/* % TYPE(TEXT) INTENS(HIGH) defaults displayed for */
/* + TYPE(TEXT) INTENS(LOW) information only */
/* _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) */
)BODY
%————————————————— ONLINE HELP FOR 'ARF' —————————————————
%SELECTION ——> ZCMD
+
%

Enter an option (A-add, D-delete, BLANK-browse). Enter the requested alias, the type (U-user, S-system, A-application), and the related catalog(s). If the type of alias is application, the JCL Check dataset must also be specified. This dataset is used by the%JCL Copy Dialog. + RACF profiles are created using IBM-RACF panels.

The following are presented in sequence, or may be selected by number:

%1+ ALIAS NAME                Options and descriptions of aliases.
%2+ ALIAS TYPE                What happens for each type of alias.
%3+ RACF INFORMATION          What gets defined to RACF and how.
%4+ CATALOG INFORMATION       User catalog and Master catalogs.
%5+ JCL COPY DATASET          Requirements for application aliases.
+
+
+ % Pf1+Help  % Pf3+End+ %ENTER+Continue

)PROC
  &ZSEL = TRANS( &ZCMD
    1,HLPNL02
    2,HLPNL03
    6,HLPNL03A
    3,HLPNL04
    4,HLPNL05
    5,HLPNL06
    *,?'*
  )
  &ZUP = HLPNL01
)END
/* END   OF HLPNL01 */

HLPNL02
/* START OF HLPNL02 */
)ATTR DEFAULT(%+_
  /* % TYPE(TEXT) INTENS(HIGH)   defaults displayed for   */
  /* + TYPE(TEXT) INTENS(LOW)    information only          */
  /* _ TYPE(INPUT) INTENS(HIGH)  CAPS(ON) JUST(LEFT)        */
)BODY
%————————————— ONLINE HELP FOR 'ARF' —————————————
%SELECTION ===>_ZCMD
+
% %
  %

  | HELP FOR ALIAS NAME |

  +
  To use a catalog, the system must be able to determine which datasets
should be defined in that catalog. A way to accomplish this is to define aliases for the catalog.

Catalog aliases are defined in the master catalog, which contains an entry for the user catalog. The number of aliases a catalog may have is limited by the maximum record size for the master catalog.

To add an alias a user must have update authority to the master catalog.

+ + + % Pf1+Help % Pf3+End %ENTER+Continue+
)PROC
   &ZUP = HLPNLØ1
)END
/* END OF HLPNLØ2 */

HLPNLØ3
/* START OF HLPNLØ3 */
)ATTR DEFAULT(%+_
   /* % TYPE(TEXT) INTENS(HIGH) defaults displayed for */
   /* + TYPE(TEXT) INTENS(LOW) information only */
   /* _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) */
)BODY
%—————————— ONLINE HELP FOR 'ARF' ———————————
%SELECTION ===>_ZCMD
+
%

| HELP FOR ALIAS TYPE |
+
+ %User+- Creates a user profile and one generic dataset profile.
+
+ %System+- Creates a group profile and one generic dataset profile.
+
+ %Application+- Creates a group profile and generic dataset profiles using BKUP, PROD, TEST, and VALT as second level qualifiers.
+
+
*****************************%IMPORTANT*****************************
+ * For user profiles, you must identify the group the user will be *
+ * added to. For system and application group profiles, you must *
+ * identify the superior group.
+ * The owner of the dataset *
+ * profile must%always+be the alias. *
Application high-level qualifiers (HLQs) are treated as special cases. All application high-level qualifiers must be maintained in a separate dataset(member). This dataset(member) is used by operations to control promotion of JCL to production.

This dataset(member) is maintained by this dialog. If an application high-level qualifier is no longer needed, it must be removed from this dataset(member) using this dialog.

Any questions or comments regarding these procedures should be submitted to the Technical Services section.
When a profile is added to RACF, ownership must be determined. The owner of the profile should be documented in the change request. The owner of a RACF profile must always be a group name (a user can also be an owner, but is not recommended). CONTACT THE SECURITY ADMINISTRATOR with any problems.

In addition to owner, the user name must also be entered. This is the name of the user. Finally, the default group must be entered. The default group will always be the same as the owner.

Group profiles are created for system and application aliases. The only additional information needed is the superior group (which will always be the same as the owner).

+ + %NOTE: + let all other fields in the RACF panels retain default values.
+ + % Pf1 + Help % Pf3 + End % ENTER + Continue +

PROC
   &ZUP = HLPNL01
END

HLPNL05

/* START OF HLPNL05 */

*/ + TYPE(TEXT) INTENS(LOW) information only */
/* + TYPE(TEXT) INTENS(HIGH) defaults displayed for */
/* _ TYPE(INPUT) INTENS(LOW) information only */
/* _ TYPE(INPUT) INTENS(HIGH) CAPS(ON) JUST(LEFT) */

*/ BODY */

%——— ONLINE HELP FOR 'ARF' ———

% SELECTION ===>_ZCMD
+
%
%

| HELP FOR RACF INFORMATION |

| HELP FOR CATALOG INFORMATION |

To list the current user catalogs enter the %TSO+ command %LISTC USERCAT+

As of 05/07/98, the user catalogs are: the Master catalogs are:

- CATALOG.ABRBASE
- CATALOG.DB2
- CATALOG.MCAT0A
- CATALOG.APPL1
- CATALOG.SYSTEM
- CATALOG.MCAT0B
- CATALOG.APPL2
- CATALOG.TSO
- CATALOG.MCAT0C
- CATALOG.CONTIG
- CATALOG.RESCUE
- CATALOG.MCAT0D

If you have any questions on catalogs or catalog aliases, please refer to the MVS/DFP Access Method Services or MVS/DFP Managing Catalogs manuals.

All production JCL must meet standards. Dataset naming standards are verified at the time the JCL is copied to the production JCL library. A list of valid high level qualifiers is maintained by this dialog.

