In this issue

3  Enhanced dataset security
9  RACF in focus – finding groups that have no permits
16 Using IRREVX01 to cross-reference OMVS segment UID assignment during ADDUSER or ALTUSER
42 RACF 101 – your questions answered
46 Checking resource profiles for orphaned IDs
62 RACF news

© Xphon Inc 2005
RACF Update

Published by
Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA
Phone: 214-340-5690
Fax: 214-341-7081

Editor
Trevor Eddolls
E-mail: trevore@xephon.com

Publisher
Colin Smith
E-mail: info@xephon.com

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 $290.00 in the USA and Canada; £190.00 in the UK; £196.00 in Europe; £202.00 in Australasia and Japan; and £200.50 elsewhere. The price includes postage. Individual issues, starting with the August 2002 issue, are available separately to subscribers for $72.75 (£48.50) each including postage.

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 the rate of $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. To find out more about contributing an article, without any obligation, please download a copy of our Notes for Contributors from www.xephon.com/nfc.

© Xephon Inc 2005. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher.

Printed in England.
Enhanced dataset security

Resource Access Control Facility (RACF) provides access control by identifying and verifying users to the system, authorizing access to DASD datasets, and logging both detected unauthorized attempts to enter the system and detected access to protected datasets. However, an additional level of security is advised for specific, sensitive, production datasets that are prone to inadvertent user modification. This security layer ensures superior dataset protection by logging a complete dataset modification history and by backing up pre-modification datasets.

Most RACF administrators, for reasons of economy, maintain a log of only the last two modifications to a dataset. This does not leave much room to explore historical changes carried out by multiple authorized users on such datasets. For similar reasons, in most shops, the Storage Management Subsystem (SMS) retains only the last two versions of a dataset. In such a scenario, the dataset owner must bear the bottom-line responsibility of ensuring that there's an historical log of modifications on such sensitive production datasets and taking complete back-ups.

One essential question that arises is, how does this additional security layer on the datasets differentiate itself from the security provided by conventional RACF and routine back-up tasks of SMS’s hierarchical storage manager? The answer is, to minimize the expense, the security and back-up facilities provided by RACF and SMS respectively are enabled to keep around two change logs (depending on the installation) for millions of datasets. There is no a way to retain complete change logs for a group of datasets, selectively. Moreover, if an essential production dataset is modified a few times a day, SMS will back it up only once during its normal back-up schedule. Consequently, several changes performed in a day go unrecorded. In such circumstances, the additional dataset security comes to your aid. This additional security is invoked
at the dataset level and not at a dataset group level.

The additional security creates a report that tells you who modified the dataset, what the modification date and time was, and where the available back up dataset is. This additional security, once activated for any dataset, will automatically record the change log and, if required, can e-mail interested parties/responsible people about the changes made to the dataset. Of course, one can expect notification for all modifications made to sensitive datasets!

The additional security macro is called SECURE. The code is shown at the end of this article. It is written to perform two functions:

- Log a complete edit history for a specific dataset.
- Keep an automatic back-up of the pre-modification dataset (optional feature).

The supplied macro could be expanded to add the following features:

- E-mail concerned parties whenever the dataset is edited.
- E-mail concerned parties whenever a specific user edits the dataset.
- Back up the datasets in a GDG instead of a flat file.

Add member SECURE to your CLIST or EXEC or any personal library and ensure that this library is concatenated to SYSPROC/SYSEXEC.

**HOW TO INSTALL SECURITY FOR A SPECIFIC DATASET**

Activation of SECURE for a particular dataset is required only once. Open the dataset in view or edit mode and then type in the command PROFILE. This will show the following lines at the top of data in the dataset:

```
****** ***************************** Top of Data *****************
=PROF> ....TITLECD (FIXED - 80)....RECOVERY ON....NUMBER OFF......
```
Then type in the command **IMACRO SECURE** and press *Enter*. This will set the macro SECURE permanently in the profile of that particular dataset. The user will see that the profile is modified to recognize macro SECURE. The profile now looks like this:

```
****** ***************************** Top of Data ****************
=PROF> ....CAPS ON....HEX OFF....NULLS ON STD....TABS OFF.......... 
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS OFF 
=PROF> ....PROFILE UNLOCK....IMACRO NONE....PACK OFF....NOTE ON... 
=PROF> ....HILITE DEFAULT CURSOR FIND............................
```

Now, whenever a user edits this dataset, this macro will execute (before the actual modification of the dataset contents) and perform the specific security task automatically. This macro will not work when the user opens the dataset in browse mode.

**HOW TO UNINSTALL SECURITY FOR A SPECIFIC DATASET**

In order to deactivate the security, open the dataset in view or edit mode and type in the command **PROFILE NONE**. This will restore the IMACRO to the default state. The profile looks like this:

```
****** ***************************** Top of Data ****************
=PROF> ....TITLECD (FIXED - 8Ø)....RECOVERY ON....NUMBER OFF.......  
=PROF> ....CAPS ON....HEX OFF....NULLS ON STD....TABS OFF..........   
=PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS OFF  
=PROF> ....PROFILE UNLOCK....IMACRO NONE....PACK OFF....NOTE ON... 
=PROF> ....HILITE DEFAULT CURSOR FIND............................
```

**HOW TO VARY THE SECURITY LEVEL FOR A SPECIFIC DATASET**

This EXEC has two subroutines. These subroutine names explain their functions:

- **LOG_USERID_DATE_TIME** – logs user-id, date, and time in a physical sequential dataset.
• BACKUP_DATASET – back-up pre-modification dataset.
The user may choose to retain the dataset back-up facility. This feature is optional.

The purpose of the supplied macro is to show the applicability of the idea that users, on specific datasets, could implement dataset-level security. Here are the few assumptions made to keep the code simple:

• A dataset is opened in edit/view mode with the intention of editing. Even if the user does not edit it, a change log is created and a back-up is saved. One can modify the macro to take care of this situation.

• Save a back-up for each edit. It does not compare the current backed up version with the previous backed up version and hence two or more datasets may contain exactly the same data.

• The back-ups are saved in a flat file. This helps this macro to be used for backing up files with different record lengths. It could easily be modified to save back-ups in GDG versions instead of flat files.

A DIFFERENT PERSPECTIVE

This EXEC can also act as spying tool. Set SECURE for a group of datasets and keep logging details or receiving notifications about all the activities performed on the dataset. It is all about how you look at it!

Recommendation: modify the following things in the macro before use:

• Ensure that the security report and back-up datasets have a valid system-recognized first qualifier (instead of user ID). This will make the use of SECURE person/userid independent.

• Ensure that all the users of the dataset on which this SECURE is activated have access to that first qualifier;
otherwise they will get a macro error and no change log will be created.

- All the users intending to use this security feature must set this macro in the profile of their datasets; otherwise this macro will not work for that user.

**CODE**

```rexx
/******************************* REXX ********************************/
/*** Purpose: Log dataset modification details and back up         ***/
/***        : pre-modification dataset contents                    ***/
/*** Input  : The macro must be customized before use             ***/
/***        : 1. Change the userid to a valid system-acceptable    ***/
/***        :    first qualifier.                                 ***/
/***        : 2. Decide how many back-up versions are required and ***/
/***        :    modify accordingly.                              ***/
/***        : 3. Modify the report and back-up file names          ***/
/*** Output : A security report and dataset back up (optional)    ***/
/*Execution : Set the macro in PROFILE of a dataset               ***/
/***        : For PDS, a change in profile for one member will     ***/
/***        : change the profile of all the members.              ***/
/***        : For PDS, this macro will back-up only a particular   ***/
/***        : member on which this macro executes. It can be tweaked**/
/***        : to back-up complete PDS (instead of a particular member)*/
/***        :                                                      ***/
/*** Author : Yash (Jun 19, 2005) - Father's day                   ***/
*********************************************************************/
ADDRESS ISREDIT "MACRO"
/* trace ?i */
/* get the executing dataset name */
"ISREDIT (mem) = MEMBER"
"ISREDIT (pds) = DATASET"
mem = strip(mem)
pds = strip(pds)
/* To back up complete PDS, comment the next line */
dsn = pds
IF mem <> '' then do
    dsn = pds ||"("||mem||")"
end
/* If no back-up is required, comment the next line */
call BACKUP_DATASET
call LOG_USERID_DATE_TIME
exit
BACKUP_DATASET:
EMPLATE dataset name that is catalogued in the system */
="/*************************************************************/
```
/* modify the root dsn to suit your requirements */
rootdsn = USERID()".TEST.BACKUP"
"ISPEXEC LMDINIT LISTID(ID1) LEVEL("rootdsn")"
"ISPEXEC LMDLIST LISTID("ID1") DATASET(DSVAR)"
COUNT = Ø
DO WHILE RC = Ø
  COUNT = COUNT + 1
  record.COUNT = DSVAR
  "ISPEXEC LMDLIST LISTID("ID1") DATASET(DSVAR)"
END
RC = Ø
lastdsn = record.count
if COUNT = Ø then do
  bkpdsn= rootdsn||".ØØØ1"
  end /* end for do */
else do
  parse var lastdsn part1 '.#' part2
  part2 = part2 + 1
  /*can back up 9,999 versions; can be easily expanded to store more*/
  select
    when length(part2) = 1 then do
      part2 = 'ØØØ'||part2
    end
    when length(part2) = 2 then do
      part2 = 'ØØ'||part2
    end
    when length(part2) = 3 then do
      part2 = 'Ø'||part2
    end
    Otherwise nop
  end /* end for select */
bkpdsn = part1 || '.#' || part2
  end /* end for do */
/* copy the dataset into a new version */
cmd = "COPY '"||dsn||"' '"||bkpdsn||"'"
interpret cmd
return /* end of BACKUP_DATASET subroutine */

LOG_USERID_DATE_TIME:
/*--------------------------------------------------------------*/
/* Create a dataset modification log file */
/*--------------------------------------------------------------*/
secdsn = USERID()".TEST.SECURE.REP"
/* If the security report dataset does not exist, allocate it */
/* Add a report title and dataset name to it */
if SYSDSN("secdsn") <> 'OK' then do
  "ALLOCATE DA("secdsn") NEW SPACE(30,20) TRACK LRECL(80)
  FILE(report) RECFM(F,B) BLKSIZE(27920) UNIT(sysda) new reu"
  /* format report title */
  /* report will contain the date and time of the day of allocation*/
RACF in focus – finding groups that have no permits

This is a regular column focusing on specific aspects of RACF. In the last issue we looked at one type of redundancy in the
RACF database – that of groups having no users connected. In this issue we look at a related issue – RACF groups having no permits (permissions) in any of the RACF profiles. These groups, too, are potentially redundant.

In the RACF in focus column in the last issue we saw that most RACF databases have redundant RACF groups. These crop up as a result of the daily RACF administrative work, and there is not much we can do about this except find ways (hopefully, automated) to address the issue.

Keeping your RACF database free from redundant groups is in your own best interest. If not addressed, redundant RACF groups will accumulate to the point where daily administrative work will get bogged down, not to mention the performance of RACF itself being slowed down.

RACF grouping simplifies the granting of access to resources and dataset profiles. So, if a RACF group exists in the database, but does not have any permits (permissions) in any of the profiles in the RACF database, and the group does not have any sub-groups, what is it doing there? Is it a redundant group? Most likely, yes.

SELECTION CRITERIA
The REXX routine described below will find all groups that meet the following criteria:

- The group is not a dataset High-Level Qualifier (HLQ).
- The group does not have any sub-groups.
- The group is not in any dataset or resource profile permission list.

Of course, even when all these conditions are met, a RACF group still may not be redundant. We can never say for sure, because it could be a newly-created group, and permissions may have been planned for the future. But if we can somehow list all groups in the RACF database that meet the above
criteria, we can then do a further manual check to make sure the groups are indeed redundant before deleting them.

The output from the REXXX routine is a list of groups that meet the above criteria. This list can be examined for redundancies. The first time you run this routine at your installation, you are likely to get a long list, but after a few iterations and clean-ups the on-going list should be very small and manageable.

The REXX routine produces, for each redundant group, commands to remove all userids connected to the group (for RACF will not allow deletion of a group if there are users connected to it), and then the delete command is produced to delete the group itself.

