31

RACF update

February 2003

In this issue

3 Checking dataset access

11 RACF in focus – the Search command

14 Pentland Utilities review: part one – the beginning

30 RACF control blocks

42 RACF availability

52 Setting up security for JES2

57 RACF – your questions answered

62 August 1995 – February 2003 index

66 RACF news

RACF Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
E-mail: fionah@xephon.com

North American office

Xephon
Post Office Box 350100
Westminster CO 80035-0100
USA
Telephone: (303) 410-9344

RACF Update on-line

Code from *RACF Update*, and complete issues in Acrobat PDF format, can be downloaded from http://www.xephon.com/racf; you will need to supply a word from the printed issue.

Subscriptions and back-issues

A year's subscription to *RACF Update* (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. The price includes postage. Individual issues, starting with the August 1999 issue, are available separately to subscribers for £48.50 ($72.75) each including postage.

Editor

Fiona Hewitt

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.

Contributions

When Xephon is given copyright, articles published in *RACF Update* are paid for at £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above or download a copy of our *Notes for Contributors* from http://www.xephon.com/index/nfc

Printed in England.

Checking dataset access

The program presented here – RACDSACC – can be invoked to check that the user has appropriate RACF access to a dataset before trying to use it. It simply ends with a return code to indicate the user's access.

Also included here is a REXX EXEC which gives an example of how to use RACDSACC. That EXEC is used to invoke EDIT of a dataset from any ISPF panel. If the user has only READ access to the data, it invokes VIEW instead. If access is NONE, a warning message is issued and no attempt is made to EDIT/ VIEW the data.

```
TITLE 'RACDSACC - CHECK RACF FOR DATASET ACCESS'
*----------------------*  FUNCTION  *---------------------------
*
* THIS MODULE PERFORMS A RACF CHECK ON THE INVOKING USER'S ACCESS
* TO A SPECIFIC DATASET, & ENDS WITH AN INFORMATIVE RETURN CODE.
*
*----------------------*  PARAMETERS  *-------------------------
*
* PARAMETERS PARSED   :  'DATASET-NAME,FUNCTION'
*
* ...WHERE DATASET-NAME = FULLY QUALIFIED DATASET WITHOUT QUOTES
*          FUNCTION     = READ, UPDATE, CONTROL, ALTER OR ACCESS
*
* WHEN FUNCTION = 'ACCESS' IT REQUIRES NO APF-AUTHORISATION,
* AND NO RACF MESSAGES ARE GENERATED.
*
* FOR ALL OTHER FUNCTIONS IT RUNS IN KEY ZERO IF POSSIBLE,
* (VIA MODESET IF THE LOAD MODULE IS IN AN APF-AUTHORISED LIBRARY).
* IF IT RUNS IN KEY ZERO IT SUPPRESSES ALL MESSAGES AND LOGGING,
* (VIA THE 'LOG=NOSTAT' PARAMETER ON THE RACROUTE MACRO).
*
*----------------------* RETURN-CODES *-------------------------
*
*   WHERE FUNCTION = READ, UPDATE, CONTROL OR ALTER
*       Ø - ACCESS LEVEL AS SPECIFIED IS OK.
*       4 - DSN IS NOT DEFINED TO RACF (& USER HAS 'SPECIAL' ATTRIBUTE)
*       8 - ACCES LEVEL ISN'T OK.
*
*   WHERE FUNCTION = ACCESS
*       Ø - USER ACCESS IS NONE    .
*       1 - USER ACCESS IS READ
```

```
*       2 - USER ACCESS IS UPDATE
*       3 - USER ACCESS IS CONTROL
*       4 - USER ACCESS IS ALTER
*
*   FOR ALL FUNCTIONS
*      16 - ERROR IN PARAMETER LIST
*      24 - SERVERE ERROR IN RACF CALL
*
*-------------------------* LANGUAGE  *--------------------------
*
* PROGRAMMING LANGUAGE: ASM H
* MACRO'S USED        : GETMAIN,FREEMAIN,SAVE,RETURN,
*                       TESTAUTH,MODESET,RACROUTE
* LINKAGE             : RENT,REUS,AC=1
*
* ------------------------* REGISTERS *--------------------------
*
* R1                    PARAMETER LIST
* R2-1Ø                 -- WORK --
* R11                   BASE OF WORKAREA
* R12                   1. BASE REG OF RACDSACC
* R13                   SAVE AREA ADDRESS
* R14                   RETURN ADDRESS OF CALLER
* R15                   RETURN CODE
*
********************************************************************
RACDSACC CSECT
         SAVE  (14,12)             SAVE REGISTERS
         LR    R12,R15             LOAD BASEADDRESS
         USING RACDSACC,R12        BASE REGISTER ESTABLISHED
         LR    R4,R1               SAVE PARAMETER ADDRESS
         L     R2,WORKBYTS         SIZE OF THE USER PROGRAM
         GETMAIN R,LV=(2)          GET STORAGE FOR PROGRAM VARIABLES
         LTR   R15,R15             DID WE GET STORAGE?
         BZ    GOTSTOR             YES - CARRY ON
         ABEND (R15)               SAVE GETMAIN RC
GOTSTOR  DS    ØH
         LR    R11,R1              R11 CONTAINS ADDRESS OF STORAGE
         USING PGMAREA,R11         ESTABLISH DATA AREA ADDRESSABILITY
         ST    R13,SAVEAREA+4            SAVE AREA CHAINING
         LA    R13,SAVEAREA              NEW SAVEAREA
         ST    R1,A_FMAIN
         LR    R1,R4               RESTORE PARAMETER ADDRESS
********************************************************************
****                     MAIN PROCEDURE                         ****
********************************************************************
         LA    R1,Ø(R1)          REMOVE HIGH ORDER BIT
         L     R2,Ø(R1)          R2 = PARMFIELD
         LH    R8,Ø(R2)          R8 = LENGTH OF PARMFIELD
```

```
         LTR   R8,R8             ZERO?
         BZ    NOGOOD            YES: ERROR
         CH    R8,=H'52'         PARM LONGER THAN 52 BYTES, THEN
         BH    NOGOOD            GOTO LABEL NOGOOD AND SET RC=16
*                                ELSE CONTINUE PARM CHECKING
PARMLOC  EQU   *
         LA    R7,INDSN          ADDRESS OF STORAGE INDSN(44 BYTES)
         LA    R2,2(R2)          ADRESS OF 1 BYTE OF PARM IN R2
         BCTR  R2,Ø              R2 = R2 - 1 CAUSE WE WANT TO INCREASE
*                                THE COUNTER IN THE BEGINNING OF THE
*                                LOOP CALLED GETINDSN
         SR    R9,R9              INITIATE COUNTER FOR BRANCH
         LA    R9,45(R9)                  ON COUNT LOOP
GETINDSN EQU   *
         LA    R2,1(R2)           MOVE POINTER 1 BYTE AND GET NEXT CHAR
         CLI   Ø(R2),C','         IS IT A DELIMITER THEN GOTO GETLEVEL
         BE    GETLEVEL
         MVC   Ø(1,R7),Ø(R2)      MOVE 1 CHAR TO STORAGE
         LA    R7,1(R7)           MOVE POINTER 1 BYTE
         BCT   R9,GETINDSN       END OF LOOP.
         CH    R8,=H'52'         PARM LONGER THAN 52 BYTE, THEN
         B     NOGOOD             NO C',' FOUND WHICH MEANS ERROR OR
*                                 LENGTH OF DSN IS LONGER THAN 44 BYTES
GETLEVEL EQU   *
         BCTR  R9,Ø               R9 = R9 - 1 FOR BRANCH ON COUNT
LOOP     EQU   *
         MVC   Ø(1,R7),=C' '      MOVE 1 SPACE TO DSN IN STORAGE
         LA    R7,1(R7)           MOVE POINTER 1 BYTE
         BCT   R9,LOOP            IS R9 = Ø?
*                                 NO: DECREASE R9 BY 1 AND JUMP TO LOOP
*                                 YES: CONTINUE WITH NEXT INSTRUCTION
NEXTADDR EQU   *
         XR    R5,R5             XOR R5 I.E SET R5 = Ø
         CLC   1(4,R2),=C'READ'  IS ACCESSLEVEL = 'READ'?
         BNE   GETLEV1           TEST FOR NEXT POSSIBLE CHAR STRING
         IC    R5,=X'Ø2'         MOVE CODE FOR READ INTO R5 AND USE IT
*                                WHEN PERFORMING RACROUTE
         B     PARMOK            BRANCH TO PARMOK AND PERFORM CHECK
GETLEV1  EQU   *
         CLC   1(6,R2),=C'UPDATE'  IS ACCESSLEVEL = 'UPDATE'?
         BNE   GETLEV2           TEST FOR NEXT POSSIBLE CHAR STRING
         IC    R5,=X'Ø4'         MOVE CODE FOR UPDATE INTO R5 AND USE
*                                THIS WHEN PERFORMING RACROUTE
         B     PARMOK            BRANCH TO PARMOK AND PERFORM CHECK
GETLEV2  EQU   *
         CLC   1(7,R2),=C'CONTROL' IS ACCESSLEVEL = 'CONTROL'?
         BNE   GETLEV3           TEST FOR NEXT POSSIBLE CHAR STRING
         IC    R5,=X'Ø8'         MOVE CODE FOR CONTROL  INTO R5 AND USE
```

```
*                                THIS WHEN PERFORMING RACROUTE
         B     PARMOK            BRANCH TO PARMOK AND PERFORM CHECK
GETLEV3  EQU   *
         CLC   1(5,R2),=C'ALTER' IS ACCESSLEVEL = 'ALTER'?
         BNE   GETLEV4           TEST FOR NEXT POSSIBLE CHAR STRING
         IC    R5,=X'8Ø'         MOVE CODE FOR ALTER    INTO R5 AND USE
*                                THIS WHEN PERFORMING RACROUTE
         B     PARMOK            BRANCH TO PARMOK AND PERFORM CHECK
GETLEV4  EQU   *
         CLC   1(6,R2),=C'ACCESS' IS ACCESSLEVEL = 'ACCESS'?
         BNE   NOGOOD            ERROR IN PARM FIELD
*
PARMOK   EQU   *
         XC    VOL,VOL           CLEAR OUT VOLUME PARAMETER
         XR    R15,R15           R15 = Ø INITIATE FOR RC
         LA    R4,DATASET
         LA    R6,RACF_WORK
         LA    R7,INDSN
         LA    R8,VOL
*
         CLC   1(6,R2),=C'ACCESS' IS ACCESSLEVEL = 'ACCESS'?
         BNE   CHKAPF            IF NOT - CHECK SPECIFIC ACCESS TYPE
*
* --- WHAT IS HIGHEST ACCESS ALLOWED? ---
         MVC   RACROUTE_LIST(AUTHSLEN),AUTHS   COPY STATIC TO DYNAMIC
*
ACCESS   RACROUTE REQUEST=AUTH,VOLSER=(R8),RELEASE=1.9,                X
               WORKA=(R6),CLASS=(R4),ENTITY=((R7),NONE),               X
               STATUS=ACCESS,MF=(E,RACROUTE_LIST)
*************************************************************
* RACROUTE WILL GIVE THE FOLLOWING REASON CODES              *
*   SAFPRREA=F'0000'        USER ACCESS IS NONE              *
*           =F'0004'        USER ACCESS IS READ              *
*           =F'0008'        USER ACCESS IS UPDATE            *
*           =F'000C'        USER ACCESS IS CONTROL           *
*           =F'0010'        USER ACCESS IS ALTER             *
*************************************************************
         LA    R1Ø,RACROUTE_LIST
         USING SAFP,R1Ø          MAPPING FROM ICHSAFP DSECT
*                                R15 HAS SAF RC (SHOULD BE Ø)
*        L     R15,SAFPRRET      RACF RETURN CODE = 14
         L     R4,SAFPRREA       GET THE RACF REASON CODE
         SRDA  R4,32(Ø)          .. SHIFT IT INTO R5
         D     R4,=F'4'          WE WANT ONLY Ø - 4 RETURNED
         LR    R15,R5            R15 GETS SAFPRREA DIVIDED BY 4
         B     EXITCC            FINISHED -> CLEANUP & RETURN
*
* --- FUNCTION = READ,UPDATE,CONTROL,ALTER -----------
CHKAPF   TESTAUTH FCTN=1         ARE WE APF AUTHORISED?
```

```
         LTR   R15,R15
         BZ    APFAUTH
*
* --- NON-AUTHORISED --------
         RACROUTE REQUEST=AUTH,VOLSER=(R8),                            X
               WORKA=(R6),CLASS=(R4),ATTR=(R5),ENTITY=((R7),NONE),     X
               MF=(E,RACROUTE_LIST)
         B     CHKRC             GO TO CHECK RACROUTE RETURN CODE
*
* --- APF-AUTHORISED --------
APFAUTH  MODESET KEY=ZERO        KEY ZERO NEEDED FOR 'LOG=NOSTAT' PARM
*
         RACROUTE REQUEST=AUTH,VOLSER=(R8),                            X
               WORKA=(R6),CLASS=(R4),ATTR=(R5),ENTITY=((R7),NONE),     X
               LOG=NOSTAT,MF=(E,RACROUTE_LIST)
*************************************************************
* RACROUTE WILL GIVE THE FOLLOWING RETURN CODE IN R15         *
*   IF  R15,=H'ØØ'        CC = Ø WHEN ALL IS OK               *
*   IF  R15,=H'Ø4'        INDSN NOT DEFINED TO RACF           *
*   IF  R15,=H'Ø8'        USER ISN'T ALLOWED TO ACCESS INDSN  *
*************************************************************
CHKRC    CH    R15,=H'Ø8'        IF RACROUTE GIVES COND CODE > 8 THEN
         BH    ERROR             IT'S A BAD ERROR, GO SET UP COND CODE
         B     EXITCC            24 AND EXIT
NOGOOD   EQU   *
         LA    R15,16(R15)       PARM IS INCORRECT
         B     EXITCC
ERROR    EQU   *
         LA    R15,24(R15)       RACROUTE GAVE AN UNEXPECTED ERROR
*************************************************************************
****                     RETURN LINKAGE                              ****
*************************************************************************
EXITCC   EQU   *
         L     R13,4(R13)              RESTORE SAVE AREA
         L     R2,WORKBYTS             SIZE OF WORKING STORAGE
         L     R3,A_FMAIN              ADDRESS OF WORKING STORAGE
         LR    RIØ,R15                 SAVE    RC
         FREEMAIN R,LV=(2),A=(3)       FREE GETMAINED STORAGE
RESTORC  DS    ØH
         LR    R15,RIØ                 RESTORE RC
         RETURN (14,12),RC=(15)        BYE BYE
*
*-----------------------------------------------------------------------
WORKBYTS DC    A(PGMSIZE)        AMOUNT OF WORKING STORAGE NEEDED
         LTORG
DATASET  DC    X'Ø7',CL7'DATASET'
         DS    ØF
RACROUTE_LIST RACROUTE REQUEST=AUTH,MF=L  RACROUTE LIST FORM
*
```

```
AUTHS          RACROUTE REQUEST=AUTH,RELEASE=1.9,MF=L
AUTHSLEN       EQU *-AUTHS
*
         YREGS                     REGISTER EQUATES
         ICHSAFP                   SAF PARAMETER LIST
*
PGMAREA  DSECT
SAVEAREA DS    18F                 SAVE AREA
A_FMAIN  DS    F                   ADDRESS OF WORKING STOR FOR FREEMAIN
RACF_WORK DS   CL512               RACROUTE WORKAREA
INDSN    DS    CL44' '
VOL      DS    CL6' '
         DS    ØD
PGMSIZE  EQU   *-PGMAREA            WORKING STORAGE SIZE CALCULATION
         END   RACDSACC
```