**** This only applies to %APPLICATION+ high level qualifiers ****

The %JCL COPY+ dataset must already exist. This dialog will not create it. The existing dataset name is%TECH.PARMLIB(APPLHLQS)+

This dataset will be edited in place, and entries will either be added or deleted or browsed. When the dataset is displayed, you must press PF3 to continue.
STDERR

/* START OF STDERR */
)ATTR_DEFAULT(%+_)
# TYPE(OUTPUT) INTENS(HIGH) CAPS(ON) PAD( ' ' ) JUST(LEFT)
$ TYPE(OUTPUT) INTENS(HIGH) COLOR(YELLOW)
)BODY LMSG(LM) EXPAND(!!)
%!-! ISPF / PDF SERVICE ERRORS !-!
+COMMAND%====>_ZCMD
#LM
$ERRMSG
+
+ An ERROR has occurred during a call to an ISPF/PDF Service
+
+ Service : #ERRSERV
+ Function : #ERRFUNC
+ Return Code : #ERRRC
+
+ System message ID (ZERRMSG):
+
+ #ZERRMSG
+
+ System short message (ZERRSM):
+
+ #ZERRSM
+
+ System long message (ZERRLM):
+
+ $ZERRLM
+
)INIT
&ZCMD = ' '
)PROC
&ERRMSG = &Z
)END
/* END OF STDERR */

Steven Windhausen
Technical Specialist
County of Onondaga (USA)
Maintaining RACF authorizations

If you have to maintain a RACF environment, you will almost certainly have encountered the following two problems:

- Getting information about the profile authorization structure of your RACF database (e.g., which dataset profiles exist for a specific high-level qualifier and which users or groups have access to them?).
- The correct command syntax (sometimes you have to enclose the profile name in quotes, sometimes in brackets, etc).

To solve the first problem, I have written a REXX procedure called RACFXACC, which generates a cross-reference between resource profiles and the users or groups that have access to them. The result is saved in a sequential file that can be edited or printed.

To solve the second problem, I have integrated the procedure into an ISPF dialog. Now the cross reference is written to an ISPF table that can be modified later on.

Each modification will be translated into the appropriate RACF command that will be executed to change the RACF database. By means of this dialog you will be able to maintain most of your RACF authorizations interactively.

WHAT IS RACFXACC?

RACFXACC is a front-end procedure designed for RACF administrators (those holding the SPECIAL attribute), but it can also be used by any TSO-user to maintain his own security profiles.

RACFXACC was developed under RACF 1.9 and is now running under RACF 2.2 without problems. It is designed to determine all active classes from SETROPTS LIST, and supports most of them, but it doesn’t make sense to use the procedure for all classes. This is the reason why some classes are excluded from the class-list (i.e., USER, GROUP, GLOBAL, GMBR, TAPEVOL).
If you enter a question mark for the class parameter, you get a selection list of all supported active RACF classes (see Figure 1).

RACFXACC is based on the RACF SEARCH, LISTDSD, and RLIST commands to determine the access control information from the RACF database.

The extracted access control information is written into an ISPF table and is displayed via ISPF panels.

Our RACF group structure has three standard user groups:

- **TSO** – standard user (application programmer, etc).
- **RZ** – production user (computer-centre, job scheduling, etc).
- **SYS** – system user (system programmer, administrator, etc).

We also have a lot of function groups, so the RACFXACC main panel
is separated into universal access, the three standard groups, and all the others. From the main panel (see Figure 2) you can shift the display to the right (see Figure 3) or to the left, by entering the commands

**Figure 2: Main panel**

```plaintext
RACF Cross Reference - Row 1 to 3 of 3

Command ===> _
Scroll ===> CSR

RACF-Class : DATASET
Access-Level : (A)lter (C)ontrol (U)pdate (R)ead (E)xecute (N)one
Action-Codes : (C)opy (D)elete (E)xpand (S)how datasets

Profile - U- *** STD Access ***
name Acc SYS RZ TSO Others

_ U8Ø2259.** N U U 2
_ U8Ø2259.DB2.CNTL N U 3
_ U8Ø2259.JCL.CNTL N U 3

*********************** Bottom of data ***********************
```

**Figure 3: Display after use of SHIFT RI command**

```plaintext
RACF Cross Reference - Row 1 to 3 of 3

Command ===> _
Scroll ===> CSR

RACF-Class : DATASET
Access-Level : (A)lter (C)ontrol (U)pdate (R)ead (E)xecute (N)one
Action-Codes : (C)opy (D)elete (E)xpand (S)how datasets

Profile - Other STD Access (without SYS RZ TSO)
name Cnt. User or groups

_ U8Ø2259.** 2 U8Ø6Ø85-U DBT1ADM-R
_ U8Ø2259.DB2.CNTL 3 U8Ø6Ø85-U DBT1ADM-R LMCSM1-U
_ U8Ø2259.JCL.CNTL 3 U8Ø6Ø85-U DBT1ADM-R LMCSM1-U

*********************** Bottom of data ***********************
```
‘SHIFT RI’ or ‘SHIFT LE’. From both display types (right or left) you can expand, copy, or delete the profile information, or you can get a list of all datasets protected by this particular profile.

A note of caution! If the profile is a main profile, HLQ.**, it could protect thousands of datasets, and the response time of the request is very high.

If you expand the profile information (see Figure 4), you will be able to alter or insert access levels for users or groups.

If a group has access to the profile, you can get a list of all connected users by entering the list option ‘L’ (see Figure 5).

RACFXACC further supports the STARTED class, which replaces the started task table ICHRIN03. So with RACFXACC you can

---

```
<table>
<thead>
<tr>
<th>Command ===&gt; _</th>
<th>Scroll===&gt; CSR</th>
</tr>
</thead>
<tbody>
<tr>
<td>RACF-Class : DATASET</td>
<td></td>
</tr>
<tr>
<td>Profile : U802259.JCL.CNTL</td>
<td></td>
</tr>
</tbody>
</table>

Universal Access | SYS-User | RZ-User | TSO-User |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User - group</th>
<th>access-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>_</td>
<td>DBT1ADM</td>
</tr>
<tr>
<td>_</td>
<td>LMCSM1</td>
</tr>
<tr>
<td>_</td>
<td>U8060085</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Action-Codes : (A)lter (D)elete (I)nsert (L)ist-Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)lter (D)elete (I)nsert (L)ist-Group</td>
</tr>
</tbody>
</table>

DBT1ADM | READ | **************
LMCSM1  | UPDATE |
U8060085| UPDATE |

*************** Bottom of data *********************
```

**Figure 4: Expanded profile information**

maintain your user information for started tasks interactively (see Figure 6).

All RACLSTED classes were recognized, and when RACFXACC is left, an appropriate SETROPTS RACLST REFRESH command will be executed.

RACFXACC has the following syntax:

```
RACFXACC filter-list | ALL class ( LIST HOLD
```

where:

- ‘filter-list’ is the profile name and can be generic – for example, ‘S’ will give all profiles beginning with S, and ‘SYS1.**’ will give all profiles with HLQ SYS1.

---

*Figure 5: Display after entering list option ‘L’*
- ‘class’ is the RACF class (default: DATASET). Class = ? gives a selection list of all active RACF classes (without USER, GROUP, GLOBAL, GMBR, and TAPEVOL).

- ‘LIST’ is optional. Output is directed to dataset user-id RACFXACC.DATAF and will be displayed with EDIT.

- ‘HOLD’ is optional. The output dataset won’t be deleted after display. HOLD can only used with LIST option.

