



# 48

# RACF

May 2007

---

## In this issue

- 3 Password phrase – beyond the 8-character limit
  - 7 Add a user to a RACF environment
  - 15 Mixed-case RACF passwords
  - 27 Win a ‘free pass’ for your next audit
  - 38 Overview of RACF enhancements in z/OS V1R8
  - 48 Automatic assignment of Unix UIDs or GIDs
  - 54 Pentland Utilities V2.0 – an update
  - 70 August 2004–May 2007 index
  - 72 RACF news
- 

update

© Xephon Inc 2007

# **RACF Update**

---

## **Published by**

Xephon Inc  
9330 LBJ Freeway  
Suite 800  
Dallas, Texas 75243  
USA

Phone: 214-340-5690

Fax: 214-341-7081

## **Editor**

Trevor Eddolls

E-mail: [trevore@xephon.com](mailto:trevore@xephon.com)

## **Publisher**

Colin Smith

E-mail: [info@xephon.com](mailto:info@xephon.com)

## **RACF Update on-line**

Code from *RACF Update*, and complete issues in Acrobat PDF format, can be downloaded from <http://www.xephonusa.com>.

## **Subscriptions and back-issues**

A year's subscription to *RACF Update* (four quarterly issues) costs \$290.00 in the USA and Canada; \$355.00 in the UK; \$358.75 in Europe; \$362.30 in Australasia and Japan; and \$361.50 elsewhere. In all cases the prices are in US Dollars and include postage. Individual issues, starting with the August 2002 issue, are available separately to subscribers for \$72.75 (\$92.00 int.) 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 \$175 per 1000 words for the first thousand, and \$80 per thousand for each additional thousand. Xephon pays \$85 per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of \$35 per 100 lines. Contributors wishing to be paid in sterling will be paid at the dollar rates converted into sterling using the prevailing exchange rate. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from [www.xephonusa.com/extras/nfc.pdf](http://www.xephonusa.com/extras/nfc.pdf).

---

© Xephon Inc 2007. 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.*

# Password phrase – beyond the 8-character limit

## INTRODUCTION

In z/OS V1.8, significant improvements have been made to RACF password processing, and this article describes some of the other updates. As an alternative to using traditional passwords, you can now use a passphrase, which is a character string made up of mixed-case letters, numbers, and special characters. Passphrase support provides infrastructure changes that you (or applications) can exploit to facilitate authentication information across environments and applications.

Users can have a password and or a password phrase, and therefore the same user ID can be used for both traditional applications that accept passwords and new applications that take advantage of the password phrase implementation.

This support is intended to help to improve system security and usability. In this article I will give you an introduction to password phrase concepts, password phrase usage and invocation, and some important RACF commands for password phrase.

## PASSWORD PHRASE CONCEPTS

In z/OS V1R8, RACF has implemented the password phrase technology as an infrastructure to provide an alternative to traditional passwords, thereby improving system security authentication to better fit with distributed operation systems. Your password might need to satisfy certain installation-defined rules and RACF is able to exploit the password phrase as a string of characters, from 14 to 100 bytes. A password phrase can be longer than the previous standard password, allowing a longer value than the current 8-byte limit on traditional passwords. Password phrases can contain characters that are not allowed in a password, including blanks.

Therefore, in addition to traditional passwords, you can also have an optional password phrase, which you can use instead of a password with applications that support password phrase technology. Where the password phrase infrastructure is enabled, there are security advantages over passwords because they are long enough to withstand most hacking attempts.

Although RACF allows setting a password phrase, currently no z/OS V1R8 component supports the use of password phrase. Password phrase is an alternative to your password for verifying your identity; however, your installation might have applications that support password phrases.

## UNDERSTANDING HOW THE PASSWORD PHRASE WORKS

You can issue the PHRASE operand with the RACF **ADDUSER** or **ALTUSER** commands to assign a password phrase for a user ID. The rules are set according to the RACF password phrase syntax rules and the ICHPWX11 exit. This enables the user to authenticate itself using a password phrase instead of a password when using an application that supports password phrase.