REXX EXEC

```
/*============================== REXX ================================*/
/*  EDI - RECURSIVE EDIT OF A DATASET                  Version 2.Ø    */
/*                                                                    */
/*     This could be defined as an ISPF command in an ISPF command    */
/*     table (in the ISPTLIB concatenation) like:                     */
/*                                                                    */
/*      ED1 : "SELECT CMD(%EDI 'dsname' PANEL('pnl') MACRO('mac')"    */
/*        to always edit the same 'dsname' using the panel and macro  */
/*                                                                    */
/*      ED  : "SELECT CMD(%EDI &ZPARM) NEWAPPL(ISR)"                  */
/*        The user enters parameter(s) like the following:            */
/*          a)  ED             - basic Edit Entry Panel               */
/*          b)  ED dsname      - EDIT "dsname"                        */
/*          c)  ED dsname vol  - EDIT "dsname" on volume "vol"        */
/*          d)  ED dsname parm - EDIT "dsname" with EDIT parameters   */
/*                                                                    */
/*           The char ! is an alternative to ', for                   */
/*           fully qualifying a dataset name.  This                   */
/*           allows: IS 'ED !dsname!' from Info/Man.                  */
/*                                                                    */
/*           Check RACF access before invoking EDIT:                  */
/*           - if only READ access is allowed, invoke VIEW instead    */
/*           - if NO access is allowed, return with a warning message */
/*                                                                    */
/*           If already being edited - invoke VIEW instead            */
/*                                                                    */
/*  written : 1991/Ø5/1Ø     updated: 2ØØ2/Ø6/2Ø      by: Ron Brown   */
/*====================================================================*/
```

```
 Address ISPEXEC
"CONTROL ERRORS RETURN"
 Parse Upper Arg dsn parm              /* get dataset-name & parms */
 If dsn = '' Then
    "SELECT PGM(ISREDIT) PARM(P,ISREDMØ1)"     /* Edit Entry Panel */
 Else Do
    dsn = Translate(dsn,"'","!")       /* allow for ! instead of ' */
    p = Pos('(*)',dsn)                 /* member mask '(*)' is not */
    If p > Ø Then                      /* supported by EDIT or VIEW*/
       dsn = Left(dsn,p-1)!!Substr(dsn,p+3)        /* remove '(*)' */

    /*---------------------------------*/
    /* check RACF access to the dataset */
    /*---------------------------------*/
    Parse Var dsn dataset '(' .        /* remove any member */
    If Pos("'",dataset) = Ø Then Do    /* no quotes, so add prefix */
       prefix = Userid()
       If Sysvar('SYSPREF') <> prefix Then
          prefix = Sysvar('SYSPREF')'.'prefix
       dataset = prefix'.'dataset
       End
    Else
       dataset =Strip(dataset,'B',"'")  /* remove quotes */

    Address TSO "CALL *(RACDSACC) '"dataset",ACCESS'"

    Select
      When rc = Ø Then
         ZERRLM = ' *** You are not authorised to',
                  'access dataset:' dataset
      When rc = 1 Then Do
         act = 'VIEW'
         ZERRLM = ' *** EDIT changed to VIEW because you have',
                  'only READ access ***'
         End
      Otherwise     /* rc > 1 */
         act = 'EDIT'
      End
    If ZERRLM <> 'ZERRLM' Then Do
       ZERRALRM = 'YES'
       ZERRHM   = '*'
      "SETMSG MSG(ISRZØØ2)"
       End
    If act = 'ACT' Then Exit 8  /* not authorised for access */

    /*-----------------------------------------------*/
    /* build the parameters and invoke EDIT (or VIEW) */
    /*-----------------------------------------------*/
    edparms = 'DATASET('dsn')'
```

```
      If Length(parm) = 6 & Pos('(',parm) = Ø Then
         edparms = 'DATASET('dsn') VOLUME('parm')'
      Else
         edparms = 'DATASET('dsn')' parm

      act edparms     /* invoke EDIT (or VIEW) */
      End
   Edit_RC = rc
   If ZERRMSG <> 'ZERRMSG' Then   /* only set when error occurs */
     "SETMSG MSG("ZERRMSG")"    /* show standard message */
/*   "SETMSG MSG(ISRZØØ2)"    <-- this message gives same result     */

   If Edit_RC = 14 Then      /* EDIT no good because 'data in use'   */
      "VIEW" edparms         /* ..... do a VIEW instead of EDIT      */

   Exit Edit_RC
```

IMPLEMENTATION

Note that the recommended link parameters are RENT,REUS,AC=1. The RACDSACC module should ideally go into an APF-authorized library to avoid any RACF messages and logging. But you can still invoke it from an unauthorized program or TSO EXEC/CLIST. The possible return codes are listed in the comments at the start of the program source. The EDI EXEC also has comments at the start to explain its use.

Ron Brown
(Germany)

Code from *RACF Update* articles

As a free service to subscribers and to remove the need to rekey the scripts, code from individual articles of *RACF Update* can be accessed on our Web site, at

http://www.xephon.com/racf

You will need the user-id shown on your address label.

RACF in focus – the Search command

'RACF in focus' is a regular column focusing on specific aspects of RACF. Here, we explore some of the functions of the Search command.

The Search command in RACF has many useful and powerful features, and a complete description can be found in the *RACF Command Language Reference Manual*. The command can be used to perform a variety of housekeeping tasks, as well as being a useful investigative tool – for example, it allows you to review how a dataset high-level index is protected.

The power and capabilities of the Search command are best illustrated by looking at several real-life examples.

REVOKING INACTIVE USERIDS

One of the tedious tasks performed by RACF administrators is to sort out a list of all the RACF users that haven't used the system for a certain period. These userids could be misused by hackers, and need to be dealt with. Your auditors may even require that you conduct a periodic review of all userids that haven't accessed the system for the last 100 days (or 150 days, or some other number, depending on your installation's security policy).

Let's say that you want to find all the userids that haven't accessed the system in the last 200 days. You also want to revoke all these userids. The following command:

```
SEARCH  CLASS(USER) AGE(2ØØ) CLIST('ALTUSER  '  '  REVOKE')  NOLIST
```

will create a CLIST under your name, called 'YOUR-USERID.EXEC.RACF.CLIST', containing the RACF commands required to revoke all userids that haven't accessed the system in the last 200 days.

The sample output would look something like this:

```
ALTUSER IDTEMP REVOKE
ALTUSER TEST123 REVOKE
ALTUSER USER276 REVOKE
ALTUSER USER3 REVOKE
ALTUSER USR9999 REVOKE
```

You can edit the list, and review each revoke for applicability. If there are some userids you don't want to revoke – disaster recovery userids, for example – simply remove them from the list before executing the CLIST. When you're ready to revoke, execute the CLIST in TSO:

```
EXEC 'YOUR-USERID.EXEC.RACF.CLIST'
```

You can even set up a procedure that will do all this over regular intervals, and keep your RACF database clean.

FINDING OUT HOW PROJECT DATASETS ARE PROTECTED

Another housekeeping challenge RACF administrators face is to review and clean up redundant profiles for specific projects. These profiles may have been useful at one time, but are no longer applicable.

For example, the PAYROLL high-level index may have many profiles covering its datasets. Imagine that you want to list all of them and review them to see whether they're still relevant. The following command will give you the desired list:

```
SEARCH FILTER(PAYROLL.**) CLASS(DATASET)
```

The sample output from this command will look as follows:

```
PAYROLL.PROD.PROJECT1.** (G)
PAYROLL.PROD.PROJECT2.** (G)
PAYROLL.RLSE2.PROD.**  (G)
PAYROLL.TEST.**  (G)
```

You can now decide whether these profiles adequately protect your PAYROLL project datasets. Depending on your needs, you may want to create more profiles to make the protection more granular; or you may find that, for example, the profile

PAYROLL.RLSE2.PROD.** is no longer relevant, since there are no datasets that match the profile. In this case, you can delete the profile. You may even find that you need less granular protection now, in which case you can collapse multiple profiles into one.

You can, of course, use this version of the command to list resource classes, too. The following command:

```
SEARCH FILTER(**) CLASS(CICSPRD)
```

will list all the profiles in the CICS class CICSPRD.

FINDING ALL DISCRETE PROFILES

There are several disadvantages to discrete profiles, the most common being that they only protect a single specific dataset, and that deleting the dataset also deletes the profile.

If you want to see all the discrete profiles at your installation, use the following command:

```
SEARCH FILTER(**) CLASS(DATASET) NOGENERIC
```

Any discrete profiles will be listed, and you can take steps to remove them after building their corresponding generic profiles.

FINDING PROFILES IN WARNING MODE

When implementing new profiles, you may have specified WARNING, so users would continue to get access to the resource (with a WARNING message) even though they may not have sufficient authority.

You can use the Search command to find all profiles with the Warning indicator. The following command will do this for dataset profiles:

```
SEARCH FILTER(**) CLASS(DATASET) WARNING
```

The output will be a list of dataset profiles. If you want to remove

the WARNING indicator from all, or some, of these profiles, you can do it as follows. First create a CLIST that will build RACF commands for you:

```
SEARCH FILTER(**) CLASS(DATASET) WARNING CLIST('ALTDSD  ' '  NOWARNING')
```

Then execute the CLIST in TSO:

```
EXEC 'YOUR-USERID.EXEC.RACF.CLIST'
```

SUMMARY

The Search command can be used for a number of RACF housekeeping tasks, and there are more uses besides. Take a look at the command syntax in the IBM manual, and you're sure to find something specific to your needs.

Dinesh Dattani (dddattani@rogers.com)
Security Consultant (Canada)

Pentland Utilities review: part one – the beginning

This article reviews a collection of DOS/Windows-based utilities written and distributed by Nigel Pentland for use with RACF. The utilities cover many reporting and JCL-building functions, which are most useful in an organization's Information Security arena.

Because there are so many utilities available and we'll be looking at them all, the article will be in two parts. Part one will deal with:

- History
- Set-up and use
- General reports
- User ID reports
- Group reports

and part two (in the next issue) with:

- Dataset reports
- General resource profile reports
- CICS reports.

I'd like to take this opportunity to thank Nigel Pentland, first, for developing these utilities, and second for reviewing the draft of these articles to make sure I didn't have too many obvious mistakes.

I'd also like to mention that this review covers my experience and my opinion of the various utilities, and does not necessarily reflect the views of anyone else. Your view of the Pentland Utilities may be quite different from mine, based on your own experience. But if you've never used these utilities, here's a chance to get an idea of what you're missing.

HISTORY

Nigel Pentland developed the first five utilities around 1995, after being transferred to mainframe security, attending a RACF course, and discovering just how limited RACF's reporting capabilities are. Since he didn't have experience in DB2 programming at that time, Nigel decided to go with PC programming tools he was familiar with (particularly Microsoft C compiler V5.1).

Over the months and years, the five programs evolved into almost 90. In May 2000, he converted the programs from 16-bit DOS code to 32-bit Windows code, and converted the original text output into HTML. He also removed around 24 utilities that were redundant, or just didn't work up to the level he considered sufficient. Some of these were pretty good, and I was sorry to see a few of them go (although the rest were, shall we say, not as nifty as they could have been).

My history with Pentland Utilities

I started working with the Pentland Utilities when I first went to Saudi Arabia, in around 1998. I heard about them on the RACF-L discussion group, which was (and is) a very good resource for RACF-related information. After several problems in downloading the files (Saudi Arabia didn't have the Internet at this point, and I was forced to link to his Web site long-distance through Bahrain, but that's another story), I finally got a copy of the utilities, and the RACF world opened up to me. I learned just how screwed up our RACF database was (and is), and came up with my ideas for RACF restructuring (see my four-part article on this topic completed last issue). I found ways to identify problems at an early stage, and to try to pre-empt them. I even found things I never would have thought of looking for in the first place, things that can leave gaping holes in your mainframe's security. And, most importantly (at least according to my employer), they're FREE!

SET-UP AND USE

To use the Pentland Utilities, you'll need to download several items from Nigel's Web site, located at http://www.nigelpentland.co.uk. The latest version (as of this writing) is 1.10, dated 04 July 2001. You can also find them on the CBTTAPE and the CBTTAPE CD. Currently there are 72 utilities in all.

- RACF.ZIP contains the executable RACF utilities.
- RACF.INI contains the parameters for running the utilities.
- ReadMe.htm contains the explanation of each utility, including output.

Download these and expand them into an x:\RACF directory. If possible, put this on a local drive and not on a LAN – considering the size of the RACF flatfile you'll need to download from your mainframe, running these jobs across a network can slow you to a crawl.