RACFXACC

/* REXX */
/* *****************************************************/
/* */
/* RACFXACC - main procedure for RACF maintenance */
/* */
/*********************************************/

trace o
arg parmlist
parse var parmlist fstrg class '( ' opt
x = outtrap(lu.)
address TSO
'LU'

x = outtrap(off)
parse var lu.3 ')'attribute
attribute = strip(attribute, 'B')
if find(attribute, 'SPECIAL') = Ø
    then do
        special = 'N'
class = ''
        if fstrg = '? ' then fstrg = userid()'.**'
        parmlist = fstrg class '(' opt
    end
else special = 'J'
drop(lu.)

/*********************************************/
/* */
/* Is RACFXACC called from a member list? */
/* */
/*********************************************/

if left(fstrg, 1) = '' & right(fstrg, 1) = ''
    then do
        memlist = 'Y'
        xx = outtrap(ld.)
        'LD DA('fstrg') GEN'
        xx = outtrap(off)
        parse var ld.1 ) DATASET sprof .
sprof = strip(sprof, 'B')
        parse var sprof hlq '.' .
fstrg = hlq'.**'
    end
else memlist = 'N'
if index(opt, 'LIST') > Ø
    then do
        call 'RACFXAC3 'parmlist
        exit
    end
if fstrg = '' then do
    fstrg = 'ALL'
class = '?'
end
if class = '' then class = 'DATASET'
if class = '?' then call moegliche_class
select
  when fstrg = '?' | class = 'ALL' then do
    call help_racfxacc
    exit
  end
  when fstrg = 'ALL' then cmd = 'SR CLASS('class') NOMASK'
  when length(fstrg) = 1 then cmd = 'SR CLASS('class') MASK('fstrg')'
  otherwise cmd = 'SR CLASS('class') FILTER('fstrg')'
end
messagep 'Search for profiles with search criterion 'fstrg' started'
x = outtrap(cmd.,999999)
address TSO
  cmd
  x = outtrap(off)
x = outtrap(xx.)
x = outtrap(li..999999)
parse var cmd.1 msgnr .
if msgnr = 'ICH31ØØ5I' then do
  lmsg = "no profiles with search criterion '"fstrg"' found"
  address ispexec "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
  exit
end
if cmd.1 = 'NO ENTRIES MEET SEARCH CRITERIA' then do
  lmsg = "no profiles with search criterion '"fstrg"' found"
  address ispexec "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
  exit
end
if cmd.Ø > 1ØØ
  then do
    frage = 'more than 1ØØ profiles meet search criterion. ',
    'Cancel processing?'
    jn = abfrage(frage)
    if jn = 'Y' then exit
  end
else messagep cmd.Ø' profiles meets search criterion 'fstrg
  do i = 1 to cmd.Ø
    parse var cmd.i prof .
    select
      when class = 'DATASET'
        then "LD DA('prof') GEN ALL"
      when class = 'STARTED'
        then "RL "class" ("prof") STDATA"
      otherwise "RL "class" ("prof") ALL"
    end
  end
x = outtrap(off)
x = outtrap(xx.)
t = '|
address 'ISPEXEC'
if class = 'STARTED'
    then do
        "TBCREATE RACFXACC",
        "KEYS(PROF) NAMES(STUSR STGRP STTRST STPRIV STTRACE) WRITE"
        "TBSORT RACFXACC FIELDS(PROF)"
    end
else do
    "TBCREATE RACFXACC",
    "KEYS(PROF) NAMES(UACC ASYS ARZ ATSO CNTW WEITER FKZ FKZW) WRITE"
    "TBSORT RACFXACC FIELDS(PROF)"
end

address 'TSO'
pr_ok = Ø
select
  when class = 'DATASET' then call dataset
  when class = 'STARTED' then call started
  otherwise                   call genresource
end

call ausgabe

drop li. /* free variables no longer used */
drop cmd.

address ispexec
  if class = 'STARTED' then panel = 'RACFXACA'
  else panel = 'RACFXACØ'

ztdtop = 1
  do while ret < 8
      "TBTOP   RACFXACC"
      "TBSKIP  RACFXACC NUMBER("ztdtop")"
  end
  if memlist = 'Y'
    then do
        prof = sprof
        "TBSCAN RACFXACC ARGLIST(PROF) CONDLIST(EQ)"
        memlist = 'N'
    end
  "TBDISPL RACFXACC PANEL("panel")"
ret = rc
select
  when zcmd = 'SHIFT RI'
    then do
        if class = 'STARTED' then panel = 'RACFXACA'
        else panel = 'RACFXAC1'
    end
  when zcmd = 'SHIFT LE'
    then do
        if class = 'STARTED' then panel = 'RACFXACA'
        else panel = 'RACFXACØ'
    end
  when zcmd = ''
    then nop
  when left(zcmd,8) = 'RACFXACC'
then do
  queue 'RACFXACC' zcmd
  ret = 8
  iterate end
when left(zcmd,1) = '#'
then do
  queue 'RACFXACC' substr(zcmd,2)
  ret = 8
  iterate end
otherwise do
  nop
/*———————————————————————————————————————————————————————— */
/* Optional. If you want to process a new racfxacc command */
/* without the command character '#' then uncomment the */
/* following 3 lines and delete the nop statement */
/*———————————————————————————————————————————————————————— */
/* queue 'RACFXACC' zcmd */
/* ret = 8 */
/* iterate */
/*———————————————————————————————————————————————————————— */
end
do while ztdsels > Ø
select
  when sel = 'D'
    then do
      call delete_profile
    end
  when sel = 'E'
    then do
      if class = 'STARTED'
        then call expand_profstdata
      else call expand_profile
    end
  when sel = 'A'
    then do
      if class = 'STARTED'
        then call expand_profstdata
    end
  when sel = 'C'
    then do
      call copy_profile
    end
  when sel = 'S'
    then do
      call show_profile
    end

otherwise nop
end
if ztdsels > 1 then "TBDISPL RACFXACC"
  else ztdsels = Ø
end
"TBEND RACFXACC"
address TSO
call pruefen_raclist
exit
AUSGABE:
if cntw > 4 then fkzw = '>
  else fkzw = ' '
address 'ISPEXEC' "TBADD RACFXACC ORDER"
pr_ok = Ø
return