BATCH JCL

The REXX EXEC (NOPERMGP) runs in batch mode for convenience, and uses the following JCL:

```
//NOPERJOB JOB ( ...),'YOUR NAME', MSGCLASS=X,CLASS=X,NOTIFY=&SYSUID
//STEPØ1 EXEC PGM=IKJEFTØ1,REGION=2M
//SYSTSPRT DD SYSOUT=*  
//INDSNHLQ DD DSN=HLQ.LIST,DISP=SHR  
//INDBU1ØØ DD DSN=REC.TYPE1ØØ,DISP=SHR  
//INDBU1Ø1 DD DSN=REC.TYPE1Ø1,DISP=SHR  
//INDBU1Ø2 DD DSN=REC.TYPE1Ø2,DISP=SHR  
//INDBU4Ø4 DD DSN=REC.TYPE4Ø4,DISP=SHR  
//INDBU5Ø5 DD DSN=REC.TYPE5Ø5,DISP=SHR  
//OUTGROUP DD DSN=OUTPUT.NOPERM.PDS(NOØ5),DISP=SHR  
//SYSTSIN DD *  
NOPERMGP  
/*

Input files:

- **INDSNHLQ** – this DDname points to the file containing a sorted list of all valid dataset high-level qualifiers at your installation. The dataset high-level qualifiers start at column 15.

- **INDBU100** – this DDname points to the sorted file containing all type 100 records from the RACF unloaded database.
INDBU101 – this DDname points to the sorted file containing all type 101 records from the RACF unloaded database.

INDBU102 – this DDname points to the sorted file containing all type 102 records from the RACF unloaded database.

INDBU404 – this DDname points to the sorted file containing all type 404 records from the RACF unloaded database.

INDBU505 – this DDname points to the sorted file containing all type 505 records from the RACF unloaded database.

Output file:

OUTGROUP – this DDname points to the output file that will contain the list of possible redundant groups, and relevant RACF commands to remove these groups. The dataset is a PDS, with member names reflecting the month in which the list was produced.

OUTPUT FROM THE REXX ROUTINE

The REXX routine below, called NOPERMGP, generates a set of RACF commands to remove the userids connected to redundant groups, followed by a command to delete the group itself.

It does this for each potentially redundant group it finds, based on criteria mentioned above. The output is placed in dataset OUTPUT.NOPERM.PDS(NOV05). The list is as follows:

REMOVX USER123 GROUP(GROUPA)
REMOVX USER678 GROUP(GROUPA)
...
DELGRX GROUPA
REMOVX USER111 GROUP(GROUP123)
REMOVX USER234 GROUP(GROUP123)
...
DELGRX GROUP123
REXX ROUTINE FLOW CHART
Here’s how the REXX routine’s logic flows:

- Is it dataset HLQ? Yes – exit.
- If no – does it have sub-groups? Yes – exit.
- If no – does it have any dataset permissions? Yes – exit.
- If no – does it have any resource profile permissions? Yes – exit.
- If no – produce userid remove commands and group delete commands for this group.

THE PROCESS
Briefly, we need to do the following periodically (once a month is recommended):

1. Identify all potentially redundant RACF groups having no permits, using the REXX routine.
2. Review manually the output list to verify that the groups are indeed redundant.
3. Execute the delete commands generated by the REXX routine to remove the redundant groups.

The program deliberately produces REMOVX and DELGRX commands instead of their correct spellings, REMOVE and DELGRP. This allows you time to review the list, and verify that the group names are indeed redundant before submitting them for deletion. It also prevents accidental execution of the commands.

Once you are satisfied that you are ready to remove the redundant groups, edit the output dataset and enter the following change commands in an ISPF session:
and:

CHANGE ' REMOVX ' ' REMOVE ' ALL 1

CHANGE ' DELGRX ' ' DELGRP ' ALL 1

You will find that the first time you run this process, you will catch many redundant groups having no permits. After that, on an on-going basis, you will only find a few.

THE REXX ROUTINE

/*    REXX                                               */
/*************************************************************/
/* NAME: NOPERMGP                                        */
/*                                                       */
/* PURPOSE: THIS REXX WILL REPORT ON GROUPS THAT -        */
/*                                                       */
/* 1. HAVE NO PERMITS IN ANY ACCESS LIST, AND -          */
/* 2. HAVE NO SUB-GROUPS, AND -                          */
/* 3. ARE NOT DATASET HLQ GROUPS                         */
/*************************************************************/
"EXECIO  *  DISKR  INDSNHLQ (STEM  INHLQ.  FINIS)";
"EXECIO  *  DISKR  INDBU100 (STEM  IN100.  FINIS)";
"EXECIO  *  DISKR  INDBU101 (STEM  IN101.  FINIS)";
"EXECIO  *  DISKR  INDBU102 (STEM  IN102.  FINIS)";
"EXECIO  *  DISKR  INDBU404 (STEM  IN404.  FINIS)";
"EXECIO  *  DISKR  INDBU505 (STEM  IN505.  FINIS)";
DO I =1 TO INHLQ.Ø
    PARSE VAR INHLQ.I  JUNK1  15  HLQGRP.I  23  JUNK2
END
DO I =1 TO IN100.Ø
    PARSE VAR IN100.I  JUNK1  6  GRP100.I  14  JUNK2
END
DO I =1 TO IN101.Ø
    PARSE VAR IN101.I  JUNK1  6  GRP101.I  15  ID101  23  JUNK2
END
DO I= 1 TO IN102.Ø
    PARSE VAR IN102.I  JUNK1  6  GRP102.I  15  ID102.I  23  JUNK2
END
DO I= 1 TO IN404.Ø
    PARSE VAR IN404.I  JUNK1  58  GRP404.I  67  JUNK2
END
DO I= 1 TO IN505.Ø
    PARSE VAR IN505.I  JUNK1  262  GRP505.I  271  JUNK2
END
DO J = 1 TO IN100.Ø
    DSNHLQ= 'NO'
    DO K = 1 TO INHLQ.Ø
IF GRP100.J = HLQGRP.K THEN DO
  DSNHLQ = 'YES'
  K = INHLQ.Ø
END

IF DSNHLQ = 'NO' THEN DO
  DO L = 1 TO IN101.Ø
    SUBGRP = 'NO'
    IF GRP100.J = GRP101.L THEN DO
      SUBGRP = 'YES'
      L = IN101.Ø
    END
  END

IF SUBGRP = NO THEN DO
  DO M = 1 TO IN404.Ø
    PER404 = 'NO'
    IF GRP100.J = GRP404.M THEN DO
      PER404 = 'YES'
      M = IN404.Ø
    END
  END

IF PER404 = NO THEN DO
  DO N = 1 TO IN505.Ø
    PER505 = 'NO'
    IF GRP100.J = GRP505.N THEN DO
      PER505 = 'YES'
      N = IN505.Ø
    END
  END

IF PER505 = 'NO' THEN DO
  DO P = 1 TO IN102.Ø
    IF GRP100.J = GRP102.P THEN DO
      QUEUE 'REMOVX' ID102.P 'GROUP('STRIP(GRP100.J)||')'
      END
  END
QUEUE 'DELGRX' GRP100.J
END

IF QUEUED() = Ø THEN DO
  QUEUE 'NO GROUPS TO REPORT'
END

"EXECIO * DISKW OUTNOPER (FINIS)"
EXIT

IN CONCLUSION

Daily RACF administration inevitably results in some group
redundancies. Finding these useless groups and removing them is necessary, otherwise they will only accumulate, like junk in the basement.

The ‘RACF in focus’ column in the last issue dealt with one type of group redundancy, and this issue deals with a related area. Periodically running the REXX routine shown above, and the one shown in the last issue, is the answer to automatically cleaning these redundant groups.

It is recommended you do this at least once a month.

These processes can be placed in your RACF ‘tool-kit’, among other aids to clean up other aspects of RACF redundancies.

_Dinesh Dattani would welcome feedback, comments and queries about this column. He can be contacted at dinesh123@rogers.com_

_Dinesh Dattani_  
Mainframe Security Consultant  
Toronto (Canada)  
© Xephon 2005

**Using IRREVX01 to cross-reference OMVS segment UID assignment during ADDUSER or ALTUSER**

An age-old problem that exists in Unix environments is the ability to assign the same numeric UID to multiple different userids. In some cases, this is a perfectly valid thing to do. For example, several userids will often be assigned a UID of 0 because they have special processing needs. Also, it might be valid to have multiple userids that belong to the same person assigned the same UID. In most cases, though, assigning the same UID to multiple different userids is, at worst, an error and at best a serious oversight.
With Unix System Services (USS) on z/OS systems that use RACF, this problem does not go away. Recent releases of RACF do a better job of maintaining the UNIXMAP class with corresponding UID profiles, but there is still no prevention or warning when a UID that is already in use is about to be reassigned through either the ADDUSER or ALTUSER command. Also, for historical RACF databases where UID assignments have been done for a few years, the UNIXMAP class may not be up to date with all the assigned UID values.

This article discusses an IRREVX01 RACF exit that can be used to warn RACF administrators when they are assigning a previously-used UID to another RACF userid.

**HOW THE IRREVX01 EXIT WORKS**

The IRREVX01 exit provided with this article captures RACF ADDUSER or ALTUSER commands and examines the command buffer for an OMVS segment UID assignment. If an OMVS UID assignment is detected, a couple of optional processes can be triggered, depending on what conditional assembly options are selected.

The IRREVX01 exit has code for both a foreground and a background component. Enabling the foreground option allows the RACF database to be searched while the ADDUSER or ALTUSER command is active. The foreground option produces messages for both the operator console and the invoking user that identify UID conflicts with the requested UID. The background option dynamically allocates the external reader and writes batch job JCL to spool for background execution. This background job produces an output listing identifying UID conflicts with the requested UID and performs an additional operation that does not occur in the foreground mode – it produces a table of available UID ranges that are not currently assigned to any userid.

Using the foreground option has the obvious advantage of notifying the user of a UID conflict while the ADDUSER or
ALTUSER command is running. Unfortunately, for sites with a large RACF database, the length of time the real-time cross-reference takes may not be acceptable. If this is the case for your site, running with only the background option enabled is probably a better alternative.

THE UIDXREFX PROGRAM FOR IRREVX01

Up to this point, we have discussed the IRREVX01 exit for RACF. The IRREVX01 exit has been defined, by IBM, using the dynamic exits facility. Because of that, new exit code can be activated dynamically through a SETPROG operator command and the module name does not have to be IRREVX01. Since it has a little more meaning, we will refer to the IRREVX01 exit program for this article as UIDXREFX (UID cross-reference exit).

Before assembling UIDXREFX, you will need to determine which processing modes you will want to enable. The program supports two conditional assembly options:

- &FOREGROUND SETC 'ON'
- &BACKGROUND SETC 'ON'.

If your RACF database is not large (under 2,000 userids), you can probably select &FOREGROUND SETC 'ON' in the program source. This will enable real-time reporting of detected UID conflicts. Because the background batch job also produces a table of available UIDs if a conflict is detected, setting &BACKGROUND SETC 'ON' is also a viable option even if your RACF database is not large. If your RACF database is relatively large (certainly anything more than 5,000 userids), you will probably want to enable only the background option.

If you enable the foreground option, expect the ADDUSER or ALTUSER command that includes a request for an OMVS UID to take a few seconds of wall clock time even on smaller RACF databases. The UID cross-reference can take a minute or more on a database with 10,000 userids and the ADDUSER
or ALTUSER command will not complete (show the READY prompt) until the UIDXREFX exit has completed its processing. Be aware of this delay when the foreground option is enabled.

If the background option is enabled, you will have to review the embedded JCL statements (program labels JCL1 through JCL7) in the UIDXREFX source. These statements will have to conform to your site's JCL standards. The UIDXREFB program referenced on the EXEC statement is the background batch program provided with this article. It must reside in the STEPLIB dataset (or, optionally, the linklist) specified in the embedded JCL.

Here's sample JCL to link-edit the two modules:

```
//IEWL    EXEC  PGM=HEWLH096,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD    SYSOUT=*  
//SYSUT1   DD    UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT   DD    DSN=object.code.pds,DISP=SHR
//SYSLIB   DD    DSN=SYS1.CSSLIB,DISP=SHR
//SYSLMOD  DD    DSN=apf.auth.library,DISP=SHR
//SYSLIN   DD    *
       INCLUDE OBJECT(UIDXREFX)
       ENTRY   UIDXREFX
       SETCODE AC(1)
       NAME    UIDXREFX(R)
       INCLUDE OBJECT(UIDXREFB)
       ENTRY   UIDXREFB
       SETCODE AC(1)
       NAME    UIDXREFB(R)

Once the UIDXREFX exit has been link-edited, it can be dynamically activated with a z/OS operator command as follows:

SETPROG
EXIT,ADD,EXITNAME=IRREVX01,MODNAME=UIDXREFX,DSNAME=apf.auth.library

If the exit is not performing as expected, or you simply want to disable its effects, it can be deleted with the following z/OS command:

SETPROG EXIT,DELETE,EXITNAME=IRREVX01,MODNAME=UIDXREFX
TRIGGERING THE UIDXREFX EXIT

Once the exit has been activated, it can be triggered with any RACF ADDUSER or ALTUSER command that includes a request for an OMVS UID. Here’s a sample TSO command:

```
ALU USER01 OMVS(UID(1106))
```

If the foreground option is enabled and there are existing conflicts with the specified UID, messages similar to the following will appear on the operator console and at the issuing user’s TSO session:

- IRREVX01 – specified UID(0000001106) for userid USER01 previously assigned to userid USER27.
- IRREVX01 – specified UID(0000001106) for userid USER01 previously assigned to userid USER122.
- IRREVX01 – specified UID(0000001106) for userid USER01 previously assigned to userid ACCT05.

If the background option is enabled, the batch job JCL embedded in the UIDXREFX program will be submitted to the internal reader. This batch job will look for UID conflicts similar to the foreground processing option, but it will also produce a table of available OMVS UIDs. Output from this batch job (written to the SYSPRINT output DD) will look similar to the following:

```
Specified UID(0000001106) for userid USER01 previously assigned to userid USER27.
Specified UID(0000001106) for userid USER01 previously assigned to userid USER122.
Specified UID(0000001106) for userid USER01 previously assigned to userid ACCT05.

Available UIDS: 000000002 - 000000047
Available UIDS: 000000049 - 000000076
Available UIDS: 000000078 - 000000122
Available UIDS: 000000124 - 000000205
Available UIDS: 000000207 - 000000415
Available UIDS: 000000417 - 000000444
Available UIDS: 000000446 - 000000446
Available UIDS: 000000536 - 000000536
Available UIDS: 000000538 - 000000997
```
Available UIDS: 000000999 - 000000999
Available UIDS: 000001005 - 000001007
Available UIDS: 000001011 - 000001011
Available UIDS: 000001013 - 000001014
Available UIDS: 000001019 - 000001052
Available UIDS: 000001055 - 000001100
Available UIDS: 000001108 - 000001199
Available UIDS: 000001201 - 000001249
Available UIDS: 000001251 - 000001255
Available UIDS: 000001257 - 000001259
Available UIDS: 000001261 - 000002000
Available UIDS: 000002011 - 000002999
Available UIDS: 000003002 - 000006665
Available UIDS: 000006668 - 000008999
Available UIDS: 000009001 - 000009897
Available UIDS: 000009907 - 000009998
Available UIDS: 000010000 - 000009897
Available UIDS: 000098980 - 000009989
Available UIDS: 000100000 - 0000818180
Available UIDS: 000818182 - 2147483647

If the background job detects no UID conflicts, a single output line is produced similar to this:

No UID conflict for userid USER01 and UID 0012345678

CONCLUSION

The IRREVX01 exit provided with this article is not designed to prevent the use of conflicting UID values, but it is designed to report on situations that will generate a conflict. This allows the RACF administrator to review the situation after the fact and make a better decision on which UID to use. Improvements to the management of OMVS UID values are evolving in RACF, but assigning multiple userids to the same UID is still possible today. The UIDXREFX exit for IRREVX01 and its associated batch job program, UIDXREFB, provide an additional set of tools to better manage this challenge. You will have to test which processing option works best for you — foreground, background, or both. When you have made that decision, the UIDXREFX exit and the associated UIDXREFB batch program should prove helpful.
UIDXREFX ASSEMBLER

&FOREGROUND SETC 'ON'  <= Set to 'ON' for foreground notification
&BACKGROUND SETC 'ON'  <= Set to 'ON' for background processing