If you're running multiple LPARs, set up a naming convention for each LPAR's flatfile. For example, if you have 3 LPARs named DEV1, PRD1, and TST1, your directory structure should look something like this:

```
x:\RACF\                    (for program storage)
       \DEV1                    (for IRRDBU00 flat file from DEV1)
           \JCL                         (for JCL output)
            \TEXT                       (for text [HTML, actually]
output)
       \PRD1                    (for IRRDBU00 flat file from PRD1)
            \JCL                        (for JCL output)
            \TEXT                       (for text output)
       \TST1                    (for IRRDBU00 flat file from TST1)
            \JCL                        (for JCL output)
            \TEXT                       (for text output)
```

For each LPAR, you need to have a modified RACF.INI file, a flatfile from IRRDBU00, and a flatfile DSMON report.

Here's what the RACF.INI file should look like if you're running utilities for a flatfile from the DEV1 LPAR:

```
[General]
input_file=x:\RACF\DEV1\FLATFILE.txt
output_jcl_file=x:\RACF\DEV1\JCL\DEV1-
output_text_file=x:\RACF\DEV1\Text\DEV1-

[Expired_users]
Last_access_date=2001-09-01
;format YYYY-MM-DD

[Revoked_connections]
Today's_date=2002-01-01
;format YYYY-MM-DD

[JCL]
header=//PENTLAND JOB (ACCT#),'PENTLAND UTILITIES JCL',CLASS=A,
header=//         MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
header=/*ROUTE XEQ DEV1
header=/*ROUTE PRINT DEV1
header=//RACF     EXEC  PGM=IKJEFT01
header=//SYSTSPRT DD SYSOUT=X
header=//SYSTSIN  DD *
header= PROFILE NOPREFIX
footer=/*
footer=//
```

For the IRRDBU00 JCL, structure it as follows:

```
//PENTLND1 JOB (ACCT#),'IS SECURITY ADMIN',CLASS=A,
//         MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*ROUTE XEQ DEV1
/*ROUTE PRINT DEV1
//*********************************************************************
//**                                                                 **
//** THIS CREATES A RESTRUCTURED RACF DATABASE                       **
//**                                                                 **
//*********************************************************************
//COPY     EXEC PGM=IRRUT2ØØ,PARM='NOLOCKINPUT'
//SYSPRINT DD SYSOUT=X
//SYSRACF  DD DSN=SYS1.RACF.DATABASE,DISP=SHR
//SYSUT1   DD DSN=NEW.RACF.DATABASE,
//            DISP=(NEW,KEEP,KEEP),
//            SPACE=(CYL,(2Ø)),
//            DCB=(RECFM=F,LRECL=4Ø96),
//            UNIT=SYSDA
//SYSUT2 DD SYSOUT=*
//SYSIN DD *
INDEX
MAP
END
//*********************************************************************
//**                                                                 **
//** THIS CREATES A SEQUENTIAL FLATFILE FROM THE RACF BACKUP DATABASE **
//**                                                                 **
//*********************************************************************
//UNLOAD   EXEC PGM=IRRDBUØØ,PARM=NOLOCKINPUT
//SYSPRINT DD SYSOUT=X
//INDD1    DD DSN=NEW.RACF.DATABASE,DISP=SHR
//OUTDD    DD DSN=NEW.RACF.FLATFILE,
//            DISP=(NEW,KEEP,KEEP),
//            SPACE=(CYL,(8Ø,5),RLSE),
//            DCB=(RECFM=VB,LRECL=4Ø96,BLKSIZE=28672),
//            UNIT=SYSDA
//
Your DSMON JCL should be something like this -
//PENTLND2 JOB (ACCT#),'IS SECURITY ADMIN',CLASS=A,
//         MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=&SYSUID
/*ROUTE XEQ DEV1
/*ROUTE PRINT DEV1
//DSMFILE  EXEC PGM=ICHDSMØØ,REGION=8M
//*********************************************************************
//**                                                                 **
//** THIS JCL GENERATES THE DSMON REPORT AND SENDS IT TO A FLATFILE   **
//**                                                                 **
```

```
//*******************************************************************
//SYSPRINT DD  SYSOUT=X
//SYSTSPRT DD  SYSOUT=X
//SYSUT2   DD  DSN=NEW.RACF.DSMON.FLATFILE,
//             DISP=(NEW,KEEP,KEEP),
//             SPACE=(CYL,(5,2),RLSE),
//             DCB=(RECFM=FB,LRECL=133,BLKSIZE=28329),
//             UNIT=SYSDA
//SYSTSIN  DD  *
  LISTGRP (*)
  LINECOUNT 55
  FUNCTION ALL
/*
//DSMPRINT EXEC PGM=ICHDSMØØ,REGION=8M
//*******************************************************************
//**                                                               **
//** THIS JCL GENERATES THE DSMON REPORT AND SENDS IT TO A PRINTER  **
//**                                                               **
//*******************************************************************
//SYSPRINT DD  SYSOUT=X
//SYSTSPRT DD  SYSOUT=X
//SYSUT2   DD  SYSOUT=X
//SYSTSIN  DD  *
  LISTGRP (*)
  LINECOUNT 55
  FUNCTION ALL
/*
//
```

When you've completed the two jobs above, download the output files into your PC. You can do this quite simply if you have IBM Personal Communications (PComm) Version 3 or above. While in TSO, go into option 6, then click ACTION on the toolbar, and choose "Receive File From Host..." and download the files in text format (not binary!). This is the IND$FILE option. You can also do this transfer through FTP (if your shop is up to that version), which is quite a bit faster. When you've copied the file to your PC, make sure you give it the name listed in your RACF.INI file (or change the name in the RACF.INI file).

When you're ready to run the jobs, open a COMMAND.exe frame in Windows. Go to the x:\ drive where your programs are located, type the following commands, and you're ready to roll:

```
cd \RACF\DEV1
path %path%;x:\RACF
```

GENERAL REPORTS

The five general reports are as follows:

- *RACF00*. Flat file pre-processor.
- *RACF01*. Summary statistics.
- *RACFAWK*. Search given record type for matching string.
- *RACFDIAG*. Diagnostic utility for developing and maintaining code.
- *RACFJCL*. General-purpose JCL generation.

Generally, the first programs you should ever run are RACF 00 and RACF 01. RACF 00 takes care of a nasty little problem that crops up for shops that have tried to upgrade their operating system (and RACF as well), and then gone back. This pre-processor filters the flatfile to get rid of any funky non-ASCII characters, which can cause the other utilities to give you nothing but empty reports. I was running into this little problem for a few months before RACF00 came out, and it was a nightmare.

RACF 01 gives you a brief HTML summary page of how many records you have for each data type. If you wish, you can compare this to the count generated in the IRRDBU00 program, just as a double-check. All in all, a nifty little page, but not really all that vital unless you've been experiencing problems with your database structure. It does, however, give a brief explanation of all the different RACF record types, which can come in handy if you want to extract specific record types.

Nigel included another quick utility to help in his development of utilities, called RACFDIAG. You'll get a brief text (not HTML) summary that looks like this:

```
DIAGNOSTIC SUMMARY
Max line length allowed =  1500
Max actual line length  =  1999

Max number of groups allowed =  2000
Actual number of groups      =  3532
```

```
Ratio of numbers of groups:bytes required = 1:18

Max number of owners allowed =  2000
Ratio of numbers of groups:bytes required = 1:2000

Max access list size allowed = 30000
Actual access list size      =  4372
Ratio of list size:bytes required = 1:9
```

You'll note statements like "Max number of groups or owners allowed = 2000". This deals, not with restrictions within RACF, but with restrictions within some earlier iterations of the Pentland Utilities. However, it can help clue you in to potential problems in your database, although you're going to get more detailed information from running IRRUT200 with MAP and INDEX. This one is a lot easier to read, though.

RACFAWK gives you the opportunity to extract specific records from the flatfile. This does require some knowledge of the field structure of the database, and knowledge of the various record types. You can get the latter quite simply from looking at the RACF01 output. So, for example, if you want to extract all of the basic data on dataset records, you'd run the following command in your DOS frame:

```
RACFAWK 0400 1 0400
```

And the data will be put into a text file. You can use this in conjunction with a utility called 'cols.exe' which is available in the optional DOS text processing utilities (DOSUTILS.zip). Or you can import it into Excel as a text file and select fixed widths (assuming you know the file layouts for each of the different record types).

Finally, RACFJCL works a bit like a CLIST option, building RACF command JCL around the output from another Pentland Utility. I've never worked with this one, mainly because it seemed easier to take the text output and use Microsoft Word to do the editing for me. Also, this only works with utilities that use text output instead of HTML. However, once again, you can import HTML into Word or Excel and delete the unwanted columns.

USER ID REPORTS

There are 19 Pentland programs that deal with user profiles (note: an asterisk next to the program name means that it generates JCL):

- *RACF02**. Profiles owned by non-existent user IDs.
- *RACF05**. Expired users.
- *RACF08*. Search on user name or string.
- *RACF09**. List users based on mask (JCL to RESUME).
- *RACF11**. List group or user access (ie XREF).
- *RACF18**. Generate data file for fast search of user IDs.
- *RACF21**. Delete ID from access lists.
- *RACF24**. List revoked users.
- *RACF36**. Compare 2 groups of user IDs.
- *RACF38**. List users with additional RACF authorities.
- *RACF46**. Delete user IDs.
- *RACF52*. List profiles and access lists belonging to owner ID.
- *RACF66*. List details of a single user ID.
- *RACF79**. List ALL user IDs with OWNER not the same as DFLTGRP.
- *RACF82*. Annotate list of user IDs giving name DFLTGRP last access etc.
- *RACF84*. List user IDs with 2nd and 3rd numeric chars but not X*nnnnnn* format.
- *RACF86**. Generates JCL to put user IDs in input list into LIMBO.
- *RACF88*. List user IDs with an OMVS UID.

- *RACF92*. List all user IDs and details (name, owner, etc).

RACF 02 is a nifty replacement for IRRRID00, IBM's answer to user ID deletion. If you use IRRRID00 with no input file, it'll find all of the 'orphaned' profiles – ie, those profiles remaining after you delete an ID without deleting the profile using IRRRID00 to delete the ID (are you with me so far?). RACF02 is much simpler to work with. It generates the JCL automatically, and the output file gives you a summary of all the stuff you're getting rid of. Very helpful to those of us who have to deal with RACF administrators who don't follow deletion procedures.

RACF 05 is another IRRRID00 replacement, but this one gets rid of actual user IDs that have not been used for *xx* number of days. You determine that number in the RACF.INI file (see the 'Expired Users' parameter). You'll note that the dates in the ini file show a 3 month difference between Last_access_date=2001-09-01 and Today's_date=2002-01-01. This can be changed to your own organization's desired user ID deletion standards. The output generates JCL to be uploaded to your mainframe, so that you can nuke all those oldies but mouldies. One point of importance here: be sure you double-check that list before you run it. More often than not, it will contain most if not all of your Started Task IDs, and if you delete those, your Operations and Technical Support staff will be coming for you. And trust me, they won't be delivering doughnuts! (However, you may have a hole in the middle after they're through with you....)

RACF 08 does a search of user IDs based on a search string (part of a name or part of the user ID or something in the Installation Data field). Generally, I don't use this one all that much. Instead, I generate a RACF92 list with all the details, and then use the search facility in Internet Explorer or MS Word. Even so, it's a pretty good utility.

RACF 09 works for those who want to search on a user ID mask (all IDs starting with A, or AB, or ABC). It also builds JCL to

RESUME all of the IDs under that mask. This works well if you have a standardized naming convention for your user IDs (okay, you can stop laughing now).

RACF 11 is one utility I like a lot. It's a cross-reference for user IDs and/or groups (you'll see it listed in both sections). It's much better than IRRUT100, because it actually gives you the level of access for the various connected profiles. It generates JCL to permit access (and at the correct level) to the profiles. I use this quite a lot, especially before I'm about to do a string of user ID or group deletions. So, in the unlikely event of my making a mistake (I said stop laughing, okay?), I have a way to recover someone's access.

RACF 18 generates a simple text file output with one user ID per line. Not that thrilling, you might think, but it does come in handy if you want to create a batch file to do RACF11s or RACF66s of every user ID. The adventure of creating a batch file in MS Word, with multiple commands for each user ID, is a thrill that shouldn't be missed. And it works very well indeed. Or, if you're not that adventurous, you can use the text file in a generalized search.

RACF 21 lists the accesses for a user ID or group. It then generates the JCL to remove that ID's access from any and all profiles. However, it doesn't delete the ID or group itself. Also, the ID or group remains attached to its owner, so it still has limited access. This is quite a good thing to have if you need to suspend someone's access temporarily.

RACF 24 lists revoked user IDs, and generates JCL to delete them if they have no TSO segment. I suppose this is to get rid of CICS-only user IDs. I don't use this too much. I figure that, if I'm going to nuke IDs, I'll get rid of the TSO and non-TSO IDs at the same time. However, you may be a bit more pernickety than me.

RACF 38 lists IDs with higher than normal authority. This is

similar to part of the DSMON 'Selected User Attribute Report'. It's a lot more readable, more comprehensive, and easier to work with than IBM's offering (shocking, isn't it?).

RACF 46 takes a list of user IDs (in text format, one ID per line with no spaces), and generates the JCL to delete them all. This one is useful if you can get a list of IDs from your Human Resources department for immediate deletion (you're laughing again, aren't you?).

RACF 52 will list all the profiles and access lists belonging to a specific owner ID (either user ID or group). It doesn't generate any JCL, but it does give you an idea of who created or owns a specific profile. This is very useful for RACF administration, which usually defaults to the user ID of the person who created the profile, instead of the actual owner.

RACF 66 is kind of a glorified LU command, but with more detailed information and additional data on TSO, OMVS, and NetView extensions. Pretty nice, but Nigel, could you parse the installation data so that it looks like a screen output (using the Courier font as well)? It would make it a bit more readable.

RACF 79 shows you all of the user IDs that are not owned by their default group (DFLTGRP). If you're running RACF V2.8 or higher, you'll find two IDs that always show up – irrcerta and irrsitec (those two annoying lower case IDs that nobody likes). You can't change them, so ignore them. However, if you've got regular IDs that don't match up properly, you should take steps to correct them.

RACF 82 will annotate a list of user IDs from an input file (one user ID per line, no spaces) and include information on the user name, date last accessed, DFLTGRP, etc. If you've got a long list of IDs to annotate, this will take a while. Quite frankly, if you can get away without the DFLTGRP information, you'd be better off using RACF92 instead (see below). It gives you every ID, but it is faster.