RACF has the following rules for password phrases:

- Must be a text string from 14 to a maximum 100 characters.
- Must not contain the user ID in sequential upper-case or sequential lower-case characters.
- Must contain at least two alphabetic characters specified (A–Z, a–z).
- Must contain at least two non-alphabetic characters specified (numeric, punctuation, special characters, or blanks).
- Must not contain more than two consecutive identical characters.

The password phrase must be changed after a certain interval

of time to help ensure that only you know it. The interval is the same one that determines when you must change your password, therefore you might need to change it periodically, according to your `SETROPTS PASSWORD INTERVAL` suboperand, as shown below:

```
PASSWORD PROCESSING OPTIONS: PASSWORD CHANGE INTERVAL IS 90 DAYS.
```

## RACF COMMANDS AND PASSWORD PHRASES

There are several RACF commands that now have modifications for password phrases. These commands are changed to have a password phrase suboperand:

- Add user ID **ADDUSER | AU** – the **ADDUSER** command with the **PHRASE** suboperand, specifies the user ID's initial password phrase, as shown below:

```
ADDUSER (ADD USER PROFILE) using the password phrase suboperand:
```

```
ADDUSER | AU (userid ...) PHRASE ('passphrase')
```

If the **PHRASE** suboperand is omitted, no password phrase is assigned. However, if you enter the **PHRASE** suboperand without a password phrase value, you are prompted for a value.

- List user profile with **LISTUSER | LU** – listing the user profile, you can verify whether a password phrase is assigned to the user ID, and, if so, you can also verify the last password phrase change date through the **PHRASEDATE** field. The **LISTUSER** command output shows the following new fields:
  - **ATTRIBUTES=PASSPHRASE** if a password phrase is assigned to the user ID.
  - **PHRASEDATE** field, which gives details of the last password phrase change date.

**LISTUSER** command output showing the **PASSPHRASE** and **PHRASEDATE** field is shown below:

```
USER=TEST1 NAME=UNKNOWN OWNER=SYS1 CREATED=06.186
```

DEFAULT-GROUP=SYS1 PASSDATE=00.000 PASS-INTERVAL= 90  
PHRASEDATE=06.186 ATTRIBUTES=PASSPHRASE

- Alter user ID with **ALTUSER | ALU** – the **ALTUSER** command with the **PHRASE** suboperand, specifies the user ID's initial password phrase. The password phrase is always set to expire unless the **NOEXPIRED** operand is issued, thus requiring the user to change it on initial use.

The **ALTUSER** command with the **PHRASE** suboperand is used in order to perform the following actions:

- add a password phrase to a user ID that does not have a password phrase
- change a user ID's password phrase
- remove a password phrase from a user ID that has one.

The RACF **ALTUSER** command syntax with the **PHRASE** and **NOPHRASE** suboperand is shown below:

```
ALTUSER | ALU (user id ...) PHRASE(' passphrase' ) | NOPHRASE
```

The **NOPHRASE** suboperand specifies that the user cannot use a password phrase for authentication. If a password phrase was previously set, it is cleared. The date of the last password phrase is also cleared from the user's profile. The **EXPIRED** and **NOEXPIRED** suboperands also apply to password phrase.

- **PASSWORD | PHRASE | PW** – this command is usually called the **PASSWORD** command even though the **PHRASE** suboperand has been added as an alias to the **PASSWORD** command. The value specified by **INTERVAL** applies to the password, as well as password phrase. The user keyword does not apply to password phrases. If the **PHRASE** keyword is specified with the **USER** keyword, the **PHRASE** keyword is ignored. The new **PASSWORD** or **PHRASE** syntax command is shown below:

```
PASSWORD | PW | PHRASE AT( node . user id ...) | ONLYAT( node . user id ...)
INTERVAL( change- interval ) | NOINTERVAL
PASSWORD( current- password new- password )
```

```
PHRASE(' current-passphrase' ' new-passphrase' )  
USER(useri d . . . )
```

## SUMMARY

In this article we have discussed one of the significant improvements in RACF in z/OS V1.8 to support the use of passwords longer than eight characters, often called passphrases. A passphrase is a character string that can comprise mixed-case letters, numbers, and special characters including blanks, from 14 to 100 characters in length. Passphrases allow for an exponentially greater number of possible combinations of characters and numbers than do passwords. A user ID can have both a password and a passphrase. The same user ID can be used for both existing applications that accept an eight-character password and those that take advantage of the passphrase infrastructure. This support is intended to help improve system security and usability.