UIDXREFX CSECT
UIDXREFX AMODE 31
UIDXREFX RMODE ANY

*---------------------------------------------------------------------*
* This IRREVXØ1 exit can be used to assist in managing OMVS segment *
* UID assignment. The exit examines the incoming command request *
* checking for ADDUSER or ALTUSER RACF commands that include a *
* request for an OMVS UID. Depending on the conditional assembly *
* settings, this exit supports both a foreground and background *
* option (they can both be enabled simultaneously).  
* *
* With &FOREGROUND SETC 'ON', this exit will produce real-time *
* messages to the console and to the issuer of the ADDUSER or *
* ALTUSER command indicating a conflict in UID assignment. Due *
* to the size of a site's RACF database and the amount of checking *
* required, waiting for this conflict assessment to occur in *
* real-time may not be feasible. If that is the case, using the *
* background processing option may be more practical.  
* *
* With &BACKGROUND SETC 'ON', this exit will allocate the internal *
* reader and submit a batch job that performs a background *
* assessment of UID conflict. The advantage of the background *
* batch job over foreground processing is that the background job *
* will not only report on UID conflicts that may occur with the *
* selected UID, but it will also produce a table of available *
* OMVS UIDs. Armed with this information, the RACF administrator *
* will be able to make more appropriate UID value selections.  
* *
* If the background option is enabled, the JCL statement images *
* defined by constants JCL1 through JCL7 in this source deck will *
* have to be changed to contain a proper jobname, account number, *
* programmer name, notify userid, and STEPLIB dataset. The *
* UIDXREFB program referenced by this JCL is the background batch *
* job also provided with this exit. The CLASS and MSGCLASS may *
* also need to be modified to meet site standards.  
* *
* This exit is entered from RACF in supervisor state, key Ø so *
* be careful.  
* *---------------------------------------------------------------------*

STM   R14,R12,12(R13)       SAVE INCOMING REGISTERS
LR    R12,R15               COPY MODULE ADDRESS
USING UIDXREFX,R12          SET ADDRESSABILITY
LR    R2,R1                 SAVE INCOMING PARM ADDRESS
LR    R11,R13               SAVE OLD SAVEAREA ADDRESS
Storage OBTAIN,LENGTH=WORKLEN,LOC=BLOW
LR    R13,R1                 GET NEW SAVEAREA ADDRESS
LR   R0,R1       COPY ADDRESS
LR   R14,R1      AGAIN
L    R1,=A(WORKLEN) GET LENGTH
XR   R15,R15     SET FILL BYTE
MVCL R0,R14     CLEAR THE STORAGE
USING WORKAREA,R13 SET ADDRESSABILITY
ST   R11,SAVEAREA+4 SAVE OLD SAVEAREA ADDRESS
***********************************************************************
USING EVXPL,R2 SET PARAMETER ADDRESSABILITY
***********************************************************************
LR   R3,EVXFLAGS GET FLAG POINTER
TM   Ø(R3),EVXPRE PREPROCESSING CALL?
BO   PRECALL YES - ISSUE WTO
TM   Ø(R3),EVXPOST POSTPROCESSING CALL?
BO   POSTCALL YES - ISSUE WTO
B    RETURN WE'RE DONE
PRECALL EQU *
***********************************************************************
L    R3,EVXCALLR GET FUNCTION CODE BYTE ADDRESS
CLI   Ø(R3),EVXADDUS ADUXER COMMAND?
BE   PREWORK YES - GO PROCESS
CLI   Ø(R3),EVXALTUS ALTUSER COMMAND?
BE   PREWORK YES - GO PROCESS
B    RETURN NO - JUST RETURN
PREWORK EQU *
BAL   R14,UIDCHK CHK FOR 'OMVS UID' IN CMD BUFFER
LTR   R15,R15   A 'UID'?
BZ    UIDOK YES - GO PROCESS
B    RETURN
UIDOK EQU *
ST   R0,UIDBIN SAVE UID
BAL   R14,GETUSRID GO ISOLATE THE USERID
.
.
AIF ('&FOREGROUND' NE 'ON').BYPASS_F1
.
.
XC   XUID(4),XUID CLEAR LENGTH AREA
MVC   XUID(2),=H'8' SET LENGTH
MVC   USERID(8),=8C' ' SET STARTING USER ID VALUE
USERIDLP EQU *
XC   RACWORK(256),RACWORK
XC   RACWORK+256(256),RACWORK+256
MVC   ROUTWRK1(ROUTLEN1),RACROUT1
RACROUTE REQUEST=EXTRACT, TYPE=EXTRACTN, ENTITYX=XUID, RELEASE=1.9.2, FIELDS=FLDLIST1, SUBPOOL=1, WORKA=RACWORK, MF=(E,ROUTWRK1)
LTR   R15,R15 EXTRACT OK?
BNZ  CHKLIST  NO - BUG OUT  
LR  R6,R1  COPY THE EXTRACT AREA ADDRESS  
***********************************************************************  
XR  R8,R8  CLEAR R8  
XR  R9,R9  CLEAR R9  
IC  R9,Ø,(R6)  SAVE THE SUBPOOL VALUE  
ICM  R8,Ø'0111',1(R6)  SAVE W/A LENGTH  
STORAGE RELEASE,LENGTH=(R8),ADDR=(R6),SP=(R9)  
***********************************************************************  
XC  RACWORK(256),RACWORK  
XC  RACWORK+256(256),RACWORK+256  
MVC  ROUTWRK2(ROUTLEN2),RACROUTE2  
RACROUTE REQUEST=EXTRACT,  X  
TYPE=EXTRACT,  X  
ENTITY=USERID,  X  
RELEASE=1.9.2,  X  
FIELDS=FLDLIST2,  X  
SUBPOOL=1,  X  
WORKA=RACWORK,MF=(E,ROUTWRK2)  
LTR  R15,R15  OMVS SEGMENT?  
BNZ  USERIDLP  NO - CHECK NEXT USERID  
***********************************************************************  
USING EXTWKEA,R6  EXTRACT WORKAREA ADDRESSABILITY  
LR  R6,R1  GET EXTRACT WORKAREA  
XR  R8,R8  CLEAR R8  
XR  R9,R9  CLEAR R7  
IC  R9,Ø,(R6)  GET SUBPOOL  
ICM  R8,Ø'0111',1(R6)  GET LENGTH  
XR  R15,R15  CLEAR R15  
ICM  R15,ØØ'11',EXTWOFF  GET OFFSET OF DATA AREA  
AR  R15,R6  POINT TO UID AREA  
ICM  R14,Ø'1111',Ø(R15)  GET UID LENGTH  
MVC  EXTUID(4),4(R15)  COPY EXTRACT UID  
STORAGE RELEASE,LENGTH=(R8),ADDR=(R6),SP=(R9)  
***********************************************************************  
CLC  EXTUID(4),=4X'FF'  RESERVED UID?  
BE  USERIDLP  YES - GET NEXT USERID  
CLC  EXTUID(4),UIDBIN  SAME UID?  
BNE  USERIDLP  NO - GET NEXT USERID  
MVC  WTOWRK(WTOLN),WTOLST  COPY WTO MODEL  
MVC  WTOWRK+4+48(8),USRIDSARV  COPY USERID  
MVC  WTOWRK+4+87(8),USERID  COPY USERID  
L  R15,EXTUID  GET UID VALUE  
CVD  R15,DBL2  CONVERT TO DECIMAL  
UNPK  DBL1(16),DBL2(8)  UNPACK THE VALUE  
OC  DBL1(16),=16X'FØ'  MAKE IT READABLE  
MVC  WTOWRK+4+25(1Ø),DBL1+6  COPY UID  
WTO  MF=(E,WTOWRK)  WRITE A MESSAGE  
B  USERIDLP  GET NEXT USERID  
***********************************************************************
**BUILD A DYNAMIC ALLOCATION PARAMETER LIST IN WORKING STORAGE FOR ALLOCATING THE INTERNAL READER.**

```
LA R0,DYNALWRK        GET TARGET AREA ADDRESS
L   R1,=A(S99LN)      GET THE LENGTH
LA R14,S99            GET SOURCE AREA ADDRESS
LR R15,R1             GET THE LENGTH
MVCL R0,R14           MOVE IN THE MODEL
LA R1,DYNALWRK        GET PARM AREA ADDRESS
LA R2,S99RB-S99(,R1)  GET RELOCATED S99RB ADDRESS
O   R2,=X'80000000'   SET FLAG
ST   R2,Ø(,R1)       SAVE RELOCATED ADDRESS IN PARMS
LA R2,S99TUPL-S99(,R1) GET RELOCATED S99TUPL ADDRESS
STCM R2,B'1111',S99TXTP-S99(R1) SV RELOCATED S99TUPL ADDR
LA R2,TUØØØ1-S99(,R1) GET RELOCATED TUØØØ1 ADDRESS
STCM R2,B'1111',S99TUPL-S99(R1) SV RELOCATED TUØØØ1 ADDR
LA R2,TUØØØ2-S99(,R1) GET RELOCATED TUØØØ2 ADDRESS
STCM R2,B'1111',TU2-S99(R1) SV RELOCATED TUØØØ2 ADDR
LA R2,TUØØØ3-S99(,R1) GET RELOCATED TUØØØ3 ADDRESS
O   R2,=X'80000000'   SET LAST TU FLAG
STCM R2,B'1111',TU3-S99(R1) SV RELOCATED TUØØØ3 ADDR
SVC 99               ALLOCATE THE INTERNAL READER
LTR R15,R15          ALLOCATE OK?
BZ ALLOCOK           YES - GO ON
MVC WTOPRK2(WTOLN2),WTOLST2 COPY WTO MODEL
ST R15,DL2           SAVE THE RETURN CDOE
UNPK DBL1(9),DBL2(5) UNPACK IT
NC DBL1(8),=8X'ØF'   TURN OFF HIGH ORDER NIBBLES
TR DBL1(8),=C'0123456789ABCDEF' MAKE THINGS READABLE
MVC WTOPRK2+4+54(4),DLB1+4 COPY RETURN CODE
LA R1,DYNALWRK       GET PARM AREA ADDRESS
MVC DBL2(4),S99ERROR-S99(R1) COPY ERROR INFO
UNPK DBL1(9),DBL2(5) UNPACK IT
NC DBL1(8),=8X'ØF'   TURN OFF HIGH ORDER NIBBLES
TR DBL1(8),=C'0123456789ABCDEF' MAKE THINGS READABLE
MVC WTOPRK2+4+59(4),DLB1 COPY ERROR INFO
WTO MF=(E,WTOPRK2)   ISSUE THE WTO
B RETURN             DON'T GO ON
```

**ALLOCOK EQU *  
MVC DCBWRK1(DCBLN1),INTRDR COPY DCB MODEL  
LA R1,DYNALWRK       GET PARM AREA ADDRESS  
MVC DCBWRK1+4Ø(8),DDNAME-S99(R1) COPY DDNAME  
MVC DDNMSAVE(8),DCBWRK1+4Ø COPY DDNAME**
GET DCB ADDRESS
OI OPENLST,X'80'
OPEN ((R8),OUTPUT),MODE=31,MF=(E,OPENLST) OPEN THE INTRDR
TM 48(R8),X'10'
OPENOK YES - KEEP GOING
WTO 'IRREVX01 - internal reader open failed'
B DALLOC DEALLOCATE THE INTERNAL READER

OPENOK EQU *

GET JCL WORK AREA
L R1,=A(JCLLEN)
GET ADDRESS OF JCL MODEL
LA R14,JCL1
COPY LENGTH
LA R7,JCLAREA
GET JCL WORK AREA ADDRESS
LA R1,JCL4-JCL1(,R7)
POINT TO 'PARM=' JCL STATEMENT
LA R1,1(,R1)
POINT PAST 'PARM='
MVI Ø(R1),C''''
SET PARM OPENING QUOTE
LA R1,1(,R1)
POINT PAST QUOTE
L R15,USRIDLEN
GET USERID LENGTH
BCTR R15,Ø
REDUCE BY ONE FOR EX
EX R15,USRIDMV2
COPY THE USERID
LA R1,1(,R15)
ADD ONE BACK TO LENGTH
LA R1,Ø(R15,R1)
POINT PAST USERID
MVI Ø(R1),C','
SET SEPARATOR
LA R1,1(,R1)
POINT PAST COMMA
L R15,UIDBIN
GET UID VALUE
CVD R15,DBL2
CONVERT TO DECIMAL
UNPK DBL1(16),DBL2(8)
UNPACK THE VALUE
OC DBL1(16),=16'F0'
MAKE IT READABLE
MVC Ø(10,R1),DBL1+6
COPY UID TO PARM
LA R1,10(,R1)
POINT PAST UID
MVI Ø(R1),C''''
SET PARM CLOSING QUOTE
LA R6,JCLSTMT#
GET NUMBER OF JCL STATEMENTS
JCLLP EQU *

WRITE THE NEXT JCL STATEMENT
LA R7,80(,R7)
POINT TO NEXT STATEMENT
BCT R6,JCLLP
IF MORE, GO WRITE
OI CLOSELST,X'80'
SET PARM BIT ON
CLOSE (((R8)),MODE=31,MF=(E,CLOSELST) CLOSE THE INTRDR

DALLOC EQU *

GET PARM AREA ADDRESS
LA R1,DYNALWRK
MVI S99VERB-S99(R1),DEALLOC SET DEALLOC VERB
OI S99TUPL-S99(R1),X'80'
SET LAST TEXT UNIT FLAG
XC S99FLAG1-S99(6,R1),S99FLAG1-S99(R1) CLEAR FLAGS
XC S99TXTP+4-S99(6,R1),S99TXTP+4-S99(R1) CLEAR FLAGS
MVC TU0001-S99(2,R1),=X'0001' SET TEXT UNIT TO DDNAME
SVC 99 DEALLOCATE THE INTERNAL READER
LTR R15,R15 DEALLOCATE OK?
BZ DALLOCOK YES - GO ON
MVC WTOWRK3(WTOLN3),WTOLST3 COPY WTO MODEL
ST R15, DBL2              SAVE THE RETURN CODE
UNPK DBL1(9), DBL2(5)       UNPACK IT
NC DBL1(8), =8X'0F'       TURN OFF HIGH ORDER NIBBLES
TR DBL1(8), =C'0123456789ABCDEF' MAKE THINGS READABLE
MVC WTOWRK3+4+56(4), DBL1+4 COPY RETURN CODE
LA R1, DYNALWRK            GET PARM AREA ADDRESS
MVC DBL2(4), S99ERROR-S99(R1) COPY ERROR INFO
UNPK DBL1(9), DBL2(5)       UNPACK IT
NC DBL1(8), =8X'0F'       TURN OFF HIGH ORDER NIBBLES
TR DBL1(8), =C'0123456789ABCDEF' MAKE THINGS READABLE
MVC WTOWRK3+4+61(4), DBL1 COPY ERROR INFO
WTO MF=(E, WTOWRK3)        ISSUE THE WTO
DALLOCOKE EQU * 
.BYPASS_B1 ANOP 
****************************************************************************** 
B RETURN
******************************************************************************
POSTCALL EQU *
B RETURN NO - JUST RETURN
******************************************************************************
RETURN EQU *
LR R1, R13               GET WORKAREA ADDRESS
L R2, SAVEAREA+4         SAVE OLD SAVEAREA ADDRESS
STORAGE RELEASE, LENGTH=WORKLEN, ADDR=(R1)
LR R13, R2               COPY OLD SAVEAREA ADDRESS
LM R14, R12, 12(R13)     RESTORE REGISTERS
XR R15, R15             SET RETURN CODE
BR R14                   RETURN
******************************************************************************
GETUSRID EQU *
ST R14, R14SAVE           SAVE RETURN ADDRESS
L R4, EVXCMBUF            GET COMMAND BUFFER ADDRESS
XR R5, R5                 CLEAR R5
LA R7, 4(R4)             GET COMMAND ADDRESS
ICM R5, B'0011', 0(R4)    GET BUFFER LENGTH
C R5, =F'4'              ANY BUFFER?
BL RETURN NO - WE'RE DONE
S R5, =F'4'             REDUCE BY HEADER LENGTH
******************************************************************************
*  FLUSH LEADING BLANKS
PSTLPØ1 EQU *
CLI Ø(R7), C' '         A BLANK?
BNE PSTEND1 NO - DONE WITH LEADING BLANKS
LA R7, 1(R7)            POINT TO NEXT BUFFER BYTE
BCT R5, PSTLPØ1 IF MORE, GO CHECK
B RETURN WE'RE DONE
PSTEND1 EQU *
LA R7, 1(R7)            SKIP PAST ENCLOSURE
UIDCHK EQU *