RACF 84 is a little strange. It will list all user IDs where the

second and third positions are numeric, but only if the entire ID does not match *Xnnnnnn* where X is any alpha character and n is any numeric. I've never been able to use this one, mainly because my installation didn't have IDs that fit this particular scenario. However, if you do, enjoy!

RACF 86 gets rid of user IDs you normally can't remove, because they still own datasets. It assumes you have a group named 'LIMBO', and then goes through some mumbo-jumbo to actually get rid of the dataset connections. This one is really nice to have around, especially if you make the mistake of creating dataset profiles by creating a user ID for them instead of a group (which is what you should do, actually).

RACF 88 is for all of you folks who need to identify that pesky OMVS segment. You know, the one that RACF doesn't show in any of its reports. Very handy, and quite helpful when you need to identify all the people who have 0000000000 as their UID (in other words, Supervisor level).

RACF 92 is one of my favourites in the user ID category. According to Nigel, it was added to the list following comments and suggestions from Jeff Loewenstein, an Auditor turned Administrator (thanks, Jeff!). It lists all of the IDs with some pretty helpful details, such as name, Owner, Create Date, Date Last Accessed, and whether or not the ID has been revoked. It's useful if you need to do a search on a partial name, or if you just want to impress your boss with how many IDs you have in a particular LPAR.

GROUP REPORTS

There are 16 Pentland programs that deal with Group profiles:

- *RACF03*. Group tree structure.
- *RACF04**. Group list with owner ID and truncated installation data.

- *RACF06**. List group with user ID, names, and authority.
- *RACF11**. List group or user access (ie XREF).
- *RACF36**. Compare two groups with user IDs.
- *RACF47**. Change default group and owner of a group of users.
- *RACF50*. List user IDs in a group, with all other connect groups.
- *RACF52*. List profiles and access lists belonging to owner ID.
- *RACF53*. List group (last access, TSO, CICS).
- *RACF69**. List revoked group connections.
- *RACF70*. Count number of users in a group.
- *RACF73*. List user IDs and names in a group, with all other connect groups.
- *RACF75*. User ID group (only one space per line).
- *RACF76**. User ID name connect-owner authority special.
- *RACF77**. Connects with connect-owner other than group.
- *RACF87*. List groups with an OMVS GID.

RACF 03 is a much nicer group tree report than you get in DSMON. It actually comes in two sections (in the same HTML report, of course). The first gives you a list of all the groups, with their installation data (Nigel, check out my comment on RACF66 regarding the format). The second section is a much nicer tree, and it includes the number of users per group. Very slick!

RACF 04 lists all groups with their owner ID and with the installation data (in a non-wrapped format). This is pretty nice too, but you get most of the data in RACF03 (except for the owner ID, of course).

RACF 06 lists all the user IDs connected to a particular group,

and also creates JCL for removing all the IDs from that group. This can be of particular use if you have a large group you want to get rid of, and don't want a bunch of orphaned profiles left in the database.

RACF 11 was mentioned in the user ID section above, so we won't go over it again here.

RACF 36 compares two groups, and lets you know if there are any user IDs common to both. So if you have two groups whose access may be in conflict, run this utility to see which IDs need to be moved (or removed) from one or both. This requires some knowledge of just what your groups have access to.

RACF 47 lists a group, and then creates the JCL to change the user IDs attached to that group, to a new group, and then remove them from the old group (you follow me so far?). Very helpful if you're renaming or restructuring your group profiles.

RACF 50 lists all the users in a particular group, with the DFLTGRP as well as all of the other groups each user ID is connected to. I like this one, although it doesn't include the user name. However, RACF73 does the same report and includes the user name, so it's an either/or choice for you.

RACF 52 was mentioned in the User ID section above, so we won't go over it again here.

RACF 53 lists a group, showing the User ID, name, date last accessed, whether or not access to that group is revoked, whether there is a TSO or CICS segment attached, and whether the CICS segment has a timeout value. Valuable for those groups which mix TSO and CICS assignments. The timeout value is a nice touch.

RACF 69 gives you a list of all the revoked group connections. However, this doesn't work quite as well as intended (that's Nigel's comment, not mine). You have to put the revoked connection date into the RACF.INI file for this to give you an accurate pointer. Still, it's helpful in finding all those annoying group-revokes.

RACF 70 just counts the number of users in a particular group. Quite frankly, you're probably better off just using the count information you get from the Group Tree report in RACF03.

RACF 73 was mentioned above in RACF50. I prefer to have the user's name in a report anyway, so I like this one a bit better.

RACF 75 generates a list of user IDs connected to a group in text format (not HTML,) with the user ID and group separated by a single space. I've never found a particular use for this utility myself, but Nigel's readme file does give some ideas.

RACF 76 is pretty much the same as RACF06, but has the added value of including the connect owner.

RACF 77 finds all of those group connections, where the connect is not owned by the group it's connected to. It shows where a connect has been done where the owner is not explicitly set to the same group. It also creates the JCL to correct this for the errant user IDs, to put them into line.

RACF 87 is the group version of RACF88. This lists all of the groups with an OMVS GID. You should run the two together and compare them, just so you have a view of your entire OMVS universe.

CONCLUSION

Part two of this article (next issue) will cover the remaining Pentland Utilities, which include dataset reports, general resource reports, and CICS reports.

Doc Farmer
Manager and Senior IS Security Analyst
(Middle East)

RACF control blocks

This article is intended for system programmers and software developers who require some basic knowledge of RACF control blocks. It also includes a section on SAF and RACF customizable modules. A future article will look in detail at RACF macros and exits.

RACF COMMUNICATION VECTOR TABLE.

The RACF communication vector table (RCVT) is the primary RACF vector table. There is one RCVT for each system image (OS/390 or z/OS), created at IPL by RACF initialization. The RCVT resides in SQA storage sub-pool 245 Key 0, and is pointed to by field CVTRAC in the CVT. The RCVT is mapped by the ICHPRCVT macro.

The following code can be used to obtain addressability to the RCVT:

```
L     R15,CVTPTR
USING         CVT,R15
ICM   R15,B'1111',CVTRAC
BZ    NO_RCVT
USING       RCVT,R15
```

The RCVT is the source for checking many system-wide options and obtaining the pointers to key RACF central storage areas. If RACF is not installed, the RCVT doesn't exist, but it's the responsibility of other OEM security products to create an RCVT look-alike.

The RCVT contains the following pointers and information:

- Pointer to the Class Descriptor Table (CDT).
- Pointer to the Data Set Descriptor Table (DSDT).
- Pointer to the Range Table (RNGP).
- Pointer to the RACF Authorized Caller Table (ICHAUTAB).

- Pointer to the Naming Convention Table (ICHNCV00).
- Pointer to the Started Procedures Table (ICHRIN03).
- Pointers to most RACF exits.
- RACF processing options (eg Audit, SETROPTS, Password, List of groups, Protect-All, JES).
- Class-related options (Active, Generic RACLISTed).

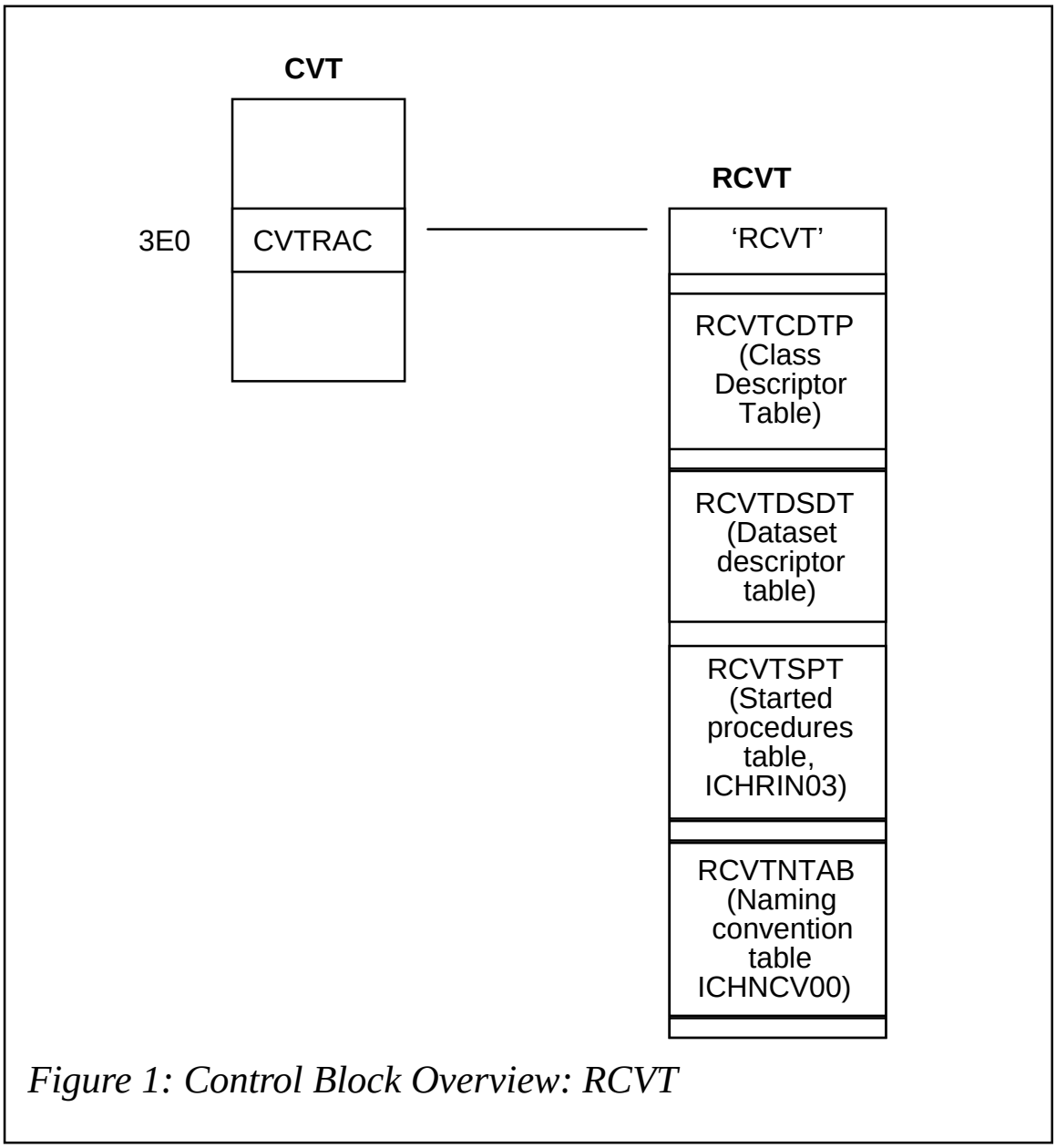

Figure 1: Control Block Overview: RCVT

- The RACF Dataset Name.
- Pointer to the RACF Dataset UCB.

Figure 1 gives an overview of the RCVT control blocks.

INVENTORY CONTROL BLOCK.

The first block in the RACF database is the Inventory Control Block (ICB). This provides the beginning pointers for all other block types. Each RACF database has an ICB, but RACF uses only the ICB for the master primary dataset when determining the setting of options. The ICB is shared by all LPARs that are using the same RACF database. An in-storage copy of the ICB is located in ECSA storage sub-pool 231 key 0, pointed to by field DSDEHDR in the DSDT data area. The ICB also contains:

- The RBA of the highest level index block
- The number of BAM blocks in the RACF dataset
- The BAM high water mark
- The RBA of the first block of the index sequence set
- Change count arrays
- SETROPTS options
- Audit options
- CLASS masks.

SAF ROUTER VECTOR TABLE.

The SAF Vector Table (SAFV) is created by SAF initialization and resides in sub-pool 245 key 0 storage. It is pointed to by field CVTSAF in the CVT and is mapped by the ICHSAFV macro.

The following code can be used to obtain addressability to the SAFV:

```
L     R15,CVTPTR
USING       CVT,R15
```

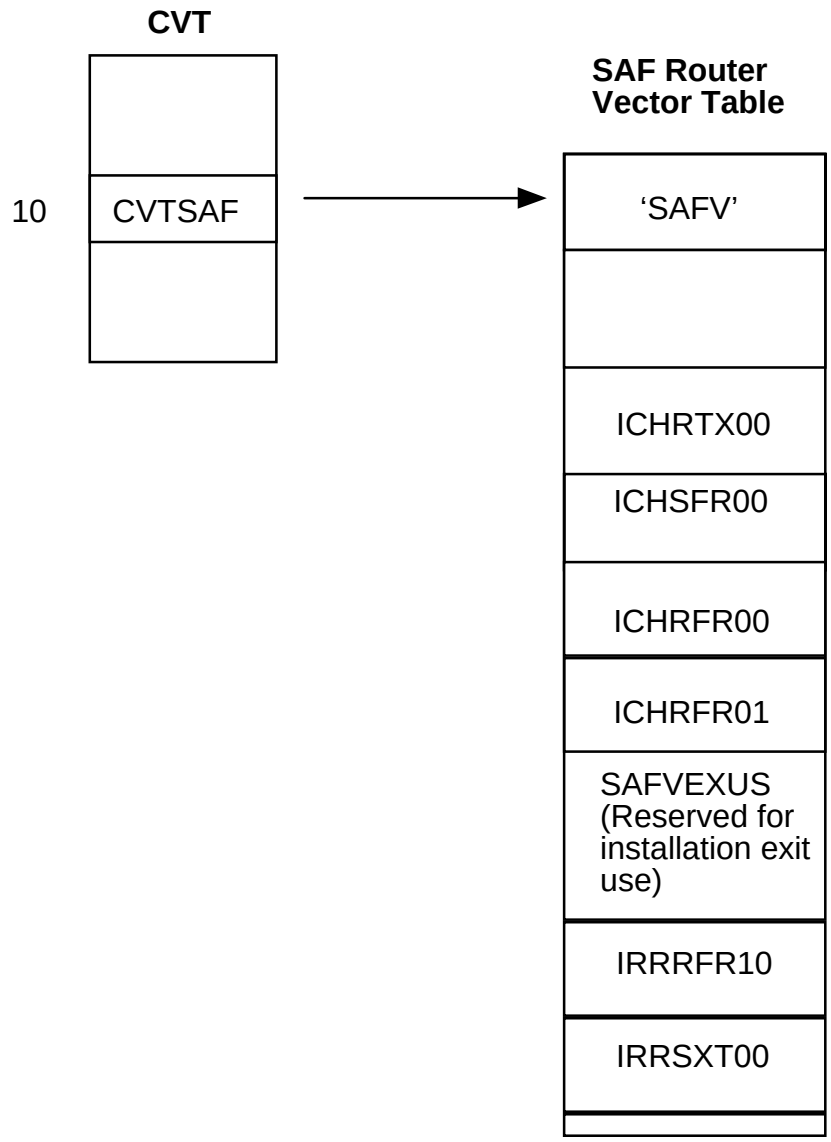

Figure 2. Control Block Overview: SAFV

```
L      R15,CVTSAF
USING         SAFV,R15
```

Figure 2 gives an overview of the SAFV control blocks.