DATASET:
do i = 1 to li.Ø
  select
    when pos('INFORMATION FOR DATASET',li.i) > Ø
      then do
        if pr_ok = 1 then call ausgabe
        parse var li.i . . prof .
        prof = strip(prof,'B')
        if length(prof) > 25 then fkz = '>
          else fkz = ' '
        pr_ok = 1
      end
    when pos('UNIVERSAL ACCESS',li.i) > Ø
      then do
        i = i + 2
        parse var li.i . . uacc .
        uacc = left(uacc,1)
      end
    when pos('ID     ACCESS',li.i) > Ø
      then do
        weiter = ''
        asys = ''
        atso = ''
        arz = ''
        cntw = Ø
        i = i + 2
        parse var li.i grp sacc .
        select
          when grp = 'SYS' then do
            asys = left(sacc,1)
          end
          when grp = 'RZ' then do
            arz = left(sacc,1)
          end
      end
  end
end
when grp = 'TSO' then do
    atso = left(sacc,1)
end
when grp = 'NO' & sacc = 'ENTRIES' then nop
otherwise do
    cntw = cntw + 1
    weiter = weiter||left(grp,9,'-')||left(sacc,1)' ' 
end
i = i + 1
do while li.i ¬= ' '
pdase var li.i grp sacc .
select
    when grp = 'SYS' then do
        asys = left(sacc,1)
    end
    when grp = 'RZ' then do
        arz = left(sacc,1)
    end
    when grp = 'TSO' then do
        atso = left(sacc,1)
    end
    otherwise do
        cntw = cntw + 1
        weiter = weiter||left(grp,9,'-')||left(sacc,1)' ' 
    end
end
i = i + 1
end
otherwise nop
end
end
return

GENRESOURCE:
do i = 1 to li.Ø
select
    when pos(class' ',',li.i) = 1
    then do
        if pr_ok = 1 then call ausgabe
        parse var li.i . prof .
        prof = strip(prof,'B')
        if length(prof) > 25 then fkz = '>'
        else fkz = ''
        pr_ok = 1
    end
when pos('UNIVERSAL ACCESS',li.i) > Ø
then do
    i = i + 2
    parse var li.i . . uacc .
    uacc = left(uacc,1)
end
when pos('USER ACCESS',li.i) > Ø then do
    weiter = ''
    asys = ''
    atso = ''
    arz = ''
    cntw = Ø
    i = i + 2
    parse var li.i grp sacc .
    select
        when grp = 'SYS' then do
            asys = left(sacc,1)
        end
        when grp = 'RZ' then do
            arz = left(sacc,1)
        end
        when grp = 'TSO' then do
            atso = left(sacc,1)
        end
        when grp = 'NO' & sacc = 'ENTRIES' then nop
        otherwise do
            cntw = cntw + 1
            weiter = weiter||left(grp,9,'-')||left(sacc,1)' '
        end
    end
    i = i + 1
do while li.i ≠ ''
    parse var li.i grp sacc .
    select
        when grp = 'SYS' then do
            asys = left(sacc,1)
        end
        when grp = 'RZ' then do
            arz = left(sacc,1)
        end
        when grp = 'TSO' then do
            atso = left(sacc,1)
        end
        otherwise do
            cntw = cntw + 1
            weiter = weiter||left(grp,9,'-')||left(sacc,1)' '
        end
    end
    i = i + 1
STARTED:
doi = 1 to li.Ø
select
when pos(class' ',li.i) = 1
then do
  if pr_ok = 1 then call ausgabe
  parse var li.i . prof .
  prof = strip(prof,'B')
  pr_ok = 1
end
when pos('STDATA INFORMATION',li.i) = 1
then do
  i = i + 2
  parse var li.i . stusr .
  i = i + 1
  parse var li.i . stgrp .
  i = i + 1
  parse var li.i . sttrst .
  i = i + 1
  parse var li.i . stpriv .
  i = i + 1
  parse var li.i . sttrace .
end
when pos('NO STDATA INFORMATION',li.i) = 1
then do
  stusr   = ''
  stgrp   = ''
  sttrst  = ''
  stpriv  = ''
  sttrace = ''
end
otherwise nop
end
return

COPY_PROFILE:
"CONTROL DISPLAY SAVE"
aprof = prof
"ADDPOP"
nok = 1
do while nok = 1
"VGET (ZFKA) PROFILE"
if zfka ≠ 'OFF'
    then do
        cmdstack = 'FKA OFF'
        'DISPLAY COMMAND(CMDSTACK)'
        cfka = '1'
    end
else cfka = Ø
"DISPLAY PANEL(RACFXAC3)"
retp = rc
if cfka = 1
    then do
        cmdstack = 'FKA ON'
        'DISPLAY COMMAND(CMDSTACK)'
        cfka = 'Ø'
    end
if retp = Ø
    then do
        prof = nprof
        if aprof = nprof
            then do
                smsg = 'copy failure'
                lmsg = 'source and target must not be the same'
                "SETMSG MSG(RACF000A)" /*MSGLOC(CLASS)*/
                iterate
            end
        if class = 'DATASET'
            then do
                parse var aprof ahlq '.' .
                parse var nprof nhlq '.' .
                kz_nprof = Ø
                if ahlq ≠ nhlq then call prf_newprof
                if kz_exist = 1
                    then do
                        call replace_verification
                        if repok = 'J'
                            then do
                                cmd = "DD ('"nprof"') GEN"
                                address TSO cmd
                            end
                        else do
                            iterate
                        end
                    end
                if kz_nprof = Ø
                    then do
                        "TBADD RACFXACC ORDER"
                        if rc = Ø
                            then do
                                cmd = "AD ('"nprof"') GEN FROM('"aprof"') FGEN"
                                address TSO cmd
                            end
if rc ≠ Ø
then do
  smsg = 'copy failure RC='rc
  lmsg = 'Cmd "'cmd"" ended with returncode 'rc
  "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
  "TBDELETE RACFXACC"
  iterate
end
else do
  call replace_verification
  if repok = 'J'
  then do
    cmd = "DD ('"nprof"') GEN"
    address TSO cmd
    cmd = "AD ('"nprof"') GEN FROM('"aprof"') FGEN"
    address TSO cmd
    "TBDELETE RACFXACC"
    "TBADD RACFXACC ORDER"
  end
  else iterate
end
else do
  if kz_nprof = 1
  then do
    parse var nprof hlq '.' prfrest
    if prfrest = '**' /* nprof is mainprofile */
    then do
      frage = 'no group defined for profile 'nprof,
             '- should group 'nhlq' be defined ?'
      antw = abfrage(.frage)
      if antw = 'Y'
      then do
        cmd = 'AG ('nhlq') OWNER (DATASET) SUPGROUP(DATASET)'
        address TSO cmd
        cmd = "AD ('"nprof"') GEN FROM('"aprof"') FGEN"
        address TSO cmd
        "TBADD RACFXACC ORDER"
      end
      else do
        smsg = 'copy failure'
        lmsg = 'no group-/userprofile 'nhlq' defined'
        "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
        iterate
      end
  end
  else do           /* nprof is no mainprofile */
    smsg = 'copy failure'
  end
lmsg = nprof 'is no mainprofile.',
   'group 'nhlq' could not be defined automatically'
"SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
iterate
   end
end
if kz_nprof = 2
   then do
      smsg = 'copy failure'
lmsg = 'copy to other HLQ not supported'
"SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
iterate
   end
end
else do      /* no Dataset Profile */
   "TBADD RACFXACC ORDER"
   if rc = Ø
      then do
         cmd = "RDEF "class" ("nprof") FROM("aprof")"
         address TSO cmd
         if rc ¬= Ø
            then do
               smsg = 'copy failure RC='rc
               lmsg = 'Cmd "cmd" ended with returncode 'rc
               "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
               "TBDELETE RACFXACC"
               iterate
            end
      end
   if class = 'STARTED' then call copy_stdata
   else do
      smsg = 'profile already defined'
lmsg = 'profile 'prof' for class',
         class 'already defined'
"SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
iterate
   end
nok = Ø
end
"REMPop"
"CONTROL DISPLAY RESTORE"
return