*******************************************************************************

UIDCHK EQU *

*******************************************************************************

* The UIDCHK routine scans the command buffer checking for the existence of an OMVS parameter (OM and OMV are synonyms) and a corresponding UID parameter (U and UI are synonyms). If an OMVS UID is detected, its value is parsed for format and if valid, the UID value is extracted and converted to binary.
* On return:
  * R15=0 if a valid OMVS UID has been extracted from the
  * command buffer. In this case, R0 will contain the
  * extracted UID value.
  *
  * R15=4 if an OMVS UID has not been extracted from the command
  * buffer
  *
*---------------------------------------------------------------------*
STM R0,R15,REGSAVE        SAVE REGISTERS
L     R4,EVXCBUF           GET COMMAND BUFFER ADDRESS
XR    R5,R5                 CLEAR R5
ICM   R5,B'Ø011',Ø(R4)      GET BUFFER LENGTH
C     R5,=F'4'              ANY BUFFER?
BL    RETNOUID              NO - WE'RE DONE
LA    R7,Ø(R5,R4)           GET BUFFER END ADDRESS
S     R7,=F'3'              MAKE SURE THERE'S ENOUGH ROOM
ICM   R5,B'Ø011',2(R4)      GET OFFSET OF KEYWORD AREA
LA    R4,4(R5,R4)           GET SEARCH START ADDRESS
**************************************************************************
CLC  Ø(2,R4)=C'OM'        'OMVS' RIGHT OFF THE BAT?
BNE   BUFLP1                NO - START THE PROCESS
LA    R4,2(,R4)             POINT PAST 'OM'
CR    R4,R7                 END OF BUFFER?
BNL   RETNOUID              YES - 'OMVS' NOT DETECTED
B     CHKUID2               GO CHECK FOR UID
BUFLP1 EQU *
CR    R4,R7                 END OF BUFFER?
BNL   RETNOUID              YES - 'OMVS' NOT DETECTED
CLC  Ø(3,R4)=C' OM'        OMVS (OR SOME SHORTFORM)?
BE    CHKUID1               YES - CHECK FOR A UID
LA    R4,1(,R4)             POINT TO NEXT BYTE
B     BUFLP1                GO CHECK IT OUT
CHKUID1 EQU *
LA    R4,3(,R4)             POINT PAST ' OM'
CR    R4,R7                 END OF BUFFER?
BNL   RETNOUID              YES - 'OMVS' NOT DETECTED
CHKUID2 EQU *
CLI  Ø(R4),C' '             A BLANK SEPARATOR?
BE    FLBLNK1                YES - FLUSH BLANKS
CLI  Ø(R4),C'('             OPENING DELIMITER?
BE    DELIM1                 YES - PROCESS DELIMITER
CLI  Ø(R4),C'V'              A 'V'?
BNE   RETNOUID              NO - CMD WON'T BE VALID
LA    R4,1(,R4)             POINT PAST 'V'
CR    R4,R7                 END OF BUFFER?
BNL   RETNOUID              YES - 'OMVS' NOT DETECTED
CLI  Ø(R4),C' '             A BLANK SEPARATOR?
BE    FLBLNK1                YES - FLUSH BLANKS
CLI  Ø(R4),C'('             OPENING DELIMITER?
BE    DELIM1    YES - PROCESS DELIMITER
CLI    Ø(R4),C'S'    A 'S'?
BNE    RETNOUID    NO - CMD WON'T BE VALID

FLBLNK1 EQU  *
LA    R4,1(,R4)    POINT TO NEXT BYTE
CR    R4,R7    END OF BUFFER?
BNL    RETNOUID    YES - 'OMVS' NOT DETECTED
CLI    Ø(R4),C'('    OPENING DELIMITER?
BE    DELIM1    YES - PROCESS DELIMITER
CLI    Ø(R4),C' '    A BLANK?
BE    FLBLNK1    YES - KEEP FLUSHING
B    RETNOUID    NO OMVS SUB-PARAMETERS

DELIM1 EQU  *
LA    R15,1    SET DELIMITER COUNT TO ONE

UIDLP EQU  *
CLC    Ø(2,R4),=C'(U    'UID'?
BE    GOTUID    YES - GO PROCESS
CLC    Ø(2,R4),=C' U'    'UID'?
BE    GOTUID    YES - GO PROCESS
CLI    Ø(R4),C'('    OPEN DELIMITER?
BE    OPENDLIM    YES - ADD ONE
CLI    Ø(R4),C')'    CLOSE DELIMITER?
BE    CLOSDLIM    YES - SUBTRACT ONE

UIDLPØ5 EQU  *
LA    R4,1(,R4)    POINT TO NEXT BYTE
CR    R4,R7    END OF BUFFER?
BNL    RETNOUID    YES - 'OMVS' NOT DETECTED
B    UIDLP    CHECK FROM NEXT BYTE

OPENDLIM EQU  *
LA    R15,1(,R15)    ADD ONE TO DELIMITER COUNT
B    UIDLPØ5    GO CHECK FROM NEXT BYTE

CLOSDLIM EQU  *
BCTR    R15,Ø    SUBTRACT ONE FROM DELIMITER COUNT
LTR    R15,R15    DOWN TO ZERO?
BZ    RETNOUID    YES - END OF OMVS SUB-PARAMETERS
B    UIDLPØ5    GO CHECK FROM NEXT BYTE

GOTUID EQU  *
LA    R4,2(,R4)    POINT PAST ' U' OR '(U'
CR    R4,R7    END OF BUFFER?
BNL    RETNOUID    YES - 'UID' NOT DETECTED
CLI    Ø(R4),C' '    A BLANK SEPARATOR?
BE    FLBLNK2    YES - FLUSH BLANKS
CLI    Ø(R4),C'('    OPENING DELIMITER?
BE    DELIM2    YES - PROCESS DELIMITER
CLI    Ø(R4),C'I'    A 'I'?
BNE    RETNOUID    NO - CMD WON'T BE VALID
LA    R4,1(,R4)    POINT PAST 'I'
CR    R4,R7    END OF BUFFER?
BNL    RETNOUID    YES - 'UID' NOT DETECTED
CLI    Ø(R4),C' '    A BLANK SEPARATOR?
BE FLBLNK2 YES - FLUSH BLANKS
CLI Ø(R4),C(' OPENING DELIMITER?
BE DELIM2 YES - PROCESS DELIMITER
CLI Ø(R4),C'D' A 'D'? 
BNE RETNOUID NO - CMD WON'T BE VALID
FLBLNK2 EQU *
LA R4,1(,R4) POINT TO NEXT BYTE
CR R4,R7 END OF BUFFER?
BNL RETNOUID YES - 'OMVS' NOT DETECTED
CLI Ø(R4),C(' OPENING DELIMITER?
BE DELIM2 YES - PROCESS DELIMITER
CLI Ø(R4),C' A BLANK? 
BE FLBLNK2 YES - KEEP FLUSHING
B RETNOUID NO OMVS SUB-PARAMETERS
DELIM2 EQU *
LA R4,1(,R4) POINT TO NEXT BYTE
CR R4,R7 END OF BUFFER?
BNL RETNOUID YES - 'OMVS' NOT DETECTED
CLI Ø(R4),C'D' A 'D'? 
BNE UIDVAL NO - MUST BE THE VALUE
B DELIM2 CHECK NEXT BYTE
UIDVAL EQU *
XR R14,R14 CLEAR UID VALUE LEN COUNTER
MVC UIDAREA(10),=10C' ' CLEAR TARGET AREA
LA R8,UIDAREA GET TARGET AREA ADDRESS
UIDVALLP EQU *
CLI Ø(R4),C' END OF VALUE?
BE VALEND YES - CHECK THINGS OUT
CLI Ø(R4),C'9' END OF VALUE?
BE VALEND YES - CHECK THINGS OUT
CLI Ø(R4),C'Ø' A VALID NUMBER?
BL RETNOUID NO - COMMAND WON'T SUCCEED
CLI Ø(R4),C'9' A VALID NUMBER?
BH RETNOUID NO - COMMAND WON'T SUCCEED
MVC Ø(1,R8),Ø(R4) COPY NEXT BYTE OF VALUE
LA R14,1(,R14) ADD ONE TO LEN COUNT
LA R8,1(,R8) POINT TO NEXT TARGET BYTE
LA R4,1(,R4) POINT TO NEXT BYTE
CR R4,R7 END OF BUFFER?
BNL RETNOUID YES - 'OMVS' NOT DETECTED
B UIDVALLP CHECK NEXT BYTE
VALEND EQU *
C R14,=F'10' VALUE LENGTH OK?
BH RETNOUID NO - COMMAND WON'T SUCCEED
LTR R14,R14 VALUE LENGTH OK?
BZ RETNOUID NO - COMMAND WON'T SUCCEED
C R14,=F'10' VALUE LENGTH IS 10?
BL VALOK NO - VALUE IS OK
CLC UIDAREA(10),=C'2147483647' VALUE IS OK?
BH RETNOUID NO - COMMAND WON'T SUCCEED
VALOK    EQU   *
XR    R15,R15               CLEAR R15
XR    R9,R9                 CLEAR R9
LA    R8,UIDAREA            GET UID AREA ADDRESS
VALLP   EQU   *
IC    R15,Ø,(R8)            GET NEXT NUMBER
N     R15,=X'0000000F'      TURN OFF ALL BUT LOW ORDER NIBBLE
MH    R9,=H'10'             MUPLY BASE BY 10
AR    R9,R15                ADD IN NEW VALUE
LA    R8,1,(R8)             POINT TO NEXT BYTE
BCT   R14,VALLP             GO PROCESS
LR    R0,R9                 COPY UID VALUE TO R0
RETTUID  EQU   *
LR    R1,R4                 SAVE BUFFER ADDRESS
LM    R2,R14,REGSAVE+8      RESTORE SOME REGISTERS
XR    R15,R15               SET RETURN CODE TO 0
BR    R14                   RETURN
RETNOUID EQU   *
LM    R2,R14,REGSAVE+8      RESTORE SOME REGISTERS
LA    R15,4                 SET RETURN CODE TO 4
BR    R14                   RETURN

***********************************************************************
*                                                                     *
*   EXECUTED INSTRUCTIONS                                             *
*                                                                     *
***********************************************************************
USRIDMVC MVC   USRIDSAV(*-*),Ø(R8)   COPY IN THE USERID
USRIDMV2 MVC   Ø(*-*,R1),USRIDSAV COPY IN THE USERID

***********************************************************************
*                                                                     *
*   CONSTANTS                                                         *
*                                                                     *
***********************************************************************
WTOLST   WTO   'IRREVXØ1 - Specified UID(nnnnnnnnnn) for userid xxxxxxxX
x previously assigned to userid xxxxxxxx.               X
',MF=L
WTOLN   EQU   *.-WTOLST
***********************************************************************
WTOLST2 WTO   'IRREVXØ1 - Allocation failed for internal reader - rc xX
xxx-xxxx','MF=L
WTOLN2  EQU   *.-WTOLST2
***********************************************************************
WTOLST3 WTO   'IRREVXØ1 - Deallocation failed for internal reader - rcX
xxx-xxxx','MF=L
WTOLN3  EQU   *.-WTOLST3

***********************************************************************
FLDLIST1 DC    F'1'
DC    CL8'PGMRNAME'
***********************************************************************
FLDLIST2 DC    F'1'

DC CL8'UID ',
***********************************************************************
RACROUT1 RACROUTE REQUEST=EXTRACT,
   TYPE=EXTRACTN,  X
   CLASS='USER',   X
   RELEASE=1.9.2,  X
   MF=L
ROUTLEN1 EQU   *-RACROUT1
***********************************************************************
RACROUT2 RACROUTE REQUEST=EXTRACT,
   TYPE=EXTRACT,  X
   CLASS='USER',   X
   SEGMENT='OMVS', X
   RELEASE=1.9.2,  X
   MF=L
ROUTLEN2 EQU   *-RACROUT2
***********************************************************************
*   DYNAMIC ALLOCATION PARAMETER AREA MODEL
S99      DC   A(X'80000000'+S99RB)
S99RB    DC   X'14'
S99VERB  DC   X'01'
ALLOC    EQU  X'01'
DEALLOC  EQU  X'02'
S99FLAG1 DC   X'0000'
S99ERROR DC   X'0000'
S99INFO  DC   X'0000'
S99TXTP  DC   AL4(S99TUPL)
   XL4'00'
S99FLAG2 DC   XL4'00'
S99TUPL  DC   AL4(TUØØØ1)
TU2      DC   AL4(TUØØØ2)
TU3      DC   AL4(X'80000000'+TUØØØ3)
* //TUØØØ1 DD SYSOUT=(TUØØØ2,TUØØØ3)
   DDNAME   DC   CL8'        '
   DD      DC   X'0055',X'0001',X'0008'   // DD
   DNAME   DC   CL8'        '
TUØØØ2 DC   X'0018',X'0001',X'0001',C'A'    SYSPRT=(A,
   TUØØØ3 DC   X'0019',X'0001',X'0008',C'INTRDR'    INTRDR)
S99LN    EQU  *-S99
***********************************************************************
*   BACKGROUND BATCH JOB JCL MODEL
JCL1     DC    CL8Ø'//jobname  JOB (acct#),'admin',MSGCLASS=O,'
JCL2     DC    CL8Ø'//         CLASS=A,NOTIFY=admin,MSGLEVEL=(1,1)'  
JCL3     DC    CL8Ø'//STEP1  EXEC PGM=UIDXREFB,'  
JCL4     DC    CL8Ø'/ PARM='  
JCL5     DC    CL8Ø'/STEPLIB DD  DSN=auth.load.library,DISP=SHR'  
JCL6     DC    CL8Ø'/SYSPRINT DD  SYSOUT=*'  
JCL7     DC    CL8Ø'/*'  
JCLLEN   EQU  *-JCL1  
JCLSTM#  EQU  JCLLEN/80
***********************************************************************
INTRDR  DCB  MACRF=(PM),DDNAME=INTRDR,LRECL=80,DSORG=PS
DCBLN1  EQU  *
***********************************************************************
LTORG
WORKAREA DSECT
SAVEAREA DS   18F
REGSAVE DS   18F
R14SAVE DS   F
RETCODE DS   F
UIDAREA DS   CL10
XUID DS   F
USERID DS   CL8
USRIDSAV DS   CL8
USRIDLEN DS   F
DDNMSAVE DS   CL8
WTOWRK DS   0D,CL(WTOLN)
WTOWRK2 DS   0D,CL(WTOLN2)
WTOWRK3 DS   0D,CL(WTOLN3)
ROUTWRK1 DS   0D,CL(ROUTLEN1)
ROUTWRK2 DS   0D,CL(ROUTLEN2)
JCLAREA DS   0D,CL(JCLLEN)
DYNALWRK DS   0D,CL(S99LN)
DCBWRK1 DS   0D,CL(DCBLN1)
OPENLST OPEN (,),MODE=31,MF=L
CLOSELST CLOSE (,),MODE=31,MF=L
DBL1 DS   2D
DBL2 DS   2D
UIDBIN DS   F
EXTUID DS   F
RACWORK DS   0D,CL(512)
WORKLEN EQU  *-WORKAREA
IRREVXP
IRRPRXTW
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
END
UIDXREFB ASSEMBLER