(A full discussion of SAF can be found in the section entitled 'System Authorization Facility' below.)

ACCESSOR ENVIRONMENT ELEMENT.

The Accessor Environment Element (ACEE) represents a single signed-on userid. Technically, it's an MVS not a RACF control block. It's mapped by the IHAACEE macro. The ACEE is created and deleted by the following RACINIT processes:

```
RACROUTE REQUEST=VERIFY ENVIR=CREATE
```

and

```
RACROUTE REQUEST=VERIFY ENVIR=DELETE
```

If a user application – such as CICS – has multiple users in a single address space, each user needs a separate ACEE. When the ACEE is created, a UTOKEN is also created. If RACF isn't active when the ACEE is created, SAF builds a default ACEE.

The first ACEE of an address space is stored in the ASXB (ASXBSENV) of the address space. If a second RACROUTE request creates an ACEE without an ACEE address being specified, the address is stored in the TCB (TCBSENV) of the program that issued the RACROUTE request.

The address space level ACEE (ASXBSENV) is normally

Function	Created	Deleted
TSO user	TSO log-on	TSO log-off
Batch job	Job initiation	Job termination
Started class	STC initiation	STC termination

Figure 3: Functions for address space level ACEE (ASXBSENV)

Function	Created/deleted
JES2 sub-task	JES processing
Batch job	FTP log-on/log-off
Other	Third-party RACHECK

Figure 4: Functions for TASK level ACEE (TCBSENV)

created under the functions shown in Figure 3. Figure 4 shows what happens when a TASK Level ACEE (TCBSENV) is created.

CICS will create a TASK Level ACEE when a user issues a CICS LOGON, and will delete the TASK Level ACEE at CICS LOGOFF.

FINDING THE ACTIVE ACEE.

It's normal programming practice to check the TCB first to find the current ACEE, and then look for an ACEE pointer in the ASXB. In an address space containing multiple tasks, where each task represents a different user, an ACEE can be built for each task. The MVS ATTACH process doesn't propagate the TCBSENV field. The new sub-task lives under the ASXBSENV

```
        L     R15,PSATOLD-PSA     Address of the current TCB
        USING TCB,R15             Establish Addressability to the TCB
        ICM   R14,B'1111',TCBSENV Pointer to RACF ACEE?
        BNZ   ADDRACEE            Y,
        L     R15,PSAAOLD-PSA     Address of our ASCB
        USING ASCB,R15            Establish Addressability to the ASCB
        L     R15,ASCBASXB        @ of the ASXB
        USING ASXB,R15            Establish Addressability to the ASXB
        ICM   R14,B'1111',ASXBSENVPointer to RACF ACEE?
        BZ    NOACEE              N, Invalid RACF Environment
ADDRACEEDS    ØH
        USING ACEE,R14            Establish Addressability to the ACEE
        ……..

NOACEE  DS    ØH
        …….
*_____________________________________________________________*
*    Mapping Macros                                           *
*_____________________________________________________________*
        IHAPSA
        IHAASCB
        IHAASXB
        IHAACEE
        IKJTCB
```

Figure 5: Example code for obtaining addressability to the ACEE

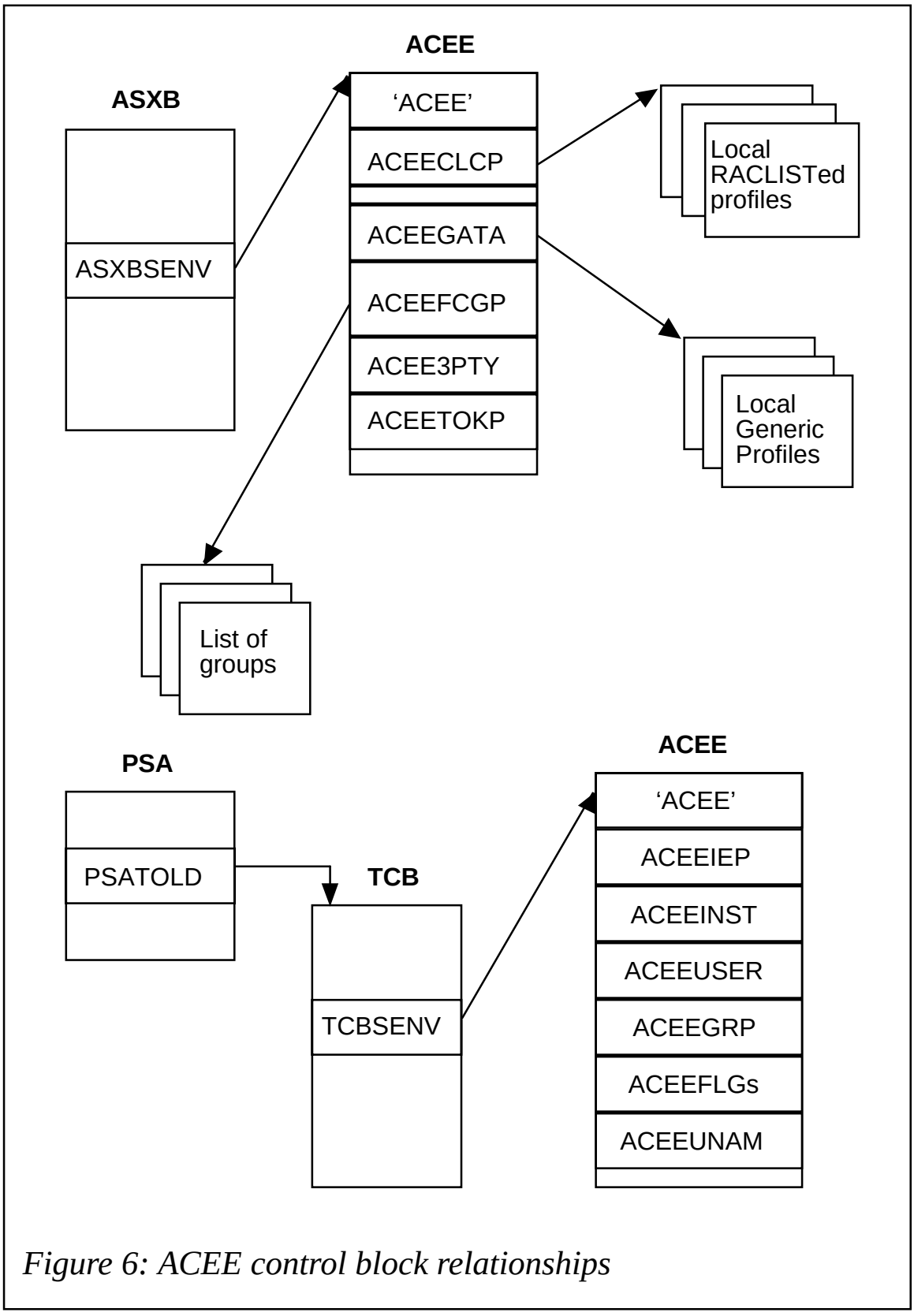

Figure 6: ACEE control block relationships

ACEE. Figure 5 shows the coding to obtain addressability to the ACEE.

THIRD-PARTY RACHECK.

A 'third-party RACHECK' is when you need to do an authorization check for a user who isn't the current user in the ACEE. When this call is made, RACF builds an ACEE for the other user specified on the macro call. The ACEE that's built can be used for many users. The ACEE that's created is chained off the caller's ACEE in field ACEE3PTY. For a caller to issue a third-party RACHECK, they must be authorized, key 0-7, or in supervisor state.

Figure 6 shows the ACEE control block relationships.

CONNECT GROUP NAME TABLE.

The connect group name table (CGRP) contains the names of the groups of which a userid (ACEEUSRI) is a member. The CGRP is built in sub-pool 255 (LSQA/ELSQA) key 0 storage, and is pointed to by field ACEECGRP or ACEEFCGP of the ACEE data area. The CGRP is mapped by the ICHPCGRP macro.

SYSTEM AUTHORIZATION FACILITY

System Authorization Facility (SAF) is part of the MVS operating system. It was first introduced in MVS/ESA 3.1.3 and RACF Release 1.9. In order for SAF to provide a security environment for address spaces before RACF initialization it needs to be initialized early in NIP processing. SAF initialization is a Resource Initialization Module (RIM) that is invoked after the LPA is built. This is known as SAF early initialization. Because many of the services that may be used by the SAF user exit are not available this early in NIP processing, a new user exit, ICHRTX01 is loaded and used as the SAF user exit from this point until the Master Scheduler Initialization replaces it with the address of the original SAF user exit, ICHRTX00.

SAF functions include:

- Providing a security environment
- Creating and maintaining security tokens
- If RACF is not active, SAF builds a default ACEE and UTOKEN to satisfy the security request.

SAF routing concept

The SAF interface is entered by issuing the RACROUTE macro from an MVS Resource Manager. In theory, Resource Managers are responsible for calling the SAF interface to determine whether to allow a user access to the system, dataset, or a resource. There are a number of resource managers in MVS, including:

- JES2

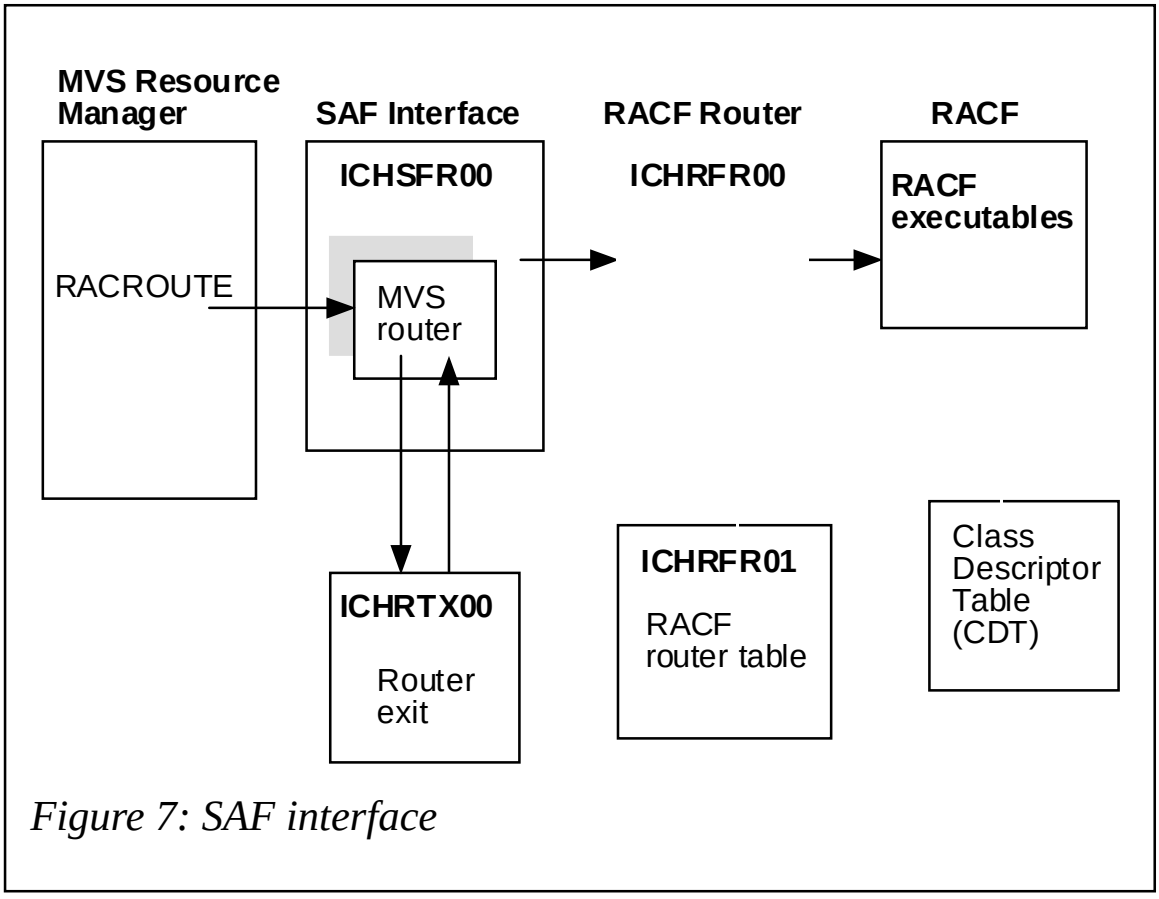

Figure 7: SAF interface

- CICS
- DFHSM
- DFP
- DFSMS
- DASDM
- DB2.

The RACROUTE macro invokes the MVS Router, ICHSFR00, which in turn invokes the router exit ICHRTX00. ICHRTX00 will then pass a return code back to the MVS Router (ICHSFR00) which will indicate whether security processing is to occur. If RACF is active and a return code of zero was returned by ICHRTX00, the MVS router will then pass control to the RACF router, ICHRFR00. The RACF router will then invoke RACF.

Figure 7 illustrates how the MVS Resource Managers invoke the SAF interface.

SAF router exit

ICHRTX00 is the SAF router exit and is part of MVS, not RACF. This exit is taken every time RACF is invoked by the RACROUTE macro. Information is passed to RACF through the RACROUTE parameter list. On return from ICHRTX00, a return code indicates whether security processing is to occur.

The ICHRTX00MVS Router exit isn't invoked for all jobs that enter the system. For example, the exit is not invoked for TSO log-ons or started procedures.

CUSTOMIZING RACF MODULES.

There are a number of RACF modules that an installation can customize:

- ICHSECOP
- ICHRDSNT

- ICHRRNG
- ICHRRCDE
- ICHRFR01
- ICHNCV00
- ICHRIN03
- ICHAUTAB.