---

*Laxminarayan Sriram*  
*Systems Programmer (USA)*

© Xephon 2007

---

## Add a user to a RACF environment

This article describes a way to define a new user to RACF and establish the user's relationship with an existing RACF-defined group. The command adds a profile for the new user to the RACF database and creates a connect profile that connects the user to whichever default group you specify.

In the main menu you can define some basic RACF parameters for the **ADDUSER** command. You must have the SPECIAL attribute to give the new user the OPERATIONS, SPECIAL, or AUDITOR attribute.

The REXX procedure runs in an ISPF environment as follows:

```
TSO RACUS
```

## RACUS – main REXX procedure:

```
/* REXX */
/* RACF - Add User */
trace r
zpfctl = 'OFF'
Y=MSG("OFF")
address ispeexec 'vput (zpfctl) profile'
cur='usr'
msg='Enter values for new RACF user!'
TOP:
address ispeexec "display panel (racusm) cursor("CUR")"
if rc=8 then Exit
/* Check Input Parameters */
if usr = '' | usr = '?' then do
  address ispeexec
  zedsmsg = "Enter Userid"
  zedlmsg = "Enter Userid - Blank not allowed"
  "setmsg msg(i srz001)"
  usr='?'
  cur='usr'
  Signal Top
end
if pwd = '' | pwd = '?' then do
  address ispeexec
  zedsmsg = "Enter Password"
  zedlmsg = "Enter Password - Blank not allowed"
  "setmsg msg(i srz001)"
  pwd='?'
  cur='pwd'
  Signal Top
end
if ime = '' | ime = '?' then do
  address ispeexec
  zedsmsg = "Enter Name"
  zedlmsg = "Enter Name - Blank not allowed"
  "setmsg msg(i srz001)"
  ime='?'
  cur='ime'
  Signal Top
end
if tusr = '' then do
  address ispeexec
  zedsmsg = "Enter YES or NO"
  zedlmsg = "Enter YES or NO - Blank not allowed"
  "setmsg msg(i srz001)"
  cur='tusr'
  Signal Top
end
select
```



```

when rtyp= 1 then do      /* system programmer */
  dfl tg = 'sys1'        /* default group */
  proc  = 'aproc'        /* procedure */
  spec  = 'special'
  oper  = 'operations'
  audi  = 'auditor'
end
when rtyp= 2 then do      /* aplicat. programer */
  dfl tg = 'program'     /* default group */
  proc  = 'aproc'        /* procedure */
  spec  = 'nospecial' /* */
  oper  = 'nooperations'
  audi  = 'noauditor'
end
when rtyp= 3 then do      /* system File Trans. */
  dfl tg = 'sys2'        /* default group */
  proc  = 'ijkacct'      /* procedure */
  spec  = 'special'
  oper  = 'operations'
  audi  = 'auditor'
end
otherwise nop;
end
if rtyp = '' then do
  address ispexec
  zedsmg = "Enter 1, 2 or 3"
  zedlmsg = "Enter 1, 2 or 3 - Blank not allowed"
  "setmsg msg(i srz001)"
  cur=' rtyp'
  Signal Top
end
if dfl tg = '' | dfl tg='?' then do
  address ispexec
  zedsmg = "Enter Default Group"
  zedlmsg = "Enter Default Group - Blank not allowed"
  "setmsg msg(i srz001)"
  dfl tg='?'
  cur=' dfl tg'
  Signal Top
end
if proc = '?' then do
  Call Logpr
  proc=Result
  cur=' proc'
  Signal Top
end
if proc = '' then do
  address ispexec
  zedsmg = "Enter Logon Procedure"
  zedlmsg = "Enter Logon Procedure - Blank not allowed"