UIDXREFB CSECT
UIDXREFB AMODE 31
UIDXREFB RMODE ANY

*---------------------------------------------------------------*
* UIDXREFB is designed to be used in conjunction with the UIDXREFX  *
* version of IRREVX$1 for RACF. UIDXREFB is a background batch *
* job that examines the RACF database for conflicts in use of the *
* specified OMVS UID. Any conflicts that are detected are        *
* identified in the SYSPRINT output dataset. As well, if any    *
* conflicts are detected, this program will produce a table of   *
* available OMVS UIDs.                                          *
* *This program is called with one program PARM that has the     *
* following basic format:                                       *
* PARM='userid,omvsuid'
* *where 'userid' is the userid detected in the ADDUSER or ALTUSER *
* command and 'omvsuid' is the OMVS UID that was parsed out of   *
* the ADDUSER or ALTUSER command buffer.                        *
* *This program requires SETCODE AC(1) on the linkedit and it must *
* reside in an APF authorized library. Unless the program name   *
* is changed in the model JCL in the UIDXREFX exit program, the *
* name of the resulting load module should be UIDXREFB.          *
*---------------------------------------------------------------*

STM   R14,R12,12(R13)       SAVE INCOMING REGISTERS
LR    R12,R15               COPY MODULE ADDRESS
USING UIDXREFB,R12           SET ADDRESSABILITY
LR    R2,R1                 SAVE INCOMING PARM ADDRESS
LR    R11,R13               SAVE OLD SAVEAREA ADDRESS
STORAGE OBTAIN,LENGTH=WORKLEN,LOC=BECLOW
LR    R13,R1                GET NEW SAVEAREA ADDRESS
LR    R0,R1                 COPY ADDRESS
LR    R14,R1                AGAIN
L     R1,=A(WORKLEN)        GET LENGTH
XR    R15,R15               SET FILL BYTE
MVCL  R0,R14                CLEAR THE STORAGE
USING WORKAREA,R13           SET ADDRESSABILITY
ST    R11,SAVEAREA+4        SAVE OLD SAVEAREA ADDRESS

**************************************************************************
L     R9,Ø(R2)             POINT TO INCOMING PARM DATA LEN
LA    R3,2(,R9)             POINT TO START OF USERID
LA    R1,USRIDSAV           GET USERID AREA ADDRESS
MVC   USRIDSAV(8),=8C' '    CLEAR THE TARGET AREA
SAVELP  EQU   *
CLI   Ø(R3),C',,'         THE SEPARATOR?
BE    SAVEEND          YES - DONE WITH THE USERID
MVC Ø(1,R1),Ø(R3)  
COPY NEXT BYTE OF USERID
LA R1,1(,R1)  
POINT TO NEXT TARGET BYTE
LA R3,1(,R3)  
POINT TO NEXT SOURCE BYTE
B SAVELP  
CHECK FOR MORE
SAVEEND EQU *
LA R3,1(,R3)  
POINT TO UID VALUE
MVC UIDAREA(10),Ø(R3)  
SAVE UID VALUE
PACK DBL1(Ø),UIDAREA(10)  
PACK THE UID
CVB R15,DBL1  
CONVERT TO BINARY
ST R15,UIDBIN  
SAVE BINARY UID
***********************************************************************
MVC DCBWRK1(DCBLN1),SYSPRINT COPY DCB MODEL
LA R1Ø,DCBWRK1  
GET DCB ADDRESS
OI OPENLST,X'8Ø'  
SET PARM BIT ON
OPEN ((R1Ø),OUTPUT),MODE=31,MF=(E,OPENLST) OPEN THE INTRDR
TM 48(R1Ø),X'1Ø'  
OPEN SUCCESSFUL?
BO OPENOK  
YES - KEEP GOING
WTO 'UIDXREFB - open failed'
B RETURN  
DONE
OPENOK EQU *
***********************************************************************
MVI FLAG,X'Ø'  
CLEAR FLAG
XC XUID(4),XUID  
CLEAR LENGTH AREA
MVC XUID(2),=H'8'  
SET LENGTH
MVC USERID(8),=8C' '  
SET STARTING USER ID VALUE
***********************************************************************
USERIDLP EQU *
XC RACWORK(256),RACWORK
XC RACWORK+256(256),RACWORK+256
MVC ROUTWRK1(ROUTLEN1),RACROUT1
RACROUTE REQUEST=EXTRACT,  
TYPE=EXTRACTN,  
ENTITY=XUID,  
RELEASE=1.9.2,  
FIELDS=FLDLIST1,  
SUBPOOL=1,  
WORKA=RACWORK,MF=(E,ROUTWRK1)
LTR R15,R15  
EXTRACT OK?
BNZ CHKLIST  
NO - BUG OUT
LR R6,R1  
COPY THE EXTRACT AREA ADDRESS
***********************************************************************
XR R8,R8  
CLEAR R8
XR R9,R9  
CLEAR R9
IC R9,Ø(,R6)  
SAVE THE SUBPOOL VALUE
ICM R8,B'Ø111',1(R6)  
SAVE W/A LENGTH
STORAGE RELEASE,LENGTH=(R8),ADDR=(R6),SP=(R9)
***********************************************************************
XC RACWORK(256),RACWORK
XC RACWORK+256(256),RACWORK+256
MVC ROUTWRK2(ROUTLEN2),RACROUT2
RACROUTE REQUEST=EXTRACT,  
  TYPE=EXTRACT,  
  ENTITY=USERID,  
  RELEASE=1.9.2,  
  FIELDS=FLDLIST2,  
  SUBPOOL=1,  
  WORKA=RACWORK, MF=(E, ROUTWRK2)

LTR   R15, R15           OMVS SEGMENT?
BNZ   USERIDLP           NO - CHECK NEXT USERID

***********************************************************************
USING EXTWKEA, R6           EXTRACT WORKAREA ADDRESSABILITY
LR    R6, R1                GET EXTRACT WORKAREA
XR    R8, R8                CLEAR R8
XR    R9, R9                CLEAR R7
IC    R9, Ø(, R6)           GET SUBPOOL
ICM   R8, B'Ø111', 1(R6)    GET LENGTH
XR    R15, R15              CLEAR R15
ICM   R15, B'ØØ11', EXTWOFF GET OFFSET OF DATA AREA
AR    R15, R6               POINT TO UID AREA
ICM   R14, B'1111', Ø(R15)  GET UID LENGTH
MVC   EXTUID(4), 4(R15)     COPY EXTRACT UID
STORAGE RELEASE, LENGTH=(R8), ADDR=(R6), SP=(R9)

***********************************************************************
CLC   EXTUID(4), =4X'FF'     RESERVED UID?
BE    USERIDLP              YES - GET NEXT USERID

***********************************************************************
CLC   UIDLIST(4), =F'Ø'     A LIST YET?
BNE   LISTADD               YES - CHECK WHERE TO ADD IN
STORAGE OBTAIN, LENGTH=UIDENTLN, LOC=ANY
USING UIDENTRY, R1
XC    UIDENXT(4), UIDENXT   CLEAR NEXT ENTRY POINTER
MVC   UIDEUID(4), EXTUID    COPY UID
B     UIDCHK                CHECK FOR A UID MATCH
DROP  R1

LISTADD EQU *
L     R7, UIDLIST           GET LIST START ADDRESS
LA    R8, UIDLIST           GET ADDRESS OF LIST SAVE LOCATION
USING UIDENTRY, R7

ADDCHK EQU *
CLC   EXTUID(4), UIDEUID    EXTRACTED UID LESS THAN LIST ENT?
BL    NEWENT                YES - ADD A NEW ENTRY
CLC   EXTUID(4), UIDEUID    EXTRACTED UID SAME AS LIST ENT?
BE    SAMEENT               YES - ADD TO ENTRY
LR    R8, R7                SAVE PREV ENTRY ADDRESS
L     R7, UIDENXT           GET NEXT ENTRY ADDRESS
LTR   R7, R7                END OF LIST?
BZ    NEWENT                YES - ADD A NEW ENTRY
B     ADDCHK                CHECK AGAINST NEXT ENTRY
DROP  R7
NEWENT EQU *
ST R1,Ø(R8) SAVE ADDRESS IN PREV ENTRY
USING UIDENTRY,R1
ST R7,UIDENTX SAVE CURRENT AS NEXT
MVC UIDEUID(4),EXTUID COPY UID
B UIDCHK CHECK FOR A UID MATCH
DROP R1
SAMEENT EQU *
B UIDCHK CHECK FOR A UID MATCH

***********************************************************************
UIDCHK EQU *
CLC EXTUID(4),UIDBIN SAME UID?
BNE USRIDLP NO - GET NEXT USERID
CLC USRIDSAV(8),USERID SAME USERID?
BE USRIDLP YES - GET NEXT USERID
OI FLAG,UIDCONF SET UID CONFLICT FLAG
MVC OUTREC(133),OUTPUT2 COPY OUTPUT RECORD MODEL
MVC OUTREC+37(8),USRIDSAV COPY USERID
MVC OUTREC+76(8),USERID COPY USERID
L R15,EXTUID GET UID VALUE
CVD R15, DBL2 CONVERT TO DECIMAL
UNPK DBL1(16), DBL2(8) UNPACK THE VALUE
OC DBL1(16),=16'FØ' MAKE IT READABLE
MVC OUTREC+14(1Ø),DBL1+6 COPY UID
PUT (R1Ø),OUTREC WRITE CONFLICT RECORD
B USRIDLP GET NEXT USERID

***********************************************************************
CHKLIST EQU *
TM FLAG,UIDCONF A UID CONFLICT?
BO CHKLIST1 YES - KEEP GOING
MVC OUTREC(133),OUTPUT1 COPY OUTPUT RECORD MODEL
MVC OUTREC+27(8),USRIDSAV COPY USERID
L R15,UIDBIN GET UID VALUE
CVD R15, DBL2 CONVERT TO DECIMAL
UNPK DBL1(16), DBL2(8) UNPACK THE VALUE
OC DBL1(16),=16'FØ' MAKE IT READABLE
MVC OUTREC+44(1Ø), DBL1+6 COPY UID
PUT (R1Ø),OUTREC WRITE OUTPUT RECORD
B CLOSE ALL DONE

CHKLIST1 EQU *
MVI OUTREC,C ' ' SET FILL BYTE
MVC OUTREC+1(132),OUTREC BLANK IT OUT
PUT (R1Ø),OUTREC WRITE OUTPUT RECORD
XR R3,R3 SET STARTING UID TO ZERO
L R2,UIDENTLIST GET LIST START ADDRESS
LTR R2,R2 ANY ENTRIES?
BZ NOENTS NO - PROBABLY NOT POSSIBLE, BUT...
USING UIDENTRY,R2
CLC UIDEUID(4),=FØ' FIRST USED UID IS ZERO?
BE UIDLOOP  YES - GET RIGHT INTO THINGS
L  R4,UIDEUID  GET FIRST USED UID
BCTR R4,Ø  SUBTRACT ONE
CVD R4,DBL2  CONVERT TO DECIMAL
UNPK DBL1(16),DBL2(8)  UNPACK THE VALUE
OC DBL1(16),=16X'FØ'  MAKE IT READABLE
MVC OUTREC(133),OUTPUT3  COPY OUTPUT RECORD MODEL
MVC OUTREC+3Ø(1Ø),DBL1+6  COPY END OF RANGE UID
PUT (R1Ø),OUTREC  WRITE OUTPUT RECORD

UIDLOOP EQU *
L  R7,UIDENXT  GET ADDRESS OF NEXT ENTRY
LTR R7,R7  A NEXT ENTRY?
BZ LASTENT  NO - NOTHING MORE TO DO
L  R4,UIDEUID  GET USED UID
L  R5,UIDEUID-UIDENTRY(,R7) GET UID FOR NEXT ENTRY
SR  R5,R4  SUBTRACT THE TWO UIDS
C  R5,=F'1'  DIFFERENCE MORE THAN ONE?
BNH NEXTUID  NO - GET NEXT ENTRY
MVC OUTREC(133),OUTPUT3  COPY OUTPUT RECORD MODEL
LA R4,1(,R4)  ADD ONE TO LOW UID RANGE VALUE
CVD R4,DBL2  CONVERT TO DECIMAL
UNPK DBL1(16),DBL2(8)  UNPACK THE VALUE
OC DBL1(16),=16X'FØ'  MAKE IT READABLE
MVC OUTREC+17(1Ø),DBL1+6  COPY START OF RANGE UID
L  R4,UIDEUID-UIDENTRY(,R7) GET UID FOR NEXT ENTRY
BCTR R4,Ø  SUBTRACT ONE FROM HI UID RANGE VAL
CVD R4,DBL2  CONVERT TO DECIMAL
UNPK DBL1(16),DBL2(8)  UNPACK THE VALUE
OC DBL1(16),=16X'FØ'  MAKE IT READABLE
MVC OUTREC+3Ø(1Ø),DBL1+6  COPY END OF RANGE UID
PUT (R1Ø),OUTREC  WRITE OUTPUT RECORD