SHOW_PROFILE:
   "CONTROL DISPLAY SAVE"
   "VPUT (prof)"
   "SELECT CMD(RACFXAC2)"
"CONTROL DISPLAY RESTORE"
return

DELETE_PROFILE:
if class = 'STARTED' & prof = '**'
then do
  smsg = 'delete impossible'
  lmsg = '"**" profile shouldn't be deleted, because",
       'it is the dummy profile for all started tasks'.
       'not named'
  "SETMSG MSG(RACFØØØA)"
  return
end

"CONTROL DISPLAY SAVE"
"ADDPOP"
"VGET (ZFKA) PROFILE"
if zfka ≠ 'OFF'
then do
  cmdstack = 'FKA OFF'
  'DISPLAY COMMAND(CMDSTACK)'
  cfka = '1'
end
else cfka = Ø

"DISPLAY PANEL(RACFXAC4)"
retp = rc
if cfka = 1
then do
  cmdstack = 'FKA ON'
  'DISPLAY COMMAND(CMDSTACK)'
  cfka = 'Ø'
end
if delok = 'J'
then do
  if class = 'DATASET'
  then do
    "TBQUERY RACFXACC ROWNUM(ANZROW)"
    parse var prof hlq '.' prfrest
    if prfrest = '**'
    then do
      if anzrow = 1
      then do
        cmd = "DD ('"prof"') GEN"
        address TSO cmd
        "TBDELETE RACFXACC"
        cmd = 'DG' hlq
        address TSO cmd
        smsg = 'group 'hlq' deleted'
        lmsg = 'profile 'prof' and group 'hlq, 'deleted'
        "SETMSG MSG(RACFØØØA)"
        return
      end
    end
  end
end

end
else do
  smsg = 'delete failure'
lmsg = 'delete only possible if mainprofile',
    'is the last one'
"SETMSG MSG(RACFØØA)"
end
else do
  cmd = "DD ('"prof"') GEN"
  address TSO cmd
  "TBDELETE RACFXACC"
end
else do
  cmd = "RDEL "class" ("prof")"
  address TSO cmd
  "TBDELETE RACFXACC"
end
end
"REMPop"
"CONTROL DISPLAY RESTORE"
return

EXPAND_PROFILE:
  "CONTROL DISPLAY SAVE"
updkz = Ø
nuacc = uacc
nasys = asys
narz = arz
natso = atso
"VPUT (nuacc,nasys,narz,natso,weiter,cntw,class,prof)"
"SELECT CMD(RACFXAC1)"
"VGET (nuacc,nasys,narz,natso,weiter,cntw)"
if left(nuacc,1) ¬= uacc then do
  if class = 'DATASET' then cmd = "ALD ('"prof"') GENERIC UACC("nuacc")"
    else cmd = "RALT "class" ("prof") UACC("nuacc")"
  address TSO cmd
  uacc = left(nuacc,1)
if class = 'DATASET' then dlm = "'"
  else dlm = ""
if left(nasys,1) ¬= asys then do
  if nasys = ' ' then cmd = "PE "dlm||prof||dlm" GENERIC CLASS("class")",.
    "ID(SYS) DELETE"
    else cmd = "PE "dlm||prof||dlm" GENERIC CLASS("class")",.
"ID(SYS) ACC("nasys")"

address TSO cmd

end

asys = left(nasys,1)
if left(narz,1) = arz then do
  if narz = ' ',
    then cmd = "PE "d1m||prof||d1m" GENERIC CLASS("class"),
        "ID(RZ) DELETE"
  else cmd = "PE "d1m||prof||d1m" GENERIC CLASS("class"),
        "ID(RZ) ACC("nasys")"

address TSO cmd

end

arz = left(narz,1)
if left(natso,1) = atso then do
  if natso = ' ',
    then cmd = "PE "d1m||prof||d1m" GENERIC CLASS("class"),
        "ID(TSO) DELETE"
  else cmd = "PE "d1m||prof||d1m" GENERIC CLASS("class"),
        "ID(TSO) ACC("natso")"

address TSO cmd

end

atso = left(natso,1)
"CONTROL DISPLAY RESTORE"
"TBPUT RACFXACC"
return

EXPAND_PROFSTDATA:
"CONTROL DISPLAY SAVE"
if sel = 'A' then do
  nprof    = ''
  nstusr   = ''
  nstgrp   = ''
  nsttrst  = 'NO'
  nstpriv  = 'NO'
  nsttrace = 'NO'
  prof     = ''
  stusr    = ''
  stgrp    = ''
  sttrst   = 'NO'
  stpriv   = 'NO'
  sttrace  = 'NO'
end
else do
  nprof    = prof
  nstusr   = stusr
  nstgrp   = stgrp
  nsttrst  = sttrst