```

```

    "setmsg msg(i srz001)"
    cur='proc'
    Signal Top
end
if acc = '' | acc='?' then do
    address ispexec
    zedsmg = "Enter Account Number"
    zedlmsg = "Enter Account Number - Blank not allowed"
    "setmsg msg(i srz001)"
    acc='?'
    cur='acc'
    Signal Top
end
if size = '' | size='?' then do
    address ispexec
    zedsmg = "Enter Region Size"
    zedlmsg = "Enter Region Size - Blank not allowed"
    "setmsg msg(i srz001)"
    size='?'
    cur='size'
    Signal Top
end
if unit = '' | unit='?' then do
    address ispexec
    zedsmg = "Enter Unit Name"
    zedlmsg = "Enter Unit Name - Blank not allowed"
    "setmsg msg(i srz001)"
    unit='?'
    cur='unit'
    Signal Top
end
if jobc = '' | jobc='?' then do
    address ispexec
    zedsmg = "Enter Job Class"
    zedlmsg = "Enter Job Class - Blank not allowed"
    "setmsg msg(i srz001)"
    jobc='?'
    cur='jobc'
    Signal Top
end
if msgc = '' | msgc='?' then do
    address ispexec
    zedsmg = "Enter Message Class"
    zedlmsg = "Enter Message Class - Blank not allowed"
    "setmsg msg(i srz001)"
    msgc='?'
    cur='msgc'
    Signal Top
end
if holc = '' | holc='?' then do

```

```

address ispexec
zedsmg = "Enter Hold Class"
zedlmsg = "Enter Hold Class - Blank not allowed"
"setmsg msg(i srz001)"
holc='?'
cur='holc'
Signal Top
end
if tusr='NO' then signal notso; /* non-TSO User: CICS */
msg='Press Enter to Continue !'
address ispexec "display panel (racusm) cursor("CUR")"
if rc=8 then Exit
'ADDUSER' usr spec oper audi 'NAME ('ime') DFLTGRP('dfi tg')
PASSWORD('pwd') TSO(PROC('proc') ACCTNUM('acc') SIZE('si ze')
UNIT('uni t') JOBCLASS('jobc') MSGCLASS('msgc') HOLDCLASS('hol c')
SYSOUTCLASS(a))'
if rc > 0 then signal end;
'PERMIT OPER CLASS(TSOAUTH)
ID('usr')'
'PERMIT JCL CLASS(TSOAUTH)
ID('usr')'
'PERMIT MOUNT CLASS(TSOAUTH)
ID('usr')'
'PERMIT RECOVER CLASS(TSOAUTH)
ID(usr)'
'PERMIT ' proc ' CLASS(TSOPROC)
ID('usr')'
'PERMIT ACCT# CLASS(ACCTNUM)
ID('usr')'
'setropts raclist(tsoproc acctnum perfgrp tsoauth) refresh'
end:
return rc
/* non TSO user, only CICS user */
notso:
'ADDUSER' usr '
DFLTGRP(cics)
PASSWORD('pwd')'
return rc

```

## LOGPR – select logon procedure:

```

/* REXX */
/* Selection Logon Procedure */
in=0
Top:
if in=0 then
address ispexec "display panel (pl ogpr) CURSOR(ffile)"
if rc=8 then Exit
/* Select DataSet Name */
if ffile = '' then Signal Top

```

```

dsn = ("ffile")
x=outtrap('var.')
address tso "listds" dsn members
if var.0 = 2 then do
  address ispexec
  zedsmg = "Not found"
  zedlmsg = var.2
  "setmsg msg(srz001)"
  Signal Top
end
address ispexec 'tbcreate "lilst" names(lproc)'
do i=7 to var.0
  lproc = substr(var.i,3)
  address ispexec 'tbadd "lilst"'
end
address ispexec 'tbttop "lilst"'
address ispexec 'tbdispl "lilst" panel (pl ogpr)'
if rc=8 then do
  address ispexec 'tbend "lilst"'
  Exit ' '
end
item=' '
/* Select Logon Procedure */
if cmd='s' | cmd='S' then do
  item = lproc
  address ispexec 'tbend "lilst"'
  Exit item
end
/* Edit Logon Procedure */
if cmd='e' | cmd='E' then do
  tempfile = "ffile"("||lproc||")'
  address ispexec
  "edit dataset("tempfile")"
  address ispexec 'tbend "lilst"'
  cmd=''; lproc=''
  Signal Top
end
/* Browse Logon Procedure */
if cmd='b' | cmd='B' then do
  tempfile = "ffile"("||lproc||")'
  address ispexec
  "browse dataset("tempfile")"
  address ispexec 'tbend "lilst"'
  cmd=''; lproc=''
  Signal Top
end
x=outtrap('off')
address ispexec 'tbend "lilst"'
in=1
cmd=''; lproc=''