NEXTUID EQU *
L  R2,UIDENXT  GET ADDRESS OF NEXT ENTRY
B  UIDLOOP  CHECK IT OUT

LASTENT EQU *
CLC UIDEUID(4),UIDL  LAST POSSIBLE UID?
BNL CLOSE  YES - WE'RE DONE
L  R4,UIDEUID  GET LAST USED UID
LA R4,1(,R4)  ADD ONE
CVD R4,DBL2  CONVERT TO DECIMAL
UNPK DBL1(16),DBL2(8)  UNPACK THE VALUE
OC DBL1(16),=16X'FØ'  MAKE IT READABLE
MVC OUTREC(133),OUTPUT3  COPY OUTPUT RECORD MODEL
MVC OUTREC+17(1Ø),DBL1+6  COPY START OF RANGE UID
PUT (R1Ø),OUTREC  WRITE OUTPUT RECORD
B  CLOSE  WE'RE ALL DONE

NOENTS EQU *
PUT (R1Ø),OUTPUT3  WRITE OUTPUT RECORD
B  CLOSE  WE'RE ALL DONE
DROP R2
CLOSE EQU *
OI CLOSELST,X'80' SET PARM BIT ON
CLOSE ((R1Ø)),MODE=31,MF=(E,CLOSELST) CLOSE THE INTRDR
***********************************************************************
B RETURN
***********************************************************************
RETURN EQU *
LR R1,R13 GET WORKAREA ADDRESS
L R2,SAVEAREA+4 SAVE OLD SAVEAREA ADDRESS
STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
LR R13,R2 COPY OLD SAVEAREA ADDRESS
LM R14,R12,12(R13) RESTORE REGISTERS
XR R15,R15 SET RETURN CODE
BR R14 RETURN
***********************************************************************
* * CONSTANTS * *
************************************************************************
FLDLIST1 DC F'1' DC CL8'PGMRNAME'
************************************************************************
FLDLIST2 DC F'1' DC CL8'UID'
************************************************************************
RACROUT1 RACROUTE REQUEST=EXTRACT, TYPE=EXTRACTN, CLASS='USER', RELEASE=1.9.2, MF=L
ROUTLEN1 EQU *-RACROUT1
************************************************************************
RACROUT2 RACROUTE REQUEST=EXTRACT, TYPE=EXTRACT, CLASS='USER', SEGMENT='OMVS', RELEASE=1.9.2, MF=L
ROUTLEN2 EQU *-RACROUT2
************************************************************************
SYSPRINT DCB MACRF=(PM),DDNAME=SYSPRINT,LRECL=133,DSORG=PS
DCBLN1 EQU *-SYSPRINT
************************************************************************
OUTPUT1 DC CL133'No UID conflict for userid xxxxxxxx and UID '
OUTPUT2 DC CL133'Specified UID(nnnnnnnnnn) for userid xxxxxxxx previouly assigned to userid xxxxxxxx.'
OUTPUT3 DC CL133'Available UIDS: ØØØØØØØØØØ - 2147483647'
************************************************************************
UIDF DC F'0'
UIDL DC X'7FFFFFFF'
************************************************************************
LTORG
WORKAREA DSECT
SAVEAREA DS 18F
REGSAVE DS 18F
R14SAVE DS F
RETCODE DS F
UIDAREA DS CL10
XUID DS F
USERID DS CL8
USRIDSAV DS CL8
USRIDLEN DS F
ROUTWRK1 DS ØD,CL(ROUTLEN1)
ROUTWRK2 DS ØD,CL(ROUTLEN2)
DCBWRK1 DS ØD,CL(DCBLN1)
OPENLST OPEN (,),MODE=31,MF=L
CLOSELST CLOSE (,),MODE=31,MF=L
DBL1 DS 2D
DBL2 DS 2D
UIDBIN DS F
EXTUID DS F
UIDLIST DS F
FLAG DS X
UIDCONF EQU X'80'
OUTREC DS CL133
RACWORK DS ØD,CL(512)
WORKLEN EQU *-WORKAREA
UIDENTRY DSECT
UIDENXT DS F
UIDEUID DS F
UIDENTLN EQU *-UIDENTRY

IRRPRTXTW ,
R0 EQU 0
R1 EQU 1
R2 EQU 2
R3 EQU 3
R4 EQU 4
R5 EQU 5
R6 EQU 6
R7 EQU 7
R8 EQU 8
R9 EQU 9
R10 EQU 10
R11 EQU 11
R12 EQU 12
R13 EQU 13
R14 EQU 14
R15 EQU 15
END
RACF 101 is a regular column for newcomers to the RACF world. It presents basic RACF topics in a tutorial format. This issue’s column answers some commonly asked RACF questions.

Q I would like to provide a user with UPDATE access to a sensitive RACF profile, but only on a temporary basis. How do I do this in RACF? (I heard that ACF2 has a way to do this.)

A You are right, ACF2 does have a direct way to do this. We need an automated method to revoke a user’s access to a profile on a certain date. RACF does not have a direct way to do this, but you can achieve the functionality indirectly, as follows:

Create a temporary RACF group:

\`\`ADDGROUP TEMPGRP SUPGROUP(...) OWNER(...)\`

Connect the user to this group using the UNTIL parameter:

\`\`CONNECT USER123 GROUP(TEMPGRP) UNTIL(01/01/2006)\`

Permit this group to the sensitive profile with UPDATE access:

\`\`PERMIT 'PAYROLL.**' ID(TEMPGRP) ACCESS(UPDATE)\`

This will have the desired effect. On 01/01/2006, the user USER123 will no longer be connected to the group TEMPGRP, and will therefore lose his update access to the sensitive profile.

Q I have heard of the term ‘segregation of duties’ from an auditor. What does this mean?

A This, in security terms, means that no single person should have more than one powerful RACF privilege. In other words, powerful privileges should be segregated
among different people, to provide for greater accountability and separation of duties. For example, a systems programmer may need the OPERATIONS attribute to access data during off-hours – if you give this privilege, make sure he does not have the SPECIAL attribute as well. Conversely, if the RACF administrator has the SPECIAL attribute to perform her duties, make sure she does not have the OPERATIONS attribute at the same time. Likewise, a person with AUDITOR attribute should not have either of the other two.

Q I have often been told that RACF exits override normal RACF processing. How do I find out whether our installation has implemented RACF exits, and if it has, which ones? (I thought SETROPTS LIST command would show the exits, but it does not.)

A Run the DSMON (Data Security Monitor) report to see whether there are any exits in place at your installation.

The following JCL will produce the desired report:

```
//STEP1     EXEC  PGM=ICHDSM00
//SYSPRINT  DD    SYSOUT=A
//SYSUT2    DD    SYSOUT=A
//SYSSIN    DD    SYSOUT=A
//SYSSIN    DD    SYSOUT=A
SYSPRINT
SYSSIN
* RACEXT
/*
```

Q When I enter the command SETROPTS LIST at our installation, there is one line that concerns me – it says, ‘AUTOMATIC DATASET PROTECTION IS NOT IN EFFECT’. Is this something I should worry about?

A No. In fact, that’s how it is supposed to be. The line is there for historical reasons. Long ago, when discrete profiles were around, you needed to enable ‘automatic dataset protection’ to protect your datasets. Nowadays, datasets are automatically protected by generic profiles, not by means of this facility.

Q I have the userid IBMUSER on my system. I have heard this userid was needed to set up the initial RACF system,
but is not needed any more. Since it is a powerful ID, should I delete it to prevent its misuse?

A No, you cannot delete IBMUSER. It is supplied as part of RACF and cannot be deleted. But you are right, it is a powerful userid, and should not be left exposed. If you are satisfied that you have enough userid with the SPECIAL privilege to cover all sorts of emergencies, you can do the following:

ALTUSER IBMUSER REVOKE
ALTUSER IBMUSER NOSPECIAL NOOPERATIONS NOAUDITOR

This will revoke the userid and strip it of its special powers, so it will be harmless even if someone resumes the userid by mistake.

Q What does the TSO ACCOUNT command do? I have been told it is a sensitive command, but do not know why.

A The TSO ACCOUNT command is used to administer the userids contained in the SYS1.UADS dataset. SYS1.UADS was the ‘old’ way of controlling passwords and access to TSO, back in the days when RACF was not invented. Nowadays, of course, RACF is used to control access to TSO. But SYS1.UADS is still used for disaster recovery situations when RACF may not be fully operational. So it is important to protect the TSO ACCOUNT command, since it determines who can use TSO during disaster recovery. Typically, the systems programmers should have access to the TSO ACCOUNT command, so they can set up the disaster recovery userids and administer them.

Q Before I delete a RACF userid, I would like to take a copy of the userid, so I can restore it if the need arises. What is the best way to do this?

A You can use a batch program to do this. Simply run the JCL, and all the attributes, group connections, etc of the userid you are about to delete will be saved in a PDS member. The only thing you will need to change in the JCL
before you run it is the userid in two places (in this example USER678).

Here is the JCL:

//RACFJOB1  JOB 1,'YOUR NAME',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
//STEP01 EXEC PGM=IKJEFT01,REGION=1M
//SYSTSPRT DD DSN=YOUR.PDS.NAME(USER678),DISP=SHR
//SYSTSIN  DD *
LISTUSER USER678
/*

Q When I see a RACF dataset profile, for example, PAYROLL.**, I would like to know which datasets it protects at our installation. How do I find out?

A We would all like to know which datasets a profile protects – perhaps it doesn’t protect any, in which case we would like to delete the profile.

The command:

LISTDSD DA('PAYROLL.**') DSNS

will list all datasets that this profile covers.

You may want to use this command to determine whether you need more granular profiles for sensitive datasets. For example, after using the above command, you may find that you have several important datasets, all of which start with PAYROLL.PROD. In this case, you may want to create a more granular profile called PAYROLL.PROD.** with its own, more restrictive, access list, which will override the profile PAYROLL.**.

A word of caution – the command shows only catalogued datasets! These days, almost all datasets are catalogued, so this is not a big concern. Also, this command applies only to dataset profiles, not other general resource profiles.

Q Conversely, if I have a dataset, for example SYS1.LINKLIB, I would like to know which RACF profile protects it. How do I do that?

A Enter the following command and it will show you the
Checking resource profiles for orphaned IDs

Whether a user can access RACF-defined resources is determined by four main factors:

1. The access level attributes assigned to the userid (RACF SPECIAL, for example).
2. The authority level of any group the userid may be connected to.
3. The userid ownership for a given resource.
4. The access privileges granted to a userid through a resource’s standard or conditional Access Control List (ACL).

When attempting to access a specific resource, RACF will determine whether the requesting userid has sufficient authority and deny or grant access accordingly.

A potential hole exists in this process. When a userid is deleted with the DELUSER command, or when a group ID is...
deleted with the DELGROUP command, RACF does not cross-reference the resource profiles to determine all the resources an ID may own or the profile ACLs to which an ID may be permitted. If an ID is deleted that owns a resource profile (non-dataset), or if the ID resides in a resource profile’s ACL, the resource profile contains an orphaned ID, either as the profile owner or in the profile’s ACL. The hole that exists occurs when that ID is re-used. The re-used ID will assume any remnant permission assignments associated with the prior existence of the ID.

To identify these conditions, a utility such as the RACFXREF utility described in this article can be invaluable.

HOW RACFXREF WORKS

RACFXREF examines every RACF dataset profile (both discrete and generic) and every general resource profile (again, both discrete and generic). For each profile that is examined, RACFXREF checks whether the owning ID exists as either a RACF userid or a RACF group ID. If the owning ID does not exist as a userid or a group ID, RACFXREF reports an orphaned ID condition. Also, all IDs that reside in each profile’s standard and conditional ACLs are checked to see whether they exist. If any ID is found to not exist as a userid or group ID, RACFXREF reports that as an orphaned ID condition.

The following is some sample output from RACFXREF:

<table>
<thead>
<tr>
<th>Orphaned ID</th>
<th>Type</th>
<th>Condition</th>
<th>CLASS</th>
<th>Profile</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSTUSR1</td>
<td>Gen</td>
<td>Standard</td>
<td>DATASET</td>
<td>DEVEL.TEST.LOAD</td>
</tr>
<tr>
<td>TSTUSR3</td>
<td>Gen</td>
<td>TAPELBL</td>
<td>DATASET</td>
<td>SYSPROG.UTILS</td>
</tr>
<tr>
<td>TSTUSR2</td>
<td>Dis</td>
<td>TESTPGM</td>
<td>DATASET</td>
<td>TESTING.LOAD.LIBRARY</td>
</tr>
<tr>
<td>TSTUSR4</td>
<td>Dis</td>
<td>Owner</td>
<td>FACILITY</td>
<td>DELETE.TEST.PROFILE</td>
</tr>
<tr>
<td>P39ØA</td>
<td>Dis</td>
<td>Standard</td>
<td>TSOPROC</td>
<td>DBSPROC</td>
</tr>
<tr>
<td>P39ØB</td>
<td>Dis</td>
<td>Standard</td>
<td>TSOPROC</td>
<td>ISPFPROC</td>
</tr>
<tr>
<td>P39ØC</td>
<td>Dis</td>
<td>Standard</td>
<td>TSOPROC</td>
<td>ISPFPROC</td>
</tr>
<tr>
<td>TESTEG</td>
<td>Dis</td>
<td>Standard</td>
<td>TSAUTH</td>
<td>RECOVER</td>
</tr>
<tr>
<td>RACUID</td>
<td>Dis</td>
<td>Standard</td>
<td>FIELD</td>
<td>USER.OMVS.HOME</td>
</tr>
</tbody>
</table>
As can be seen, RACFXREF reports orphaned IDs, the RACF class and profile where the orphaned ID was detected, the profile type (either ‘Dis’ for discrete or ‘Gen’ for generic), and the condition under which the orphaned ID was detected (‘Owner’ if the ID was detected as the profile’s owner, ‘Standard’ if the ID was detected in the profile’s standard ACL, or an upper-case value that indicates the specific conditional ACL if the ID was detected in a profile’s conditional ACL).