ICHSECOP

ICHSECOP can be used to:

- Bypass RACF initialization at IPL.
- Select the number of resident data blocks if the dataset name table is not used.
- Disallow duplicate dataset names.

This option is not recommended.

ICHRDSNT

The database name table (ICHRDSNT) module provides the following options:

- Define the names of the RACF databases.
- Define the RACF database duplexing options (updates and statistics).
- Define the number of resident data blocks.
- Define the use of RACF sysplex communication.

ICHRRNG

The database range table (ICHRRNG) which must be kept in sync with the database name table can be used to:

- Physically split one or more RACF databases across multiple datasets.

- Allow the distribution of RACF I/O overhead across multiple DASD drives.

ICHRRCDE

The installation-defined class descriptor table (ICHRRCDE) defines installation class entries or OEM vendor classes. RACF is – with the exception of user, group, and dataset profiles – strictly table-driven. For each general resource class, there's a unique entry in the table. RACF references the CDT whenever it receives a resource class name other than DATASET, USER, or GROUP. The IBM classes are provided in ICHRRCDX. At IPL, RACF merges the ICHRRCDX and ICHRRCDE tables. If a class is added to the installation class-descriptor table, a corresponding entry should be added to the RACF router table (ICHRFR01).

ICHRFR01

The RACF router table contains installation- and OEM-defined classes to the RACF Router. It is used by the RACF portion of the SAF router (see Figure 7 above). The entries should match the entries contained in ICHRRCDE. ICHRFR01 is merged with the IBM-supplied Router table (ICHRFR0X).

ICHNCV00

The RACF naming conventions table (ICHNCV00) is used to rearrange dataset names, to reference different profiles based on any combination of:

- Dataset names (or parts thereof)
- Userids (or parts thereof)
- Current group (or part thereof).

ICHRIN03

The started procedures table (ICHRIN03) assigns a user and group to a started task. There is a single entry per STC, with one

default entry. If changes are required to ICHRIN03 they will not take effect until the next IPL. An alternative to ICHRIN03 is the RACF STARTED class. The biggest benefit of using the RACF STARTED class is that any changes that are made will take effect immediately.

ICHAUTAB

The RACF authorized caller table (ICHAUTAB) contains the names of programs that an installation authorizes to issue the RACROUTE REQUEST=LIST or RACROUTE REQUEST=VERIFY without the NEWPASS keyword.

R F Perretta
Millenium Computer Consultancy (UK)

E-mail alerts

Our e-mail alert service will notify you when new issues of *RACF Update* have been placed on our Web site. If you'd like to sign up, go to http://www.xephon.com/racf and click the 'Receive an e-mail alert' link.

RACF availability

With the marked increase in emphasis on security over the last year and a half, proactive approaches are gaining popularity. Whether you're the resident RACF expert or a consultant just arriving in an organization, you probably need an independent (ie, more reliable than just asking someone) method of determining whether RACF is installed and running. And, if so, what version.

There are several programming solutions to this problem.

REXX has a TSO/E external function called SYSVAR that returns various types of system information. SYSVAR works only in TSO/E address spaces, so REXX must be run in TSO. In batch, this means executing IKJEFT01 (TSO), and coding a TSO EXEC command to execute the REXX EXEC from within TSO, as in the following example:

```
//RACFSTAT EXEC PGM=IKJEFTØ1
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
EXEC 'JONPE.ACTIVE.EXEC(RACFSTAT)'
//
```

SYSVAR has two arguments that return information on RACF. SYSVAR('syslracf') returns Null if RACF is not installed, and a four-digit number if RACF is installed. The number is the numeric part of the FMID of the release of RACF. But since RACF lost its separate identity as a product beginning with OS/390 Version 1.3, I've listed below the values from RACF 2.2 onwards:

- 2220 – RACF 2.2, OS/390 1.1 or 1.2
- 2230 – OS/390 1.3
- 2240 – OS/390 2.4
- 2260 – OS/390 2.6
- 2280 – OS/390 2.8
- 7703 – OS/390 2.10
- 7705 – z/OS 1.2
- 7706 – z/OS 1.3
- 7707 – z/OS 1.4.

SYSVAR('sysracf') returns one of three values, indicating the current state of RACF:

- NOT INSTALLED

- NOT AVAILABLE
- AVAILABLE.

REXX PROGRAM

If you turn this into a REXX program, it might look something like this:

```
/* REXX */
status = sysvar("sysracf")
if status = "AVAILABLE" then do
   exitrc = Ø
   end
else do
   exitrc = 8
   end
say "RACF is" status
fmid = sysvar("syslracf")
if fmid <> "" then do
   say "RACF Version/FMID is" fmid
   end
exit exitrc
```

Run as a batch job, under TSO (IKJEFT01), variable exitrc is used to set the z/OS Batch COND CODE to 8 if RACF is not available, zero otherwise. Automated Operations systems can use this to trigger appropriate action – or you can perform your own, as in this example batch job where you e-mail yourself:

```
//* CHECK THE STATUS OF RACF.
//* FOR ANY PROBLEMS, E-MAIL DETAILS
//*
//* THIS JCL FILE MUST BE UNNUMBERED
//RACFSTAT EXEC PGM=IKJEFTØ1         TSO
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD DSN=&&REXX,SPACE=(TRK,(1,5),RLSE),
//            RECFM=FB,LRECL=8Ø,
//            DISP=(NEW,PASS)
//SYSTSIN  DD *
EXEC 'JONPE.ACTIVE.EXEC(RACFSTAT)'
//*
//         IF (RACFSTAT.RC > Ø |
//             RACFSTAT.ABEND = TRUE |
//             RACFSTAT.RUN = FALSE) THEN
//EMAIL   EXEC  PGM=ICEMAN
//SYSOUT   DD    SYSOUT=*
```

```
//SYSIN   DD  *
  SORT FIELDS=COPY
//SORTIN   DD    *
HELO S390
MAIL FROM:<JON.PEARKINS@ADIANT.COM>
RCPT TO:<JON.PEARKINS@ADIANT.COM>
DATA
//         DD DSN=&&REXX,DISP=(OLD,PASS)
//SORTOUT  DD SYSOUT=(B,SMTP)
//         ENDIF
//*
//         IF (RACFSTAT.RUN = TRUE) THEN
//PRINT   EXEC PGM=ICEMAN
//SYSOUT   DD  SYSOUT=*
//SORTIN   DD  DSN=&&REXX,DISP=(OLD,DELETE,DELETE)
//SORTOUT  DD  SYSOUT=*
//SYSIN   DD  *
  SORT FIELDS=COPY
//         ENDIF
//
```

The most important point to note is that the JCL must not contain sequence numbers. The ISPF Editor's UNNUM command is the simplest way to be sure that there are no sequence numbers present. In fact, the one place where sequence numbers will cause problems is the e-mail message lines beginning HELO and ending DATA.

The first JCL IF statement checks for any of three conditions being true about the first job step, where the REXX code was run:

- Non-zero COND CODE (RC)
- Job step abended
- Job step not run.

Any of these conditions will trigger an e-mail message that will contain the output from the REXX program, stored in the &&REXX temporary dataset. The e-mail arrives with no Subject or To field, and your e-mail address as the From field.

The next job step takes the same data from &&REXX and sends it to SYSOUT for anyone looking at the output of the batch job.

Normal output for &&REXX for OS/390 Version 2.10 would be as follows:

```
READY
EXEC 'JONPE.ACTIVE.EXEC(RACFSTAT)'
RACF is AVAILABLE
RACF Version/FMID is 77Ø3
READY
END
```

Although it's beyond the scope of this article to explain how to add Subject and To fields to the e-mail, the HELO S390 deserves some attention. S390 is the name of the local host that handles the outgoing e-mail. I found it in the system dataset TCPIP.HOSTS.LOCAL, in one of three uncommented lines:

```
HOST : 197.Ø.Ø.1 : localhost ::::
HOST : 219.2Ø7.253.168 : s39Ø, s39Ø.adiant.com ::::
HOST : 253.288.147.112 : gollo2ØØØ ::::
```

CLIST

CLIST is quite similar to REXX, in that the CLIST Control Variables have the same names and return the same values as the REXX SYSVAR parameters. &SYSRACF has three possible values:

- NOT INSTALLED
- NOT AVAILABLE
- AVAILABLE.

&SYSLRACF is null if RACF is not installed, and the four-digit numeric portion of the FMID if RACF is installed. Possible values are explained above, in the REXX section.

The CLIST equivalent of the RACFSTAT REXX program would be as follows:

```
SET &STATUS = &SYSRACF
IF &STATUS = AVAILABLE THEN DO
   SET &EXITRC = Ø
   END
ELSE DO
   SET &EXITRC = 8
```

```
   END
WRITE RACF IS &STATUS
SET &FMID = &SYSLRACF
IF &FMID NE THEN DO
   WRITE RACF VERSION/FMID IS &FMID
   END
EXIT CODE(&EXITRC)
```

And the JCL to run this CLIST is almost the same as for REXX. The only change is the TSO command in the first job step, where the Low Level Qualifier (LLQ) changes to CLIST, as shown here:

```
EXEC 'JONPE.ACTIVE.CLIST(RACFSTAT)'
```

Although, technically, you can mix CLIST and REXX in a single PDS library, doing this can lead to a lot of confusion. Especially considering the decision of what LLQ to use, CLIST or EXEC, since each defines what you expect to find in each: CLISTs or REXX EXECs.

TRADITIONAL PROGRAMMING LANGUAGES

But what if a program needs to know whether RACF is available – proactive programming may mean checking to be sure that security is in place before doing anything 'sensitive'.

Neither COBOL nor CICS offers any direct way to check on RACF. However, you can use the QUERY SECURITY function in CICS to check a specific RACF resource. An INVREQ will occur, with a RESP2 value of 10, if neither RACF nor a competitive product is present and active. CICS manuals refer to RACF and its competitors as an External Security Manager (ESM). For more information on QUERY SECURITY, see:

- *CICS Transaction Server for z/OS CICS Application Programming Reference* (SC34-5994).
- *CICS Transaction Server for z/OS RACF Security Guide* (SC34-6011).

Although normally used to check a specific RACF resource, the RACROUTE REQUEST=STAT macro does provide some

information on the current status of RACF when no resource is specified. But, as the word Macro suggests, RACROUTE is available only in Assembler. Of course, COBOL, PL/I, and other languages can call Assembler subroutines, if the Assembler is written with standard high-level language calling conventions in mind.

ASSEMBLER

Here is how you might code an Assembler routine to make a STAT request to RACROUTE and return the results – SAF return code, RACF return code, and RACF reason code – to the calling program:

```
*  RACFSTAT ROUTINE
*  DETERMINE IF RACF IS INSTALLED AND WORKING.
*  CALLING PROGRAM PROVIDES THREE FULL WORD (4 BYTE) BINARY
*  VARIABLES AS PARAMETERS THAT WILL BE RETURNED BY RACFSTAT
*  WITH VALUES:
*  CALL RACFSTAT(SAF-RETURN-CODE,RACF-RETURN-CODE,RACF-REASON-CODE)
RØ       EQU   Ø                  REGISTERS USED IN ROUTINE
R1       EQU   1
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
PARMLIST DSECT ,                  CALLING PROGRAM'S PARAMETER LIST
ASAFRC   DC    A(PSAFRC)          R1 -> PARMLIST
ARACFRET DC    A(PRACFRET)
ARACFRES DC    A(PRACFRES)
PARMS    DSECT ,                  PARAMETER VALUES RETURNED TO CALLER
PSAFRC   DS    F                  SAF RETURN CODE
PRACFRET DS    F                  RACF RETURN CODE
PRACFRES DS    F                  RACF REASON CODE
RACFRC   DSECT ,                  FIRST 2 BYTES OF RACROUTE MF=L
RACFRETC DS    FL1                RACF RETURN CODE ONE BYTE BINARY
RACFREAS DS    FL1                RACF REASON CODE ONE BYTE BINARY
RACFSTAT CSECT ,                  RACFSTAT – FINDS RACF STATUS
*  STANDARD PROLOGUE
         STM   R14,R12,12(R13)    SAVE CALLER'S REGISTERS
```

```
         LR    R12,R15            ESTABLISH ADDRESSABILITY USING R12
         USING RACFSTAT,R12       AS BASE REG. FROM R15 ENTRY POINT.
         ST    R13,SAVEAREA+4     OLD SAVEAREA POINTER IN NEW SA
         MVC   8(4,R13),=A(SAVEAREA)    NEW SA POINTER IN OLD SA
         LA    R13,SAVEAREA       START USING NEW SAVE AREA
*  END PROLOGUE
*  REGISTERS USED IN ROUTINE:
BICHSAFP EQU   R11
RACFRCRC EQU   R3
BSAFRC   EQU   R4
BRACFRET EQU   R5
BRACFRES EQU   R6
WORKREG  EQU   R7
*  ACTUAL (USEFUL) CODE BEGINS HERE:
         ST    R1,PARMADDR        SAVE CALLER'S PARAMETER LIST
         RACROUTE REQUEST=STAT,                                       X
               RELEASE=(1.9,CHECK),  RACF VERSION 1.9 IS THE FIRST    X
               WORKA=RRWORKA,        SUPPORTING STAT                  X
               DECOUPL=YES,                                           X
               MF=(E,LICHSAFP)
         LA    BICHSAFP,LICHSAFP  RACROUTE MF=L AREA
         USING RACFRC,BICHSAFP    RACF RETURN/REASON CODE IS THERE
         L     R1,PARMADDR        RESTORE CALLER'S PARAMETER LIST
         USING PARMLIST,R1
         L     BSAFRC,ASAFRC
         USING PSAFRC,BSAFRC
*  RETURN SAF RETURN CODE, RETURNED BY RACROUTE STAT MACRO CALL,
*  TO CALLING PROGRAM VIA FIRST PARAMETER
         ST    R15,PSAFRC
         DROP  BSAFRC
         L     BRACFRET,ARACFRET
         USING PRACFRET,BRACFRET
         SR    WORKREG,WORKREG    CLEAR REGISTER FOR LOADING 1 BYTE
         IC    WORKREG,RACFRETC   RACF RETURN CODE FROM RACROUTE MF=L
         ST    WORKREG,PRACFRET   RETURN RACF RETURN CODE TO CALLER
         DROP  BRACFRET
         L     BRACFRES,ARACFRES
         USING PRACFRES,BRACFRES
         IC    WORKREG,RACFREAS   RACF REASON CODE FROM RACROUTE MF=L
         ST    WORKREG,PRACFRES   RETURN RACF REASON CODE TO CALLER
         DROP  BRACFRES
*  RESTORE/SET REGISTERS
         L     R13,4(R13)         CALLER'S SAVE AREA
         L     R14,12(R13)        RESTORE R14 (CALLER RETURN ADDRESS)
*  R15 HAS BEEN PRESET BY RACROUTE WITH SAF RETURN CODE
         LM    RØ,R12,2Ø(R13)     RESTORE REGISTERS AS CALLER LEFT THEM
         BR    R14                RETURN TO CALLING PROGRAM
*
SAVEAREA DS    18F                THIS ROUTINE'S SAVE AREA
PARMADDR DS    A                  SAVE THE CALLER'S REGISTER 1
```

```
RRWORKA  DS    CL512              RACROUTE WORK AREA (REQUIRED)
LICHSAFP RACROUTE REQUEST=STAT,   STORAGE AREA OF RACROUTE STAT       X
               RELEASE=1.9,                                           X
               MF=L
         LTORG
         END
```

The SAF return code is returned as a standard return code, in Register 15, by RACROUTE. The RACF return and reason codes are returned in the first two bytes of the RACROUTE storage area, as one-byte binary values. To label and access the storage area, the Execute (MF=E) and List (MF=L) forms of RACROUTE must be used, rather than the simpler, but non-re-entrant, Standard form (MF=S).