```

Signal Top  
Exit

## RACUSM – main menu:

```
)Attr Default(%+_)
! type(text) intens(high) caps(on) color(yellow)
$ type(output) intens(high) caps(off) color(yellow)
] type(output) intens(high) caps(off) color(green) hi li te(reverse)
? type(text) intens(high) caps(on) color(green) hi li te(reverse)
# type(text) intens(high) caps(off) hi li te(reverse)
} type(text) intens(high) caps(off) color(yellow) hi li te(reverse)
[ type( input) intens(high) caps(on) color(green) pad(_)
)Body Expand(//)
! -/-/- ]fi el d +! -/-/-
%Command ==>_zcmd
+
+
#PARAMETER #PARAMETER VALUE #PROMPT
+
+
+User =>[usr + User id
+Password =>[pwd + Password
+Name =>[ime + Name
+TS0 User =>[z + TS0 User! YES+or! NO
+Type =>[z+ Type of User
+Dfl tgrp =>[dfl tg + Default Group
+Procedure=>[proc + Logon Procedure
+Acctnum =>[acc + Account Number
+Si ze =>[si ze + Region Si ze
+Uni t =>[uni t + Unit Name
+JobCl ass =>[z+ Job Cl ass
+MsgCl ass =>[z+ Message Cl ass
+Hol dCl ass=>[z+ Hol d Cl ass
+
$msg
+
+
} PF3 Return +
)I ni t
&fi el d=' °°° RACF - ADD USER °°°'
.ZVARS = '(tusr rtyp j obc msgc hol c)'
if (&usr ^= ' ')
.attr (usr) = 'pad(null s)'
if (&pwd ^= ' ')
.attr (pwd) = 'pad(null s)'
if (&ime ^= ' ')
.attr (ime) = 'pad(null s)'
if (&tusr ^= ' ')
.attr (tusr) = 'pad(null s)'
```

```

if (&rtyp ^= ' ')
    .attr (rtyp) = 'pad(nul l s)'
if (&dfl tg ^= ' ')
    .attr (dfl tg) = 'pad(nul l s)'
if (&proc ^= ' ')
    .attr (proc) = 'pad(nul l s)'
if (&acc ^= ' ')
    .attr (acc) = 'pad(nul l s)'
if (&si ze ^= ' ')
    .attr (si ze) = 'pad(nul l s)'
if (&uni t ^= ' ')
    .attr (uni t) = 'pad(nul l s)'
if (&j obc ^= ' ')
    .attr (j obc) = 'pad(nul l s)'
if (&msgc ^= ' ')
    .attr (msgc) = 'pad(nul l s)'
if (&hol c ^= ' ')
    .attr (hol c) = 'pad(nul l s)'
)Rei ni t
)Proc
    &tu = TRUNC(&tusr, ' ')
    if (&tu='Y' ! &tu='YE') &tusr = 'YES'
    if (&tu='N')          &tusr = 'NO'
    VPUT (usr pwd ime tusr rtyp jobc msgc holc) PROFILE
    VPUT (dfl tg proc acc si ze uni t) PROFILE
)End

```

## PLOGPR – selection logon procedure menu:

```

)Attr Defaul t(%+_)
! type(text)   intens(high) caps(on ) col or(yel low)
$ type(output) intens(high) caps(off) col or(yell ow) hi li te(reverse)
§ type(output) intens(high) caps(off) col or(whi te) hi li te(reverse)
? type(text)   intens(high) caps(on ) col or(green) hi li te(reverse)
# type(text)   intens(high) caps(off) hi li te(reverse)
} type(text)   intens(high) caps(off) col or(whi te)
[ type( i nput) intens(high) caps(on ) j ust(l e f t )
{ type( i nput) intens(high) caps(on ) j ust(l e f t ) pad(' _')
] type( i nput) intens(high) caps(on ) j ust(l e f t ) pad(' -')
^ type(output) intens(low ) caps(off) j ust(asi s ) col or(turquoi se)
)Body  Expand(//)
%-/-/- $title                               +% - / - / -
%Command ==>_zcmd                             / /%Scrol l
==>_amt +
+-----+
-----
+DataSet Name: {ffile                          +
+-----+
-----
+Val id cmd: !S+Sel ect Logon Procedure!B+Browse Fi le!E+Edi t Fi le+