nstpriv = stpriv
nsttrace = sttrace
end

"ADDPOP"
nok = 1
do while nok = 1
"VGET (ZFKA) PROFILE"
if zfka ≠ 'OFF'
  then do
    cmdstack = 'FKA OFF'
    'DISPLAY COMMAND(CMDSTACK)'
    cfka = '1'
    end
  else cfka = Ø
"DISPLAY PANEL(RACFXACB)"
retp = rc
if cfka = 1
  then do
    cmdstack = 'FKA ON'
    'DISPLAY COMMAND(CMDSTACK)'
    cfka = 'Ø'
    end
if retp = Ø
  then do
    chng_kz = Ø
    if nprof ≠ prof
      then do
        prof = nprof
        chng_kz = 1
        end
    if nstusr ≠ stusr
      then do
        stusr = nstusr
        chng_kz = 1
        end
    if nstgrp ≠ stgrp
      then do
        stgrp = nstgrp
        chng_kz = 1
        end
    if nsttrst ≠ sttrst
      then do
        sttrst = nsttrst
        chng_kz = 1
        end
    if nstpriv ≠ stpriv
      then do
        stpriv = nstpriv
        chng_kz = 1
        end
if nsttrace != sttrace
    then do
        sttrace = nsttrace
        chng_kz = 1
    end
if chng_kz = 1
    then do
        if nstgrp = '' then stgruppe = 'NOGROUP'
            else stgruppe = 'GROUP('nstgrp')'
        if sel = 'A'
            then do
                if nstgrp = '' then stgruppe = ''
                cmd = "RDEF "class" ("prof") STDATA( USER("nstusr")", stgruppe" PRIVILEGED("nstpriv") TRUSTED("nsttrst")", "TRACE("nsttrace")")
                "TBADD RACFXACC ORDER"
                if rc = Ø
                    then do
                        address TSO cmd
                        if rc ≠ Ø
                            then do
                                smsg = 'add failure RC='rc
                                lmsg = 'Cmd ""cmd"", '
                                'ended with returncode 'rc
                                "SETMSG MSG(RACFØØØA)"
                                "TBDELETE RACFXACC"
                                iterate
                            end
                        else do
                            smsg = 'profile already defined'
                            lmsg = 'profile 'prof' for class',
                            class 'already defined'
                            "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
                            iterate
                        end
                    end
                else do
                    cmd = "RALT "class" ("prof") STDATA( USER("nstusr")", stgruppe" PRIVILEGED("nstpriv") TRUSTED("nsttrst")", "TRACE("nsttrace")")
                    address TSO cmd
                    "TBPUT RACFXACC"
                end
            end
        else do
            smsg = 'profile already defined'
            lmsg = 'profile 'prof' for class',
            class 'already defined'
            "SETMSG MSG(RACFØØØA)" /*MSGLOC(CLASS)"*/
            iterate
        end
    end
nok = Ø
end
"CONTROL DISPLAY RESTORE"
return
PRUEFEN_RACLIST:
    if special = 'N' then return
    xx = outtrap(li.,999999)
    'SETROPTS LIST'
    xx = outtrap(off)
    do i = 1 to li.Ø
        if find(li.i,'RACLASS CLASSES =') > Ø
            then do
                parse var li.i . '=' raclist
                raclist = strip(raclist,'B')
                leave
            end
        i = i + 1
    end
    do while left(li.i,16) = ' '
        raclist = raclist' 'strip(li.i,'B')
        i = i + 1
    end
    drop li.
    if find(raclist,class) > Ø
        then 'SETROPTS RACLST('class') REFRESH'
    return

MOEGLICHE_CLASS:
    xx = outtrap(li.,999999)
    'SETROPTS LIST'
    xx = outtrap(off)
    call laden_classtext
    address ispexec
    'TBCREATE RACFXAC4 KEYS(ACLASS) NAMES(CTYP CTXT)'
    'TBSORT RACFXAC4 FIELDS(CTYP,C,A,ACLASS,C,A)'
    do i = 1 to li.Ø
        if left(li.i,16) = 'ACTIVE CLASSES ='
            then do
                parse var li.i . '=' actclass
                actclass = strip(actclass,'B')
                do while length(actclass) > Ø
                    parse var actclass aclass actclass
                    prf = 'USER GROUP GLOBAL GMBR TAPEVOL'
                    if find(prf,aclass) = Ø
                        then do
                            ix = value(aclass)
                            ctyp = ctyp.ix
                            ctxt = ctxt.ix
                            'TBADD RACFXAC4 ORDER'
                        end
                end
                leave
            end
        i = i + 1
    end
i = i + 1
do while left(li.i, 17) = ' '  
    actclass = strip(li.i,'B')
    do while length(actclass) > 0
        parse var actclass aclass actclass
        prf = 'USER GROUP GLOBAL GMBR TAPEVOL'
        if find(prf,aclass) = 0
            then do
                ix = value(aclass)
                ctyp = ctyp.ix
                ctxt = ctxt.ix
                'TBADD RACFXAC4 ORDER'
            end
        end
    end
    i = i + 1
end
'TBTOP RACFXAC4'
'TBDISPL RACFXAC4 PANEL(RACFXAC8)'
if rc = 8 then fstrg = '?'
else class = aclass
'TBEND RACFXAC4'
address TSO
return

Editor’s note: this article will be concluded in the next issue.

Roman Hawlitschek  
Systems Programmer (Germany)  © Xephon 1999

Call for papers

Why not share your expertise and earn money at the same time? RACF Update is looking for REXX EXECs, macros, program code, etc, that experienced RACF users have written to make their life, or the lives of their users, easier. We will publish it (after vetting by our expert panel) and send you a cheque when the article is published. Articles can be of any length and can be sent or e-mailed to Trevor Eddolls at any of the addresses shown on page 2. Why not call now for a free copy of our Notes for contributors?
Identifying orphan entries in RACF access lists

Every RACF administrator has to deal with the problem of orphan PERMITs accumulating. In the past there was no tool to delete an ID automatically from every access list; indeed, there is still no obvious IBM tool to identify orphan PERMITs. RACF administration software packages can do this, but they are rather expensive, especially for smaller shops with only a few thousand, or fewer, users and groups.

Almost unnoticed by RACF administrators, ICETOOL, a powerful universal tool to aid RACF administration, has been introduced as part of IBM’s DFSORT.

The advantages are obvious:

- It does not require a database system.
- There is no query language to learn.
- There are no additional fees to pay for additional software packages.
- Once installed, it can be used for a long period of time with no maintenance and no upgrades.

The following job stream is a sample of the usability of ICETOOL. It may look a little bit complicated, but it works.

The idea is that any ID in an access list must be either a known user-id or a known group-id. Any other ID must be an orphan.

PREREQUISITES
The prerequisites are as follows:

- DFSORT Release 13 or higher.
- Any RACF release that supports the IRRDBU00 utility.

Warning: this job stream may not be suitable for installations with more than ten thousand users and groups.

You should expect a return code of 4.
DESCRIPTION

Step one extracts all IDs from all access lists using a flat file created by IRRDBU00 unload utility (refer to the related IBM documents). Because of the different offsets in dataset profiles and general resource profiles, there has to be a different SORT (marked S01 and S03) for each type of profile. SORTs S02 and S04 convert these extracted records to ‘LU xxxxxxx TSO NORACF’ commands. SORT S05 eliminates duplicates. Copies to LIST1, LIST2, and LIST3 are for documentation only.

At this point we know all IDs that are contained in access lists.