As well as returning the SAF and RACF return/reason codes as parameters, the SAF return code is also used to set the return code (in Register 15) from the RACFSTAT Assembler routine. The RACROUTE STAT request first became available in RACF Version 1.9. RELEASE=1.9 was coded because the RACROUTE macros default to Version 1.6.

COBOL

A simple COBOL program to call RACFSTAT could be coded as follows:

```
ID DIVISION.
PROGRAM-ID. CLRACFST.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
DATA DIVISION.
FILE SECTION.
WORKING-STORAGE SECTION.
Ø1  RACFSTAT-PARMS.
Ø5  SAF-RETURN-CODE PIC S9(9) USAGE BINARY VALUE –1.
Ø5  RACF-RETURN-CODE PIC S9(9) USAGE BINARY VALUE –1.
Ø5  RACF-REASON-CODE PIC S9(9) USAGE BINARY VALUE –1.
PROCEDURE DIVISION.
CALL 'RACFSTAT' USING SAF-RETURN-CODE,
RACF-RETURN-CODE,
RACF-REASON-CODE.
DISPLAY RETURN-CODE.
DISPLAY SAF-RETURN-CODE,
RACF-RETURN-CODE,
```



```
RACF-REASON-CODE.

STOP RUN.
```

The DISPLAY statements output to the DD name SYSOUT. The JCL to run it would look as follows:

```
//CLRACFST EXEC PGM=CLRACFST
//STEPLIB  DD  DSN=JONPE.ACTIVE.LOAD,DISP=SHR
//         DD  DSN=CEE.SCEERUN,DISP=SHR
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//CEEDUMP  DD  SYSOUT=*
//
```

RETURN AND REASON CODES

What are the possible values of the SAF Return Code and RACF Return and Reason Codes? The *z/OS Security Server RACROUTE Macro Reference* (SA22-7692) manual lists a large number of combinations in the description of the standard form of the RACROUTE STAT request. But only a few apply to requests without a RACF resource specified.

All three SAF and RACF values are zero if RACF is active. The SAF Return Code is 64 if the RELEASE= values, either specified or defaulted, do not match between the Execute (MF=E) and List (MF=L) forms of the RACROUTE macro. But only if the CHECK parameter is specified in the Execute form:

```
RELEASE=(1.9,CHECK)
```

Note that this is what the manual says. In practice, I was unable to trigger the SAF Return Code of 64 when I let the List form of RACROUTE default while specifying 1.9,CHECK in the Execute form.

If the SAF Return Code is four, there are three possibilities:

- RACF Return and Reason Codes are zero – the RACF Router was not loaded.
- RACF Return Code is 18 (hexadecimal) – RACF is either not installed, or an old version of RACF – before Version 1.9

– is installed.

- RACF Return Code is 1C – the RACROUTE parameter list is the wrong length.

Jon E Pearkins
(Canada)

Setting up security for JES2

PROBLEM

In an attempt to set up security for JES2, for a remote system, I issued the following commands:

```
SETROPTS GENERIC(STARTED)

RDEFINE STARTED JES2.* STDATA(USER(JES2) GROUP(STCGRP) TRUSTED(YES))

RDEFINE STARTED ** STDATA(USER(MEMBER) GROUP(STCGRP) TRACE(YES))

SETROPTS CLASSACT(STARTED)

SETROPTS RACFLIST(STARTED)
```

I didn't define a profile for the started classes. I re-IPL'd the system, and encountered the following problems:

- I couldn't bring up the system properly as RACF had the started class activated.
- JES2 was not coming up at all.
- All the started tasks were abending because the generic ** was activated and JES2 was not available.

SOLUTION

We turned RACF Inactive by going to the Console to alter the memory. On asking for help, we found that we needed to use the Flex-ES command level interface to alter the memory. We

pointed the Communication Vector Table (CVT) to RCVT, and from the RCVT manipulated the address.

Our first task was to figure out the memory and understand it. We found that at address x'10' in storage in the CVT, adding x'3E0' to the address enables you to see the eyecatcher RCVT.

We used

```
D RMEM 10
```

to point to the memory location x'10'.

This produced the following result:

```
00000010: 0FE8C010   00FD7FF0 00000000 070C0000 80FEE140
```

Note that the address is FD7FF0.

We did a display to see what it contained, by issuing the following command

```
D lmem fd7ff0
```

The contents were as follows:

```
00FD7FF0:  00000218 00FE9D80 00FCF23C 00FD2C88   *.........2....h*
```

For this address 00FE9D80**,** we added x'3E0', and obtained the following result:

```
FD83D0
```

This address pointed to some further address in memory. To see what that address contained, we issued the following:

```
D lmem FD83D0
```

This produced the following:

```
00FD83D0:0F77C3D0 00F9B6F8 00FE7278 00FD7F6C 00FE29C8 *.9.8.....r%...H*
```

To look at the contents of address 00F9B6F8, we issued the following:

```
D lmem F9B6F8
```

This gave the following result:

```
00F9B6F0:02F776F0 40404040 40400000 D9C3E5E3 00C00028 *      ..RCVT.{..*
```

On the right you can see RCVT the eyecatcher.

We therefore issued the following command again, for 100 addresses:

```
d lmem f9b6f8 100
```

The actual contents available in RCVT were as follows:

```
00F9B6F0:02F776F0 40404040 40400000 D9C3E5E3 00C00028 *      ..RCVT.{..*
00F9B700:02F77700 00C00080 0598B380 00000000 00000000 *.{...q.........*
00F9B710:02F77710 00000000 00000000 00000000 00F3C418 *.............3D.*
00F9B720:02F77720 00102C80 00C000B0 00000000 093E000A *.....{.........*
00F9B730:02F77730 E2E8E2F1 4BD9C1C3 C6404040 40404040 *SYS1.RACF       *
00F9B740:02F77740 40404040 40404040 40404040 40404040 *                *
00F9B750:02F77750 40404040 40404040 40404040 E2E8E2F1 *            SYS1*
00F9B760:02F77760 4BE4C1C4 E2404040 40404040 40404040 *.UADS            *
00F9B770:02F77770 40404040 40404040 40404040 40404040 *                *
00F9B780:02F77780 40404040 40404040 E2C3D7D4 E5F53500 *        SCPMV5..*
00F9B790:02F77790 0003811E 80C370EC 00000000 00000000 *..a..C.........*
00F9B7A0:02F777A0 00C4E000 08000000 00000000 00000000 *.D\............*
00F9B7B0:02F777B0 00000000 00F98DA0 00C43000 00DF9000 *.....9...D......*
00F9B7C0:02F777C0 00000000 00000000 00000000 80C1DEB0 *............A..*
00F9B7D0:02F777D0 00000000 00000000 05A9B530 00C20000 *.........z...B..*
00F9B7E0:02F777E0 00D41150 00000000 0A05191E 0508C1C1 *.M.&.........AA*
00F9B7F0:02F777F0 C1C1C1C1 C1C106080 C1C1C1C1 C1C1D5D5 *AAAAAA..AAAAAANN*
```

The best way to understand this is to find the VOLID of UADS dataset or zero. This information can be found at the following address:

```
00F9B780:02F77780 40404040 40404040 E2C3D7D4 E5F53500 *        SCPMV5..*
```

For this address we have to add x’35’ to make RACF inactive**.**

The information in Figure 1 is taken from the RCVT table for x’35’.

Our next task was to stop the processor to alter the storage. To do this, we issued the following command:

```
H or stop
```

To alter the storage, we issued the following command:

```
A lmem FBF2C5  FE
```

The reason for doing this was that earlier it contained 03. We

Offsets			Len		
Dec	**Hex**	**Type**		**Name (Dim)**	**Description**
53	(35)	BITSTRING	1	RCVTSTAT	STATUS
		1...		RCVTRNA	RACF NOT ACTIVE
		.1..		RCVTNLS	BYPASS RACINIT STATISTICS
		..1.		RCVTNDSS	BYPASS DATASET STATISTICS
		...1		RCVTNTVS	NO TAPE VOLUME STATISTICS
		.... 1...		RCVTNDVS	NO DIRECT ACCESS VOLUME STATISTICS
		.... .1..		RCVTNTMS	NO TERMINAL STATISTICS
		.... ..1.		RCVTNADS	NO ADSP PROTECTION
		1		RCVTEGN	EGN SUPPORT IN EFFECT

Figure 1: Information from the RCVT table for x'35'

therefore had to change the content to make RACF inactive.

The next job was to restart the processor. To do this, we issued the following command:

```
G or start
```

Next, in the master console, we issued the start command for JES2, and JES2 started again.

We had to reply to around 40 messages like the following:

```
*nn ICH802D REPLY Y OR N TO THE REQUEST.

IEE600I REPLY TO 30 IS;SUPPRESSED

ICH801I 'JES2'  ATTEMPTING 'READ' ACCESS of ENTITY 'SYS1.PROCLIB'
```

Offsets			Len		
Dec	Hex	Type		Name (Dim)	Description
152	(98)	BITSTRING	1	RCVTAXTA	RESERVED

Figure 2: Information from the RCVT table for x'98'

After JES2 came back up, we experienced problems bringing up TSO, and decided to change the RCVT + '98' back to its original value (0003811E).

The information in Figure 2 is from the RCVT table for x'98'.

TSO now came up and we issued the following commands:

```
SETR NORACLIST(STARTED)

SETR NOCLASSACT(STARTED)
```

We re-IPL'd the system, and everything came back up just fine.

Richard Daniel Gunjal
Systems Programmer, GlobalSoft
(Singapore)

RACF – your questions answered

DISCRETE DATASET PROFILES

Dear Doc

How can I prevent users from creating discrete dataset profiles?

Signed

Frustrated in Fort Wayne

Dear Frustrated

I don't think you can 'prevent' that from happening. However, you can control it. First, understand that only a few TSO users have the ability to do this, and they're most likely your tech support staff. As I've mentioned in some of my previous articles, you shouldn't allow discrete dataset profiles, because anyone with ALTER access on them can actually assign security levels. Your best bet is to regularly scan your RACF database for this by using Pentland Utility RACF80. And when you get any hits on that report, contact the person who created the profile and let them know that profiles should be created only by the Security Department.

EXPERIENCE FOR RACF ADMINISTRATORS

Dear Doc

I've got some trainee RACF administrators, and I don't want them to have SPECIAL access yet. However, I do want them to get some experience with things like resetting passwords. Any suggestions?

Signed

Worried in Walla-Walla Washington

Dear Worried

You'll need to activate a set of little-known RACF general

resource profiles called IRR.PASSWORD.RESET and IRR.LISTUSER (within FACILITY). Define the GenRes profile and link your trainees to it. This will give them inquiry functions in RACF and allow them to reset passwords, but nothing else. For full details, check out the *RACF Security Administrator's Guide* (SC28-1915-06).

ICHPWX01 OVERHEAD

Dear Doc

What is the operational/overhead impact of ICHPWX01 on an IBM ES/9672-RC6 with 1664MB of main memory, single LPAR, with approximately 2,500 users?

Signed

Techie in Tamworth

Dear Techie

ICHPWX01 (the password filter) would have little, if any, impact on a medium- to large-scale mainframe. The real question here, though, is how much data is included in the program? If you use the words already in the standard code, you've got no problem. If, on the other hand, you load it with Webster's Unabridged Dictionary, search times will increase significantly. I don't think anyone has done actual timings on ICHPWX01 in a live environment, so there is no 'specific' answer to that question. But even if it adds a full second to the log-on process, if your data is valuable enough, it's well worth the time.

SPECIAL AND AUDITOR FOR SECURITY ADMINISTRATION TEAM

Dear Doc

Our auditors have reported that the security administration team should have access to SPECIAL only, and not SPECIAL and AUDITOR. They say it defeats the segregation of duties, but it cuts down on some of our reporting capabilities. Are the auditors right?

Signed

Miffed in Milwaukee

Dear Miffed

No, your auditors aren't right. It sounds like they're doing what's called a 'cookie-cutter' audit – asking questions, getting answers, but not analysing the actual risks involved. AUDITOR capability only permits the generation of certain special reports, as well as assigning UAUDIT to user profiles. I don't think that's outside the normal duties of a security administrator. If they give you any static on this issue, make them outline the specific, quantifiable risks of the SPECIAL/AUDITOR combination. Chances are, they won't have any.

SECURING THE CEMT COMMAND

Dear Doc

Is there a way to provide greater security and control over the CEMT command, beyond just restricting access to the transaction itself? Some of our Technical Support team need access for maintenance purposes, but there are some programmers asking for it as well to do inquiries. What should I do?