To ‘clean up’ the orphaned ID conditions similar to the above, RACF commands such as the following could be considered:

```
PERMIT DEVEL.TEST.LOAD CLASS(DATASET) ID(TSTUSR1) DELETE

PERMIT SYSPROG.UTILS CLASS(DATASET) ID(TSTUSR3) WHEN(PROGRAM(TAPELBL)) DELETE

RALTER FACILITY (DELETE.TEST.PROFILE) OWNER(NEWUSR4)

PERMIT RECOVER CLASS(TSOAUTH) ID(TESTEG) DELETE
```

The first command deletes an orphaned ID from the standard ACL of a *DATASET* profile. The second command deletes an orphaned ID from the conditional ACL of a *DATASET* profile. The third command changes the ownership of a *FACILITY* profile. The fourth command deletes an orphaned ID from the standard ACL of a TSOAUTH profile.

**PREPARING RACFXREF FOR ACTION**

Assemble RACFXREF with a standard assembly job that includes SYS1.MACLIB and SYS1.MODGEN in the SYSLIB dataset concatenation. Here’s sample JCL to linkedit RACFXREF:

```
//IEWL     EXEC  PGM=HEWLH96,PARM='XREF,LIST,MAP'
//SYSPRINT DD    SYSOUT=* 
//SYSTUT1 DD    UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT DD    DSN=object.code.pds,DISP=SHR
//SYSLMOD DD    DSN=apf.auth.library,DISP=SHR
//SYSLIN DD    *
    INCLUDE OBJECT(RACFXREF)
    ENTRY RACFXREF
    SETCODE AC(1)
    NAME RACFXREF(R)
```
USING THE RACFXREF UTILITY

When the RACFXREF load module has been created, the utility is ready to use. To examine a system’s active RACF database, run the following JCL:

```
//RACFXREF EXEC PGM=RACFXREF
//STEPLIB DD DSN=apf.auth.library,DISP=SHR
//SYSPRINT DD SYSOUT=*  
```

If RACFXREF detects any orphaned IDs, the SYSPRINT output dataset will contain output similar to that shown earlier in this article.

CONCLUSION

RACFXREF has proved to be a very useful utility in identifying orphaned ID exposures. Based on the information reported by RACFXREF, a RACF administrator can clean up potential exposure conditions that are identified. Try running RACFXREF against your RACF database. You may be surprised at what you find out!

RACFXREF ASSEMBLER

```
*--------------------------------------------------------------------*
*                                                                     *
*   The RACFXREF utility searches the RACF database for IDs that      *
*   are defined as owners of resource profiles and for IDs that      *
*   are assigned to standard or conditional access lists for         *
*   resource profiles and these IDs are no longer defined as RACF    *
*   userids or groups. These IDs are often referred to as RACF        *
*   orphaned IDs.                                                     *
*                                                                     *
*   This anomaly can exist because on a userid or group delete,      *
*   RACF does not clean up resource profile ownership nor does it     *
*   remove the deleted ID from resource profile access lists. This    *
*   causes an exposure in that if a deleted ID is ever reused, the    *
*   new owner of the ID will assume the security permissions that     *
*   were previously assigned to that ID for any remnant access       *
*   list residence or resource profile ownership.                    *
*                                                                     *
*   RACFXREF will scan the DATASET and GENERAL resource profiles      *
*   and report on any orphaned ID it detects as being the owner of   *
*   a resource profile or any orphaned ID it detects in a standard   *
*   or conditional access list of a resource profile. If any          *
```
* orphaned IDs are detected, the RACF administrator can *
* proactively decide if resource profile ownership should be *
* reassigned or if the ID should be removed from the corresponding *
* access list.
*
*---------------------------------------------------------------------*

RACFXREF CSECT
RACFXREF AMODE 31
RACFXREF RMODE ANY
STM R14,R12,12(R13)         Save incoming registers
LR  R3,R13                  Copy R13
LR  R2,R1                   Copy R1
LR  R12,R15                 Copy R15
LA  R11,4095(R12)           Set second base ...
LA  R11,1(R11)               register address
USING RACFXREF,R12,R11      Set module addressability
STORAGE OBTAIN,LENGTH=WORKLEN,LOC=ANY
LR  RØ,R1                   Copy storage address
LR  R14,R1                  Again
LR  R13,R1                  Again
L   R1,=A(WORKLEN)          Get storage length
XR  R15,R15                 Set fill byte
MVCL RØ,R14                 Clear the storage
USING WORKAREA,R13          Set storage addressability
ST  R3,SAVEAREA+4           Save incoming savearea addr
ST  R2,PARM                 Save PARM addr

*--------------------------------------------------------------------*
STORAGE OBTAIN,LENGTH=SYSPRNTL,LOC=BETWEEN
LR  R9,R1                   Copy DCB area addr
MVC Ø(SYSPRNTL,R9),SYSPRINT Copy DCB model
OI  OPENLST,X'80'           Set parm bit on
OPEN ((R9),OUTPUT),MODE=31,MF=(E,OPENLST) Open SYSPRINT
TM  48(R9),X'10'             Open successful?
BNO RETURNØ4                No - all done
MVC SAFWORK(2),=H'44'       Set buffer length
MVC SAFWORK+2(2),=H'1'      Set entry length
MVI SAFWORK+4,C' '           Set fill byte
MVC SAFWORK+5(43),SAFWORK+4 Initialize the buffer
MVC CLASS(8),=C'DATASET '   Set CLASS to 'DATASET'
DSLP1 DS ØH
USING EXTWKEA,R6

*--------------------------------------------------------------------*
* Check all DATASET profiles to determine if any access list         *
* entries do not have a userid or group definition.                 *
*--------------------------------------------------------------------*
MVC ROUTWRK3(ROUTLEN3),RACROUT3 Move in RACROUTE model
RACROUTE REQUEST=EXTRACT, X
   TYPE=EXTRACTN, X
ENTITYX=SAFWORK, X
RELEASE=1.9.2, X
FIELDS=FLDLIST2, X
GENERIC=YES, X
SUBPOOL=1, X
WORKA=RACWORK, MF=(E,ROUTWRK3)

LTR R15,R15 Any data?
BNZ DSEND No - done with DATASET profiles
LR R6,R1 Save extract area address
MVC FLAGSAVE(1),EXTFLAG Save extract flag
B DSPRFOWN Process as profile owner

DLSP2 DS ØH

* * Extract discrete DSN profiles. *
* *--------------------------------------------------------------------*
MVC ROUTWRK3(ROUTLEN3),RACROUT3 Move in RACROUTE model
RACROUTE REQUEST=EXTRACT, X
TYPE=EXTRACTN, X
ENTITYX=SAFWORK, X
RELEASE=1.9.2, X
FIELDS=FLDLIST2, X
GENERIC=YES, X
SUBPOOL=1, X
WORKA=RACWORK, MF=(E,ROUTWRK3)

LTR R15,R15 Any date?
BNZ DSEND No - done with DATASET profiles
LR R6,R1 Save extract area address
MVC FLAGSAVE(1),EXTFLAG Save extract flag
B DSPRFOWN Process as profile owner

DSPRFOWN DS ØH

* * Do profile ownership check. *
* *--------------------------------------------------------------------*

XR R15,R15 Clear R15
ICM R15,'Ø011',EXTWOFF Get offset of data area
AR R15,R6 Point to OWNER
MVC USERID(8),4(R15) Copy userid
BAL R14,USRGRPCH Check if id exists
LTR R15,R15 Exists?
BZ OWNEROK1 Yes - go on
MVC CONDIT_N(8),=C'Owner ' Set Owner condition
BAL R14,WRITEREC Write output record

OWNEROK1 DS ØH

* * Check the ids assigned to the standard access list to see if they *
* exist. *
* *--------------------------------------------------------------------*

LR R7,R6 Save a copy
XR R8,R8 Clear R8
ICM R8,'Ø011',EXTWOFF Get offset of data area
ACLLP1 DS ØH
CR R3,R4
Done with standard ACL?
BNL NOSACL1
Yes - check conditional ACL
ICM R5,B'1111',Ø(R3)
Get userid length
LTR R5,R5
Any data?
BZ NOUSRID1
No - flush to next entry
MVC USERID(8),4(R3)
Copy userid
BAL R14,USRGRPCH
Check if id exists
LTR R15,R15
Exists?
BZ NOUSRID1
Yes - go on
MVC CONDIT_N(8),=C'Standard'
Set ACL condition
BAL R14,WRITEREC
Write output record

NOUSRID1 DS ØH
LA R3,4(R5,R3)
Skip past userid
ICM R5,B'1111',Ø(R3)
Get ACS length
LA R3,4(R5,R3)
Skip past ACS
B ACLLP1
Check next ACL entry

NOUSACL1 DS ØH
LA R6,4(R8,R6)
Point to ACL2
ICM R8,B'1111',Ø(R6)
Get length of ACL2 data
LTR R8,R8
Any data?
BZ NOUSACL1
No - nothing to do

ACLLP2 DS ØH
CR R3,R4
Done with conditional ACL?
BNL NOUSACL1
Yes - check general profiles
ICM R5,B'1111',Ø(R3)
Get PROGRAM name length
MVC CNDPGMNM(8),4(R3)
Save PROGRAM name
LA R3,4(R5,R3)
Point past PROGRAM name
ICM R5,B'1111',Ø(R3)
Get userid length
LTR R5,R5
Any data?
BZ NOUSRID2
No - flush to next entry
MVC USERID(8),4(R3)
Copy userid
BAL R14,USRGRPCH
Check if id exists
LTR R15,R15
Exists?
BZ NOUSRID2
Yes - go on
MVC CONDIT_N(8),CNDPGMNM
Set ACL condition
BAL R14,WRITEREC
Write output record

NOUSRID2 DS ØH

LA R3,4(R5,R3)   Skip past userid
ICM R5,'1111',Ø(R3) Get ACS length
LA R3,4(R5,R3)   Skip past ACS
ICM R5,'1111',Ø(R3) Get access count length
LA R3,4(R5,R3)   Skip access count
ICM R5,'1111',Ø(R3) Get conditional data length
LA R3,4(R5,R3)   Skip conditional data
B ACLLP2   Check next ACL entry
NOCACL1 DS ØH
LR R1,R7   Get extract area addr
LR R6,R7   Again
XR R7,R7   Clear R7
XR R8,R8   Clear R8
ICM R7,'0111',EXTWLN Get storage length
ICM R8,'0001',EXTWSP Get storage subpool
STORAGE RELEASE,LENGTH=(R7),ADDR=(R1),SP=(R8)
TM FLAGSAVE,X'80'   A GENERIC profile?
BZ DSLP2   No - process DISCRETE
B DSLP1   Get next dataset profile
DSEND DS ØH

* The dataset profiles have been checked. Now let's check the
* general resource profiles.
* Get class descriptor table address from RACF RCVT.
*--------------------------------------------------------------------*
L R7,16   Get CVT address
USING CVT,R7 Set addressability
L R8,CVTRAC Get RCVT address
USING RCVT,R8 Set addressability
L R2,RVTCDTDP Get CLASS descriptor table addr
DROP R7,R8
LTR R2,R2   A CDT entry?
BZ GENEND   No - we're done with resources
*--------------------------------------------------------------------*
CLASSLP2 DS ØH
* Check the discrete profiles in this class first.
*--------------------------------------------------------------------*
CLC Ø(2,R2),=H'0'   Any data?
BE GENEND No - done with GENERAL resources
MVC SAFWORK(2),=H'255' Set buffer length
MVC SAFWORK+2(2),=H'1' Set entry length
MVI SAFWORK+4,C' ' Set fill byte
MVC SAFWORK+5(254),SAFWORK+4 Initialize the buffer
MVC CLASS(8),3(R2) Move in CLASS name
MVI FLAGSAVE,X'00' Set flag to DISCRETE
ENTLP2 DS ØH
* Extract discrete profiles for this class.
*--------------------------------------------------------------------*
*--------------------------------------------------------------------*
* MVC   ROUTWRK4(ROUTLEN4),RACROUT4 Move in RACROUTE model          *
* RACROUTE REQUEST=EXTRACT,                                       X *
*     TYPE=EXTRACTN,                                                X *
*     ENTITYX=SAFWORK,                                             X *
*     RELEASE=1.9.2.,                                              X *
*     FIELDS=FLDLIST2,                                             X *
*     GENERIC=ASIS,                                                X *
*     CLASS=CLASS,                                                 X *
*     SUBPOOL=1,                                                  X *
*     WORKA=RACWORK,MF=(E,ROUTWRK4)                                 X *
* LTR   R15,R15                         Any data?                   *
*        BNZ   ENTGEN                  No - check GENERIC             *
* ENTCHK  DS    ØH                  Copy extract work area addr       *
*--------------------------------------------------------------------*
* Do profile ownership check.                                      *
*--------------------------------------------------------------------*
* XR    R15,R15                     Clear R15                        *
* ICM   R15,B'Ø011',EXTWOFF      Get offset of data area          *
* AR    R15,R6                       Point to OWNER                   *
* MVC   USERID(8),4(R15)            Copy userid                        *
* BAL   R14,USRGRPCH              Check if id exists                   *
* LTR   R15,R15                     Exists?                             *
* BZ    OWNEROK2                   Yes - go on                           *
* MVC   CONDIT_N(8),=C'Owner       ' Set Owner condition         *
* BAL   R14,WRITEREC                     Write output record        *
*--------------------------------------------------------------------*
* Check the discrete profiles for the current class and determine   *
* if any access list entries do not have a userid or group          *
*--------------------------------------------------------------------*
* LR    R7,R6                           Save a copy                      *
* XR    R8,R8                          Clear R8                              *
* ICM   R8,B'Ø011',EXTWOFF      Get offset of data area          *
* AR    R6,R8                           Point to OWNER                   *
* ICM   R8,B'1111',Ø(R6)               Get length of OWNER data      *
* LA    R6,4(R8,R6)                       Point to ACL                     *
* ICM   R8,B'1111',Ø(R6)               Get length of ACL data        *
* LTR   R8,R8                               Any data?                     *
* BZ    NOSACL2                       No - check conditional ACL            *
*--------------------------------------------------------------------*
* A standard access list for this profile exists. Check each       *
* entry in the access list to see if it exists as a RACF userid or *
* group.                                                            *
*--------------------------------------------------------------------*
* LA    R4,4(R8,R6)                         Set end of ACL area addr   *
* LA    R3,4(,R6)                           Point to first ACL entry    *
*--------------------------------------------------------------------*
CR    R3,R4                   Done with standard ACL?
BNL   NOSACL2                 Yes - check conditional ACL
ICM   R5,B'1111',Ø(R3)        Get userid length
LTR   R5,R5                   Any data?
BZ    NOUSRID3                No - flush to next entry
MVC   USERID(N),4(R3)         Copy userid
BAL   R14,USRGRPCH            Check if id exists
LTR   R15,R15                 Exists?
BZ    NOUSRID3                Yes - go on
MVC   CONDIT_N(8),=C'Standard' Set ACL condition
BAL   R14,WRITEREC            Write output record