Step two executes the ‘LU xxxxxxx TSO NORACF’ commands. The only information of interest is the ICH300011 message, and so ‘LU xxxxxxx TSO NORACF’ has been chosen because it produces very little output.

At this point we know all IDs that are contained in access lists and are not user-ids.

Step three copies the IKJEFT01 output to LIST4 for documentation (S06) and generates new ‘LG xxxxxxx DFP NORACF’ commands for all IDs that are not user-ids. This command produces very little output.

Step four executes the ‘LG xxxxxxx DFP NORACF’ commands.

At this point we know all IDs that are contained in access lists and are neither group-ids nor user-ids – therefore, they must be orphans.

Step five eliminates all lines from the IKJEFT01 output that do not contain ‘LG’ or ‘ICH510031’. Since ICH510031, unfortunately, does not repeat the group-id which has not been found, there is no direct path to generate input for utility IRRUT100. Copies to LIST6 and LIST7 are for documentation only.

Step six executes a REXX IRA001, which examines the output of step five and creates input for the IRRUT100 utility. To avoid the failure of step seven, at least one dummy entry will be created.

Step seven executes the IRRUT100 utility, which tells us in which access list an orphan-id is contained. Depending on the number of orphan-ids, this step may take a considerable amount of time to run.
and you may have to increase the space request for SYSUT1. Use the output of IRRUT100 to eliminate the unwanted IDs.

SOURCE CODE

//job JOB (account)....... 
/**
/**
/** This job finds orphan entries in permit lists.
/**
/** Step1 find all permits and generate LISTUSER commands
/**
/** Step2 execute LISTUSER commands
/**
/** Step3 find all IDs that are not actual user-ids and
/** generate LISTGRP commands
/**
/** Step4 execute LISTGRP commands
/**
/** Step5 extract all IDs that are not actual group-ids
/**
/** Step6 generate IRRUT100 input
/**
/** Step7 run IRRUT100 utility
/**
/**
/**STEP1 EXEC PGM=ICETOOL
/**
//SYSOUT DD SYSOUT=* 
/**
//SORTIN DD DISP=SHR,DSN="IRRDBU00.unloaded.RACF dataset"
/**
//SORTOUT DD SYSOUT=* 
/**
//SORTXX1 DD DISP=(NEW,PASS),DSN=&amp;&amp;SOUTXX1,UNIT=SYSDA,
// SPACE=(TRK,(15,15),RLSE)
//SORTXX2 DD DISP=(NEW,PASS),DSN=&amp;&amp;SOUTXX2,UNIT=SYSDA,
// SPACE=(TRK,(15,15),RLSE)
//SORTXX3 DD DISP=(MOD,PASS),DSN=&amp;&amp;SOUTXX3,UNIT=SYSDA,
// SPACE=(TRK,(15,15),RLSE)
//SORTXX5 DD DISP=(NEW,PASS),DSN=&amp;&amp;SOUTXX5,UNIT=SYSDA,
// SPACE=(TRK,(15,15),RLSE).DCB=(LRECL=0,BLKSIZE=0),
// RECFM=FB
/**
//LIST1 DD SYSOUT=* 
//LIST2 DD SYSOUT=* 
//LIST3 DD SYSOUT=* 
/**
//TOOLMSG DD SYSOUT=* 
//DFSMSG DD SYSOUT=*
//TOOLIN DD *
*
* SØ1 : find all permits for dataset profiles
*   SORT FROM(SORTIN) TO(SORTXX1) USING(DSN1)
* SØ2 : list entries
*   SORT FROM(SORTXX1) USING(DSN3)
* SØ3 : find all permits for general resource profiles
*   SORT FROM(SORTIN) TO(SORTXX2) USING(GEN1)
* SØ4 : list entries
*   SORT FROM(SORTXX2) USING(GEN4)
* SØ5 : delete duplicates
*   SORT FROM(SORTXX3) USING(SUM1)
*
/**
***
//DSN1CNTL DD *
*
* extract permits for dataset profiles
*
*   SORT FIELDS=(62,Ø8,CH,A)      | SORT BY ID
*     INCLUDE COND=(5,4,CH,EQ,C'Ø4Ø4')  | DATASET PROFILES ONLY
/**
***
//DSN3CNTL DD *
*
* generate list of entries
*
*   SORT FIELDS=(62,8,CH,A)       | SORT BY ID
*   SUM FIELDS=NONE
*   OUTFIL FNAMES=(LIST1,SORTXX3),LINES=99.
*     HEADER2=(5:'Permits for dataset profiles'.
*                6Ø:'Page: ',PAGE,
*               1/,6Ø:'Datem: ',DATE=(DMY.).
*               1/),
*     OUTREC=(5:C'LU'.
*                15:62,8,
*                3Ø:C'TSO NORACF'.
*                4Ø:41X),CONVERT
/**
***
//GEN1CNTL DD *
*
* extract permits for general resource profiles
SORT FIELDS=(266,8,CH,A) INCLUDE COND=(5,4,CH,EQ,C'Ø5Ø5') | GEN. RES.PROF. ONLY
/*
//GEN4CNTL DD *
*
* generate list of entries
*
SORT FIELDS=(266,8,CH,A) | SORT BY ID
SUM FIELDS=NULL
*
OUTFIL FNAMES=(LIST2,SORTXX3),LINES=99,
  HEADER2=(5:'permits for general resource profiles',
  60:'Page: 'PAGE,
  1/,60:'Date: ',DATE=(DMY.),
  1/),
  OUTREC=(5:C'LU',
  15:266,8,
  30:C'TSO NORACF',
  40:41X),CONVERT
/*
//SUM1CNTL DD *
*
* delete duplicates
*
SORT FIELDS=(16,8,CH,A) SUM FIELDS=NONE
INCLUDE COND=((6,2,CH,EQ,C'LU'),AND,(16,1,CH,NE,C'*'))
*
OUTFIL FNAMES=(LIST3,SORTXX5)
/*
/**
/**
/// Step2 execute LISTUSER commands
/**
/**
///STEP2 EXEC PGM=IKJEFT01
//SYSPRINT DD SYSOUT=* //SYSTSPT RT DD DISP=(NEW,PASS),DSN=&&PRINT01,UNIT=SYSDA,
 // SPACE=(TRK,(15,15),RLSE),DCB=(LRECL=80,BLKSIZE=0),
 // RECFM=FB
//SYSTSIN DD DISP=(OLD,PASS),DSN=&&SOUTXX5
/**
/**
///STEP3 EXEC PGM=ICETOOL
/**
//SYSDUT DD SYSOUT=* /**
/**
///SORTIN DD DISP=(OLD,PASS),DSN=&&PRINT01 /*
SORT XX6   DD DISP=(NEW,PASS),DSN=&SOUTXX6,UNIT=SYSDA.
//SPACE=(TRK,(15,15),RLSE),DCB=(LRECL=0,8,BLKSIZE=0).
//RECFM=FB
//*
LIST 4   DD SYSOUT=* 
LIST 5   DD SYSOUT=* 
//*
TOOLMSG DD SYSOUT=* 
DFSMSG  DD SYSOUT=* 
//TOOLIN DD * 
* SØ6 : copy IKJEFTØ1 output 
* 
SORT FROM(SORTIN) TO(LIST4) USING(SRT1) 
* 
SØ7 : generate IKJEFTØ1 input 
* 
SORT FROM(SORTIN) USING(SRT2) 
* 
/* 
//SRT1CNTL DD * 
* 
SØ6 : copy output 
* 
SORT FIELDS=COPY 
/* 
SRT2CNTL DD * 
* 
SØ7 : convert all ICH3ØØØ1I messages to LG commands 
* 
SORT FIELDS=(428.CH,A) INCLUDE COND=((1,22,SS,EQ,C'ICH3ØØØ1I')) SUM FIELDS=NONE OUTFIL FNAMES=(LIST5,SORTXX6),OUTREC=(5:C'LG',15:428,3Ø:C'DFP NORACF',4Ø:41X),CONVERT /* 
/* 
//** execute LISTGRP commands 
//** 
//STEP4 EXEC PGM=IKJEFTØ1 
//SYSPRINT DD SYSOUT=* 
//SYSTSPRT DD DISP=(NEW,PASS),DSN=&PRINTØ2,UNIT=SYSDA, 
//SPACE=(TRK,(15,15),RLSE),DCB=(LRECL=0,8,BLKSIZE=0), 
//RECFM=FB 
//SYSTSIN DD DISP=(OLD,PASS),DSN=&SOUTXX6 
//*
//STEP5 EXEC PGM=ICETOOL
//SYSOUT DD SYSOUT=* 
//SORTIN DD DISP=(OLD, PASS), DSN=& &PRINTØ2
//SORTXX7 DD DISP=(NEW, PASS), DSN=& &SORTXX7, UNIT=SYSDA,
// SPACE=(TRK,(15,15), RLSE), DCB=(LRECL=80, BLKSIZE=0),
// REC FM=FB
//LIST6 DD SYSOUT=* 
//LIST7 DD SYSOUT=* 
//TOOLMSG DD SYSOUT=* 
//DFSMSG DD SYSOUT=* 
//TOOLIN DD *