Signed

Control Conscious in Kalamazoo

Dear Control Conscious

This is where CICS security control gets kind of awkward in RACF. You can further segregate CEMT functionality in the VCICSCMD/CCICSCMD general resource profiles. However, it's a very difficult and time-consuming analysis to determine the specifics of the segregation.

I worked on an analysis of this many years ago, but I don't have the documentation on that any more. If any of our other readers can shed some light on this issue, I'll be sure to include it in a

future column.

MAINFRAME VIRUSES

Dear Doc

Is there such a thing as a 'mainframe virus'?

Signed

Paranoid in Peoria

Dear Paranoid

Actually, yes. I've only ever heard of two or three, however, and they're only 'lab' viruses. They're pretty old now, and I'm not sure if they would function in an OS/390 or z/OS environment (backward compatibility notwithstanding). Also, since mainframes are generally not networked as PCs are, viral propagation is severely limited. It would take someone from inside an organization to introduce the code into your system, and they would probably have to have a high level of system authority to even run it.

However, you do bring up one point that is worthy of note. Even though a mainframe cannot be directly affected (or infected) by the plethora of viruses 'in the wild' for PCs, they can still be a conduit or carrier for them. For example, if your mainframe is connected to the Internet, or is used to store/distribute PC files between sites, the mainframe could be used as a distribution point. That's why it's very important to ensure you've got adequate firewalls and virus detection mechanism between your mainframe(s) and the Web.

FUTURE COLUMNS

Dear Readers,

In case you haven't already guessed, the questions in this article were from my own fevered imagination. This is the third 'Your questions answered' column, and I've still received no questions, comments, or suggestions from you. If this trend

continues, I'll have no alternative but to include the following subject matter in future columns:

- My world-famous lasagne recipe.
- Long, boring, detailed specifications for my new standards of time measurement (clock and calendar).
- Verbatim transcripts of all my favourite comedy sketches.
- My poem, 'An Ode to Baldness'.

I don't think any of us really want that, now, do we?

Please feel free to make suggestions or pose questions of your own (or else!). That's what this column is here for. You can contact me at DocFarmer@qatar.net.qa

Doc Farmer
Security and business continuity expert
(Middle East)

August 1995 – February 2003 index

Items below are references to articles that have appeared in *RACF Update* since August 1995. References show the issue number followed by the page number(s). Back issues of *RACF Update* can be ordered from Xephon. See page 2 for details.

Access checking 1.50-52, 26.3-11, 31.3-10
Access list 18.52-59
Access level 30.42-58
ACCHECK 1.50-52
ACEE 15.28-34
ACF2 26.32-35, 28.51-58
ADDGROUP 1.22-27
ADDUSER 1.22-27
ADMIN 5.3-10, 3.40-49
Administration 2.3-17, 2.50-63, 3.40-49, 5.3-10, 7.16-25, 9.11-42, 28.3-20
Aliases 7.53-59
Amending user-ids 3.40-49
Announcements 26.21-22, 27.22-32, 28.21-34, 29.37-43
APF dataset checker 6.33-60, 10.20-22
Application Interface link 3.8-11
AUDIT 5.41-61, 30.22-26, 30.60-61
Auditing with SAS 5.41-61
Auditor access 31.56-57
Authentication 20.6-10
Authorization 3.29-36, 18.28-51, 19.12-38, 21.37-48
Authorized functions 22.3-27
Automation 9.11-42
Availability 31.41-50
Belkin 28.59-64
BLKUPD 3.37-40
CANCELID 5.34-41
CAPTURE 5.11-17
Catalog alias 17.24-34, 18.17-27
CATEGORY 11.3-6
CDSPNLO1 1.3-10
CEMT 31.57
CEXDRACF 7.53-59
Checking access 4.28-37
Checking resources 7.53-59
CICS 3.8-11, 7.16-25, 29.57-58
CICSDISP 3.40-49
Class masks 7.3-15
Cloning 14.50-59
CMSK 7.25-52
CNST 9.45-59
CNSX 9.45-59
COBOL transaction 23.22-27
Company list 23.57-63
Consulting RACF 27.33-46
Control blocks 31.29-40
Cryptography 11.36-59, 12.2-26
CSVAPF macro 10.20-22
CVH macro 1.10-21
Database 29.3-12
Database name table 23.3-21
Dataset profile 1.22-47, 31.55
Dataset protection 21.49-51
DDoS 25.38-43, 26.36-40
Decentralized RACF administration 28.3-20
Decryption of datasets 11.36-59, 12.6-26
Deleting profiles 5.34-41
Deleting user-id 3.37-40
DFHRMM 19.39-59
DFHSM 1.48-49, 19.39-59
DFSMS 20.11.18
Discrete profiles 31.55
Displaying system data 3.3-8
DSET 9.11-42
DSETMAIN 9.11-42
DSMON 30.15-21
DSN 30.42-58
Dynamic access 20.3-5
Enhancements 26.21-22
Encryption of datasets 11.36-59, 12.6-26
Extracting user information 3.52-63
Firewall 25.44-55, 27.47-53, 27.54-58, 28.59-64
Front-end module 1.22-27

Front-end reset 9.3-11
FTCHKPWD 8.11-14
FTP 26.23-31
FTP security exit 8.11-14
GCICS 7.16-25
Generation datasets 26.3-11
GID 25.10-37, 30.61-62
Glossary 23.28-56
Group commands 11.36-59
Groups' accesses 5.11-17, 27.3-7
GTF 16.59
Handling RJE jobs 2.17-21
Hardware protection 8.3-11
HASP 29.57-58
Help desk facilities 4.40-62
ICHNCVOO 4.37-39
ICHPWX01 exit 6.3-11, 12.3-5, 29.17-19, 31.56
ICHRCX02 14.3-8, 20.47-53
ICHRIXO2 exit 6.61-66
ICHRRCDE 9.45-59
ICHRRCDX 9.45-59
IDCAMS 7.53-59
Information 20.54-58, 21.52-58, 22.54-62, 25.3-9, 27.59-63, 28.65-67, 29.60-62, 30.63-66
Inquiry program 8.14-29
INSO7O 10.23-42
IRRADUOO 12.28-57
IRRBDUOO 5.41-61
IRREUXO1 9.43-45
IRRPNLOO API 5.18-33
IRRUT100 5.11-17
ISFUSER 1.53-62
ISPF 16.7-19, 17.24-34, 17.34-42, 18.17-27
ISPF dialogue 10.23-42
ISPF table interface 1.22-47, 8.32-59, 10.3-19, 11.36-59
JEDSPSVC 3.29-36
JES2 31.50-54
JSEAPIO1 8.14-29
JSERACFQ 8.14-29
LDMAPFOO 6.33-60
LDMAPFOI 10.20-22
LISTGRP 11.36-59
Listing accesses 5.11-17
LISTUSER 10.43-59, 11.6-18, 11.36-59, 13.56
McAfee 26.41-44
McAfee e50 26.45-59
MVS consoles 3.49-52
MVS definitions 15.34-55, 16.36-55
Naming conventions 29.57-58
Naming convention exit 4.37-39
Non-generic dataset profiles 29.57-59
Orphan entries 18.52-59
OPERATIONS attribute 26.12-20, 29.13-16
OPERESET 4.3-7
OS/390 Security Server Release 3 9.43-45
PASSRST 9.3-11
PassTicket 20.6-10
PASSWORD 6.31-33
Password 2.3-17, 2.22-27, 4.40-62, 6.61-66, 9.3-11, 12.3-5, 14.39-49, 16.3-8, 20.19-46, 22.33-41, 30.3-15, 30.59-60
Password crackers 22.33-41
PASSWORD/PW 2.45-50
Pentland Utilities 31.14-28
PNLSRAPI 5.18-33
Product list 23.57-63
Profile Name List 5.18-33
Profiles 8.32-59
Profiles 16.3-6
PROTECT ALL 14.3-8
Protecting TSO session 6.31-33, 7.15-16
Protection mechanisms 8.3-11
PWLCHCK 4.28-37
PWLRACF 3.3-8
RACDEF exit 21.3-6
RACF database 1.48-49, 29.3-12
RACF education 24.33-58
RACF group 24.3-14, 24.15-28
RACF installation data 19.3-8
RACF internals 21.14-36
RACF members 10.3-19
RACFPROF 16.7-19, 17.34-42
RACF restructuring 27.8-22, 28.35-50, 29.20-36, 30.27-41
RACFREV 19.8-12
RACF simulator 10.23-42
RACF validate 13.57-59
RACF Version 2.2 3.24-29
RACFCMSK 7.3-15
RACFDSN 1.27-47, 10.3-19
RACFGEN 8.32-59, 10.3-19

RACFGRP 11.36-59
RACFINF 2.28-44
RACFMEM 10.3-19
RACFOPTS 7.25-52
RACFPROF 5.11-17
RACFROUT 9.45-59
RACFSIM 10.23-42
RACFUSER 10.43-59, 11.6-18, 5.11-17
RACFUTIL 4.40-62
RACFVARS 10.3-19
RACFX 4.14-28
RACFXREF 5.11-17
RACFY 5.61-66
RACGET 1.50-52
RACHECK exit 21.3-6
RACROUTE 4.3-7, 4.28-37, 9.3-11
RAIL 3.8-11
RCUT 7.3-15
Remote security 22.28-32, 25.38-43, 26.36-40, 26.41-44, 26.45-59, 28.59-64
Removing commands 2.45-50
Report Writer 12.28-57
Report writer 13.24-55, 14.9-35, 15.7-27, 16.20-35, 17.8-23
Resetting password 2.3-17, 4.3-7
Resource checking 7.53-59
Resource Class Descriptor Table 9.45-59
Retrieving dataset profiles 5.61-66
Retrieving USERDATA 4.7-13
Revoked users 15.3-7
REXRACFG REXX function 24.3-14
REXRACFS REXX function 24.15-28
REXX 21.6-13
RFRONT 4.62-63
RJE jobs 2.17-21
RRSF 16.56-59, 16.59, 17.3-8
RSETUSER 2.3-17
SAF router exit 2.17-21
SASADUOO 12.28-57
SAS/CPE 18.10-16
Save RACF database 1.48-49
Scrollable commands 4 62-63
SDSF exit ISFUSER 1.53-62
SDSF 18.3-10
Search and display 4.14-28
Search command 31.11-14
SECDATA 11.3-6
Security checking 21.6-13
Security investigation 3.49-52
SETROPTS 13.3-18
SETROPTS LIST 7.25-52
Simulator 10.23-42
SKED8OF 11.36-59, 12.6-26
SMRM 3.12-24
Software protection 8.3-11
SPECIAL access 31.55-57
SQL 27.33-46
STANDBY 6.31-33
STARTCLS 6.11-31
STARTED 6.11-31
Started procedures 6.11-31
Started task 12.27
STGADMIN 2.50-63
Storage manager 3.12-24, 26.12-20
Structured display 10.43-59, 11.6-18
Symantec Firewall/VPN 100 27.47-53, 28.59-64
SYSGCATA 7.53-59
System ‘hacks’ 14.36-38
System symbols 23.3-21
Tape 26.3-11
Timed Permit Facility 1.3-10
TIMEDPE 1.3-10
TIMEDREM 1.3-10
TMON 13.19-23
TSO 17.42-59
TSO/E PROTECT/PROT 2.45-50
TSOPREF 1.10-21
UID 25.44-55, 30.61-62
Universal access 29.13-16
Unknown cateeories 11.3-6
Updating TSO prefix 1.10-21
User access 2.28-44, 4.14-28
USERDATA 4.7-13
User data 17.42-59
User id 13.19-23, 27.3-7, 29.44-56
User profiles 19.8-12
Users and datasets 4.14-28
Users’ accesses 5.11-17
USS 25.44-55
Virus 26.36-40, 26.41-44, 26.45-59, 31.58
VRA 15.56-59
Watchguard SOHO 25.44-55, 27.54-58, 28.59-64
WEAKPASS 2.22-27
Web user identification 22.42-53
Year 2000 8.30-31
zSeries 24.29-32, 26.21-22

Contributing to *RACF Update*

In addition to *RACF Update*, the Xephon family of *Update* publications now includes *CICS Update*, *MVS Update*, *DB2 Update*, *TCP/SNA Update*, *AIX Update*, *and MQ Update*. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon's wide range of products and services, simply by telling us all about it.

More information about contributing an article to a Xephon Update, and an explanation of the terms and conditions under which we publish articles, can be found at http://www.xephon.com/index/nfc. Alternatively, please write to the editor, Fiona Hewitt, at any of the addresses shown on page 2, or e-mail her at fionah@xephon.com

IBM is adding a new product to the SCLM (Software Configuration and Library Manager) Suite family: Enhanced Access Control for SCLM for z/OS. Rules specify exactly which programs can be used to access which datasets, which SCLM functions each user can access, and a user's read or write access to specific documents.

URL: http://www.ibm.com/software/ad/sclmsuite/accesscontrol

* * *

Vanguard Security Solutions Version 4.3 includes two new products. ezRIGHTS extends the mainframe's security authorization, verification, and auditing to Windows NT/2000/XP. ezADMIN is a RACF administration API that runs on almost any platform, including laptops, Personal Digital Assistants (PDAs), and cell phones. Novell is a new platform for ezSIGNON while ezAPI now runs on z/OS Unix System Services (USS) and the iSeries 400. Vanguard Administrator gets a Drill Down feature and Enforcer's Started Task sensor has been rewritten with several enhancements.

URL: http://www.go2vanguard.com/software_solutions/new_release.cfm

* * *

Systor has integrated its Security Administration Manager (SAM) products with SAM Jupiter. Its Java-based Web-enabled interface provides a central point of security administration, including Role-Based Access Control (RBAC) and policy enforcement. Security provisioning makes tools and resources immediately available for new employees, business partners, and customers.

Product components include a provisioning engine, off-the-shelf target system interfaces, repository, delegation authority, and auto discovery, auto policy enforcement, and reconciliation facilities. SAM Jupiter Provisioning Workflow automates the application process for accounts and access rights.

URL: http://www.sam-security.com

* * *

Tivoli Access Manager for Operating Systems is a policy-based access control system that provides RACF-like security to Unix platforms. Version 4.1 can now run on the Red Hat and SuSE distributions of Linux for zSeries (zLinux). AIX, HP-UX, Solaris and Intel-based Linux continue to be supported.

URL: http://www.tivoli.com/products/index/access-mgr-operating-sys

* * *