```

```

}F3+-->}End
+-----+
-----
# S #Logon Procedure+
)Model
  ]z+^z          +
)Init
  . ZVARS = '(cmd |proc) '
  &amt = PAGE
  &title = 'Selection Logon Procedure'
)Reinit
)Proc
  VPUT (ffile) PROFILE
)End

```

---

*Bernard Zver (bernard.zver@informatika.si)*  
*DBA*  
*Informatica (Slovenia)*

© Xephon 2007

---

## Mixed-case RACF passwords

Since the beginning of time, or the mid-1970s at least, RACF and the applications that interface with it have supported only upper-case password values. What this means is that regardless of what case the alpha characters of a password value were entered as, before RACF would process the password value for either verification or reset, the alpha characters would need to be converted to upper case. This behaviour is inconsistent with virtually any other computing environment. With z/OS 1.7, RACF has introduced support for mixed-case password values. This expands the realm of possible unique password values by several orders of magnitude and it also positions z/OS systems to be more consistent with other environments (Windows, Unix, etc) with respect to case-sensitive passwords.

### HOW IT'S DONE

With the introduction of mixed-case password support, RACF can be set to one of two basic operational modes:

- 1 Enabled for mixed-case passwords
- 2 Disabled for mixed-case passwords.

Setting RACF to accept mixed-case passwords is done with the **SETROPTS** RACF command. The command syntax is as follows:

```
SETROPTS PASSWORD(MIXEDCASE)
```

Disabling mixed-case passwords is equally simple and also done with the **SETROPTS** RACF command. The command syntax for that is:

```
SETROPTS PASSWORD(NOMIXEDCASE)
```

Care must be taken if RACF is toggled to `PASSWORD(NOMIXEDCASE)` after `PASSWORD(MIXEDCASE)` has been in place. If `PASSWORD(NOMIXEDCASE)` is ever set after `PASSWORD(MIXEDCASE)` has been active, any password verification request for a password value containing lower-case alpha characters will fail. The message here is that it is impossible, through standard system password verification techniques, to verify a password containing lower-case alpha characters if RACF is subsequently set to `PASSWORD(NOMIXEDCASE)`. If RACF is subsequently set back to `PASSWORD(MIXEDCASE)`, passwords containing lower-case alpha characters can then be verified during standard system password verification techniques such as TSO logon.

As mentioned, the basic operation of RACF, when it is password mixed-case capable, is either to be enabled for mixed-case passwords or to be disabled for mixed-case passwords. Those two options can present four possible scenarios for the password status of any user ID. These options are:

- 1 RACF is not set to accept mixed-case passwords and the user ID's password is currently not a mixed-case password.
- 2 RACF is not set to accept mixed-case passwords and the user ID's password is potentially a mixed-case password.



- 3 RACF is set to accept mixed-case passwords and the user ID's password is currently not a mixed-case password.
- 4 RACF is set to accept mixed-case passwords and the user ID's password is potentially a mixed-case password.

Of these four possible options, options 2 presents a problem for standard system password verification techniques.

RACF is able to determine whether a user ID's password has been set while RACF has been enabled for mixed-case passwords through a new RACF BASE segment field. The PASSASIS field has been introduced with RACF on z/OS 1.7. If an existing, pre-z/OS 1.7 RACF database is used after converting to z/OS 1.7, a RACROUTE EXTRACT request for the PASSASIS field will return a zero value. Only when a user ID's password value has been changed while RACF is enabled for mixed-case passwords will the PASSASIS field for a user ID be set to X'80'.

## APPLICATIONS THAT CAN DIRECTLY VERIFY OR CHANGE PASSWORD VALUES

Applications that can directly verify or change password values for user IDs need to make some minor code modifications to properly operate in a mixed-case password environment. Let's look at these two requests.