* SØ8 : copy IKJEFTØ1 output to list dataset
* 
SORT FROM(SORTIN) TO(LIST6) USING(SRT1)
*
* SØ9 : copy IKJEFTØ1 output for further processing
* 
SORT FROM(SORTIN) USING(SRT2)
*
/

//SRT1CNTL DD *

* copy IKJEFTØ1 output to list dataset
*
SORT FIELDS=COPY
/
//SRT2CNTL DD *

* copy IKJEFTØ1 output for further processing
*
SORT FIELDS=COPY
INCLUDE COND=((1,22, SS, EQ, C'ICH51ØØ3I'), OR,(5,2, CH, EQ, C'LG'))
OUTFIL FNAMES=(LIST7,SORTXX7)
/
/
/// generate IRRUTIØØ input
///
///
//STEP6 EXEC PGM=IKJEFTØ1
//SYSPROC DD DISP=SHR, DSN=your.ISPF.EXEC
//SYSPRINT DD SYSOUT=*
IRA001

/* REXX IRA001 */
/* This REXX converts output of step 5 to IRRUT100 input */
/* */
/* */
ARG DEBUG
IF DEBUG = "DEBUG" THEN TRACE I
ADDRESS TSO
"EXECIO * DISKR IRAIN (STEM GROUPLIST.)"
I = Ø
J = 1
OUTLIST.J = "     " /* DUMMY ENTRY */
IMAX = GROUPLIST.Ø
DO WHILE (I < IMAX)
   I = I + 1
   IF FIND(GROUPLIST.I,"ICH51003I") > Ø THEN DO
      J = J + 1
      I1 = I - 1
      OUTLIST.J = SUBSTR(GROUPLIST.I1,10,15)
      OUTLIST.Ø = J
   END
END
"EXECIO "OUTLIST.Ø" DISKW IRAOUT (STEM OUTLIST. FINIS)"
EXIT Ø

Editor’s note: any comments on this article would be welcomed and should be addressed either to Xephon or directly to the author at k.r.blatt@t-online.de.

Karl Reinhard Blatt
System Programmer (Germany) © Xephon 1999
Sites running RACF as well as NDS will be interested in Blockade’s new Password Synchronization for Novell Directory Server, enabling them to extend the reach and flexibility of their existing security investments with a product that requires virtually no changes to their current system architecture.

Password synchronization enables users’ IDs and passwords to be synchronized – or harmonized – dynamically and in real-time across distributed computing networks using heterogeneous computing environments, platforms, and operating systems, thereby dramatically tightening system security and reducing the number of passwords users must remember to just one per person. Additionally, Blockade’s Password Synchronization propagates user-related administration activity (password creation, revocation, etc) from a single point, thereby eliminating the need for an administrator to enter new or modified user information repeatedly onto each platform or operating system.

Blockade’s Password Synchronization Services software are compatible with Novell, NT, HP-UX, Sun Solaris, AIX, MVS/OS/390 (through RACF, CA-ACF2, and CA-TopSecret), a variety of distributed databases, and AS/400.

For further details contact:
Blockade Systems, 2200 Younge Street, Number 1400, Toronto, ON, M4S 2C6, Canada.
Tel: (416) 482 8400.

IBM has announced Version 2 Release 8 of OS/390. Central to the upgrade are security and systems management features.

Of interest to RACF users is the fact that the LDAP server has been enhanced to support LDAP Version 3 protocol, enabling OS/390 LDAP Server to interoperate with other LDAP Version 3 clients and servers. LDAP on OS/390 includes Java support, LDAP access to RACF information, and LDAP client authentication using RACF. It also supports SSL for encrypted privacy of communication and it supports multiple LDAP servers on multiple systems in a parallel sysplex.

Other security-related features include IPSec VPN, providing a secure pathway between OS/390 and other IPSec VPN-capable systems, routers, and firewalls through encryption using the System/390 hardware CMOS Cryptographic Coprocessor. A new feature is the exchange of encryption keys between the end-points of IPSec VPN, which can be automated and dynamically managed through Internet Key Exchange (IKE).

Also new is centralized management of digital certificates belonging to server applications and their related private encryption keys, allowing users and application developers to provide common secure management of these certificates as well as the chain of trust needed to verify user certificates presented to these applications.

For further information contact your local IBM representative.

xephon