NOUSRID3  DS    ØH
LA    R3,4(R5,R3)             Skip past userid
ICM   R5,B'1111',Ø(R3)        Get ACS length
LA    R3,4(R5,R3)             Skip past ACS
B    ACLLP3                  Check next ACL entry

NOSACL2   DS    ØH
LA    R6,4(R8,R6)             Point to ACL2
ICM   R8,B'1111',Ø(R6)        Get length of ACL2 data
LTR   R8,R8                   Any data?
BZ    NOCACL2                 No - nothing to do

*--------------------------------------------------------------------*
*   A conditional access list for this profile exists. Check each      *
*   entry in the access list to see if it exists as a RACF userid or  *
*   group.                                                             *
*--------------------------------------------------------------------*
LA    R4,4(R8,R6)             Set end of ACL area addr
LA    R3,4(,R6)               Point to first ACL entry

ACLLP4   DS    ØH
CR    R3,R4                   Done with conditional ACL?
BNL   NOCACL2                 Yes - get out
ICM   R5,B'1111',Ø(R3)        Get ACL2NAME length
MVC   CONDIT_N(8),4(R3)       Set ACL condition
LA    R3,4(R5,R3)             Point to ACL2UID
ICM   R5,B'1111',Ø(R3)        Get ACL2UID length
LTR   R5,R5                   Any data?
BZ    NOUSRID4                No - flush to next entry
BAL   R14,USRGRPCH            Check if id exists
LTR   R15,R15                 Exists?
BZ    NOUSRID4                Yes - go on
BAL   R14,WRITEREC            Write output record

NOUSRID4  DS    ØH
LA    R3,4(R5,R3)             Skip past userid
ICM   R5,B'1111',Ø(R3)        Get ACL2ACC length
LA    R3,4(R5,R3)             Skip past ACL2ACC
ICM   R5,B'1111',Ø(R3)        Get ACL2ACNT length
LA    R3,4(R5,R3)             Skip past ACL2ACNT
ICM   R5,B'1111',Ø(R3)        Get ACL2RSVD length
LA    R3,4(R5,R3)             Skip past ACL2RSVD
B    ACLLP4                  Check next ACL entry
**NOCACL2** DS ØH
LR R1,R7 Get extract area addr
LR R6,R7 Again

*--------------------------------------------------------------------*
XR R7,R7 Clear R7
XR R8,R8 Clear R8
ICM R7,B'Ø111',EXTWL Get storage length
ICM R8,B'ØØØ1',EXTWSP Get storage subpool
STORAGE RELEASE,LENGTH=(R7),ADDR=(R1),SP=(R8)
TM FLAGSAVE,X'8Ø' Processing generics?

BO ENTLP3 Yes - do next GENERIC
B ENTLP2 Get next profile

**ENTGEN** DS ØH
MVC SAFWORK(2),=H'255' Set buffer length
MVC SAFWORK+2(2),=H'1' Set entry length
MVI SAFWORK+4,C' ' Set fill byte
MVC SAFWORK+5(254),SAFWORK+4 Initialize the buffer
MVI CLASS(8),3(R2) Move in CLASS name
MVI FLAGSAVE,X'8Ø' Set flag to GENERIC

**ENTLP3** DS ØH

*--------------------------------------------------------------------*
* Extract generic profiles for this class. *
*--------------------------------------------------------------------*
MVC ROUTWRK4(ROUTLEN4),RACROUTE Move in RACROUTE model
RACROUTE REQUEST=EXTRACT,
  TYPE=EXTRACTN,
  ENTITYX=SAFWORK,
  RELEASE=1.9.2,
  FIELDS=FLDLIST2,
  GENERIC=YES,
  CLASS=CLASS,
  SUBPOOL=1,
  WORKA=RACWORK,MF=(E,ROUTWRK4)
LTR R15,R15 Any data?
BZ ENTCHK Yes - check things out

**ENTEND** DS ØH
XR R15,R15 Clear R15
ICM R15,B'ØØ11',Ø(R2) Get CLASS entry length
LA R2,Ø(R15,R2) Point to next entry
B CLASSLP2 Check it out

**GENEND** DS ØH

*--------------------------------------------------------------------*
**RETURNØØ** DS ØH
OI CLOSELST,X'8Ø' Set parm bit on
CLOSE ((R9)),MODE=31,MF=(E,CLOSELST) Close SYSPRINT
STORAGE RELEASE,LENGTH=SYSPRNTL,ADDR=(R9)
L R3,SAVEAREA+4 Save savearea address
LR R1,R13 Get temporary storage address
STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
LR R13,R3 Copy savearea address
LM    R14,R12,12(R13)         Restore registers
XR    R15,R15                 Set return code
BR    R14                     Return

RETURNØ4 DS    ØH
OI    CLOSELST,X'8Ø'          Set parm bit on
CLOSE ((R9)),MODE=31,MF=(E,CLOSELST) Close SYSPRINT
STORAGE RELEASE,LENGTH=SYSPRNTL,ADDR=(R9)
L     R3,SAVEAREA+4           Save savearea address
LR    R1,R13                  Get temporary storage address
STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
LR    R13,R3                  Copy savearea address
LM    R14,R12,12(R13)         Restore registers
LA    R15,4                   Set return code
BR    R14                     Return

*--------------------------------------------------------------------*
*   SUBROUTINES                                                      *
*--------------------------------------------------------------------*
USRGRPCH DS    ØH
*--------------------------------------------------------------------*
*   The USRGRPCH routine checks the id specified in USERID for        *
*   existence as either a RACF userid or a RACF group. If it finds   *
*   USERID as either a userid or group, USRGRPCH returns a return     *
*   code of Ø. If USERID is detected as neither a userid or group,   *
*   USRGRPCH returns a return code of 4.                             *
*--------------------------------------------------------------------*
STM   RØ,R14,SVØ2AREA         Save registers
MVC   ROUTWRK1(ROUTLEN1),RACROUT1 Move in RACROUTE model
RACROUTE REQUEST=EXTRACT,                                     X
   TYPE=EXTRACT,                                            X
   ENTITY=USERID,                                       X
   FIELDS=FLDLIST1,                                   X
   SUBPOOL=1,                                           X
   RELEASE=1.9.2,                                       X
   WORKA=RACWORK, MF=(E,ROUTWRK1)
LTR   R15,R15                 Userid located?
BNZ   CHKGRP                  No - check group
LR    R6,R1                   Get extract workarea addr
XR    R8,R8                   Clear R8
XR    R7,R7                   Clear R7
IC    R7,Ø(,R6)               Get subpool
ICM   R8,B'Ø111',1(R6)        Get length
STORAGE RELEASE,LENGTH=(R8),ADDR=(R6),SP=(R7)
B     USRGRPRØ                Return rc=Ø

CHKGRP DS    ØH
MVC   ROUTWRK2(ROUTLEN2),RACROUT2 Move in RACROUTE model
RACROUTE REQUEST=EXTRACT,                                     X
   TYPE=EXTRACT,                                            X
   ENTITY=USERID,                                       X
   FIELDS=FLDLIST1,                                   X
   SUBPOOL=1,                                           X
   SUBPOOL=2,                                           X
**WRITEREC DS ØH**

* The WRITEREC routine is used to write an orphaned id record to the SYSPRINT output dataset.

* On entry:
  * USERID - contains the orphaned id
  * CONDIT_N - contains the owner or access list indicator that identifies the location of the orphaned id
    * Owner - indicates that the orphaned id is the owner of the indicated profile
    * Standard - indicates that the orphaned id is in the standard access list of the indicated profile
    * condacl - a CONDIT_N value of anything other than 'Owner' or 'Standard' is the conditional access list name that the orphaned id is in in the indicated profile
  * CLASS - contains the RACF CLASS name for the indicated profile
  * SAFWORK - at offset +4 into SAFWORK, the profile name containing the orphaned userid is specified

**STM R0,R14,SV02AREA**  
**TM OUTFLAG,HDRENONE**  
**BO NOHDR**  
**MVI OUTREC,C' '**  
**MVC OUTREC+1(132),OUTREC**  
**MVC OUTREC+Ø(8),=C'Orphaned'**
PUT (R9),OUTREC             Write first header
MVC OUTREC+Ø(44),=C' ID   Type  Condition  CLASS  Prox
      file'             Set second header record
PUT (R9),OUTREC             Write second header
MVC OUTREC+Ø(44),=C'========  ====  =========  ========  ===x
      ==='             Set third header record
MVC OUTREC+44(88),OUTREC+43 Finish third header
PUT (R9),OUTREC             Write third header
OI OUTFLAG,HDRDONE         Set headers written flag

NOHDR DS ØH
MVI OUTREC,C' '          Set fill byte
MVC OUTREC+1(132),OUTREC Initialize to blanks
MVC OUTREC+Ø(8),USERID    Copy userid
MVC OUTREC+16(8),CONDIT_N Copy ACL condition
MVC OUTREC+27(8),CLASS    Copy CLASS
MVC OUTREC+1Ø(3),=C'Gen'  Set Generic as type default
TM FLAGSAVE,X'8Ø'        A GENERIC profile?
BO GENERIC               Yes - go on
MVC OUTREC+1Ø(3),=C'Dis'  Set Discrete as type

GENERIC DS ØH
XR R7,R7                  Clear R7
ICM R7,B'ØØ11',SAFWORK+2  Get profile length
LA R8,SAFWORK+4           Get profile start address
MVC OUTREC+37(MAXPRFLN),SAFWORK+4 Copy 1st part of prof name
PUT (R9),OUTREC           Write output record
LA R15,MAXPRFLN           Max line len for profile name
CR R7,R15                 Prof name len <= line max?
BNH WRITEEND               Yes - done
MVI OUTREC,C' '          Set fill byte
MVC OUTREC+1(132),OUTREC Initialize to blanks
SR R7,R15                 Reduce remaining length
LA R8,Ø(R15,R8)           Addr of next part of prof name
MVC OUTREC+37(MAXPRFLN),Ø(R8) Copy 2nd part of prof name
PUT (R9),OUTREC           Write output record
LA R15,MAXPRFLN           Max line len for profile name
CR R7,R15                 Prof name len <= line max?
BNH WRITEEND               Yes - done
MVI OUTREC,C' '          Set fill byte
MVC OUTREC+1(132),OUTREC Initialize to blanks
SR R7,R15                 Reduce remaining length
LA R8,Ø(R15,R8)           Addr of next part of prof name
MVC OUTREC+37(255-(2*MAXPRFLN)),Ø(R8) 3rd part of prof name
PUT (R9),OUTREC           Write output record
WRITEEND DS ØH
XR R15,R15                Set rc=Ø
LM RØ,R14,SVØ2AREA        Restore registers
BR R14                    Return

MAXPRFLN EQU 95

*--------------------------------------------------------------------*
*   CONSTANTS                                                         *
*--------------------------------------------------------------------*
*--------------------------------------------------------------------*
SYSPRINT DCB MACRF=(PM),DDNAME=SYSPRINT,LRECL=133,DSORG=PS
SYSPRNTL EQU *-SYSPRINT
*--------------------------------------------------------------------*
FLDLIST1 DC F'1'
   DC CL8'AUTHOR '
*--------------------------------------------------------------------*
FLDLIST2 DC F'3'
   DC CL8'AUTHOR '
   DC CL8'ACL1 '
   DC CL8'ACL2 '
*--------------------------------------------------------------------*
RACROUT1 RACROUTE REQUEST=EXTRACT, TYPE=EXTRACT, CLASS='USER',
   RELEASE=1.9.2, MF=L
ROUTLEN1 EQU *-RACROUT1
*--------------------------------------------------------------------*
RACROUT2 RACROUTE REQUEST=EXTRACT, TYPE=EXTRACT, CLASS='GROUP',
   RELEASE=1.9.2, MF=L
ROUTLEN2 EQU *-RACROUTE2
*--------------------------------------------------------------------*
RACROUT3 RACROUTE REQUEST=EXTRACTN, TYPE=EXTRACTN,
   CLASS='DATASET', RELEASE=1.9.2, MF=L
ROUTLEN3 EQU *-RACROUTE3
*--------------------------------------------------------------------*
RACROUT4 RACROUTE REQUEST=EXTRACT, TYPE=EXTRACTN,
   RELEASE=1.9.2, MF=L
ROUTLEN4 EQU *-RACROUTE4
*--------------------------------------------------------------------*
LTORG
*--------------------------------------------------------------------*
WORKAREA DSECT
SAVEAREA DS 18F
SVØ2AREA DS 18F
PARM DS F
OPENLST OPEN (,),MODE=31,MF=L
CLOSELST CLOSE (,),MODE=31,MF=L
OUTREC DS CL133
USERID DS CL8
RACWORK DS ØD,CL(512)
Trustgenix and IdentityForge have announced a technology partnership that enables enterprises to use their existing mainframe identity repositories to provide Single Sign-On (SSO) to internal as well as partner-hosted applications. The integration of Trustgenix IdentityBridge with IdentityForge LDAP Gateway significantly reduces SSO administration costs for companies with mainframe applications.

For further information contact:
URL: www.identityforge.com/content/view/73/1.

---

Innovation Data Processing has announced FDRCRYPT, the first encryption back-up utility specifically designed for z/OS. It will transparently employ the new CP Assist for Cryptographic Function (CPACF) Advanced Encryption Standard (AES) hardware feature for both data encryption and decryption on all z9-109 models.

FDRCRYPT is an optional add-on to the FDR back-up and recovery suite. It can be used to encrypt back-up data being sent off-site (such as disaster recovery tape back-ups) against unauthorized access. It can also be used to encrypt all back-up data.

For further information contact:

---

BMC Software has announced BMC Identity Management Suite, which provides an integrated comprehensive solution set allowing customers to navigate across all of BMC Software's identity management applications.

The BMC Identity Management Suite focuses on the allocation and management of identity and access rights of both internal and external users. BMC Software's identity management solutions link entire user populations (employees, partners, suppliers, and customers) to processes, systems, and business services, allowing customers to further simplify their identity management operations, comply with regulatory mandates, and protect sensitive information while streamlining and improving business services.

For further information contact:

---

SSH Communications Security has announced the SSH Tectia Server for mainframes, which allows mainframe users to secure file transfers, system administration, and other TN3270 applications running on IBM mainframes.

SSH Tectia integrates the Secure Shell protocol for z/OS mainframes. SSH Tectia Server incorporates standards-based SFTP (Secure File Transfer Protocol) functionality to ensure confidentiality, integrity, and authentication of critical file transfers. Command-line tools and file transfer client programs enable easy scripting of automated file transfers such as overnight JCL batch transfers, log file gathering, and database back-ups. It supports RACF authentication.

For further information contact:
URL: www.ssh.com/company/newsroom/article/669.