### A VERIFY password example

Because the password encryption process in RACF is case sensitive, even prior to RACF on z/OS 1.7, a successful RACROUTE REQUEST=VERIFY(X) can occur only if the proper case-sensitive password value is used for the verify. To properly stage the verify request, a test should be made on the PASSASIS flag for the user ID. The code snippet for that would be as follows:

```
*****
* Assumption at this point in the code is that USERIDLN already *
* contains the length of the userid, USERID contains the uppercase *
* userid, OLDPWDLN contains the length of the existing password *
*****
```

```

*   value, PASSWORD contains the current password entered as is.   *
*****
MVI   PASSASIS,X' 00'          CLEAR THE PASSASIS FLAG BYTE
MVC   ROUTWRK1(ROUTLEN1),RACROUT1 MOVE IN RACROUTE MODEL
RACROUTE REQUEST=EXTRACT,
      TYPE=EXTRACT,
      ENTITY=USERID,
      FIELDS=FLDLIST1,
      SUBPOOL=1,
      RELEASE=1.9.2,
      WORKA=RACWORK,MF=(E,ROUTWRK1)
LTR   R15,R15                  EXTRACT SUCCESSFUL?
BZ    XTRACTOK                 YES - GO ON
*   If things fall through to here, the PASSASIS field has not been
*   extracted and it can be assumed that the current password value
*   is not a mixed-case password.
OC    PASSWORD(8),=8C' '      SET PASSWORD TO UPPER CASE
B     DOVERIFY                 GO VERIFY
XTRACTOK DS   0H
USING EXTWKEA,R1              EXTRACT WORKAREA ADDRESSABILITY
XR    R15,R15                  CLEAR R15
ICM   R15,B' 0011',EXTWOFF     GET OFFSET OF DATA AREA
*****
AR    R15,R1                   POINT TO PASSASIS FLAG LEN
ICM   R7,B' 1111',0(R15)      GET LENGTH OF FLAG FIELD
ST    R7,ASISLEN              SAVE THE LENGTH
LTR   R7,R7                    ANY DATA?
BZ    NOASIS                   NO - BYPASS
MVC   PASSASIS(1),4(R15)      SAVE THE FLAG BYTE
NOASIS DS   0H
*****
XR    R7,R7                    CLEAR R7
XR    R8,R8                    CLEAR R8
ICM   R7,B' 0111',EXTWLN      INSERT STORAGE LENGTH
ICM   R8,B' 0001',EXTWSP      INSERT STORAGE SUBPOOL
STORAGE RELEASE,LENGTH=(R7),ADDR=(R1),SP=(R8)
DROP  R1
*****
CLI   PASSASIS,X' 00'          PWD NOT LOWER CASE?
BNE   DOVERIFY                 YES - GO VERIFY AS IS
OC    PASSWORD(8),=8C' '      SET PASSWORD TO UPPER CASE
*****
*   At this point, the old password value in PASSWORD has been
*   appropriately upper cased if required or left as is if the
*   PASSASIS flag is non-zero.
*****
DOVERIFY DS   0H
XC    ACEEADDR(4),ACEEADDR     SANITIZE THE ADDR FIELD
MVC   ROUTWRK2(ROUTLEN2),RACROUT2 MOVE IN RACROUTE MODEL
RACROUTE REQUEST=VERIFY,

```

X

```

                ENVI R=CREATE,                                X
                PASSCHK=YES,                                  X
                PASSWRD=OLDPWDLN,                            X
                USERI D=USERI DLN,                           X
                ACEE=ACEEADDR,                                X
                RELEASE=1. 9. 2,                              X
                MSGSUPP=YES,                                  X
                WORKA=RACWORK, MF=(E, ROUTWRK2)
ST      R15, RETCODE                SAVE RETURN CODE
CLC    ACEEADDR(4), =F' Ø'          AN ACEE?
BE     NOACEE                        NO - DON' T DELETE IT
MVC    ROUTWRK2(ROUTLEN2), RACROUT2 MOVE IN RACROUTE MODEL
RACROUTE REQUEST=VERI FY,                                X
                ENVI R=DELETE,                                X
                PASSCHK=NO,                                    X
                ACEE=ACEEADDR,                                X
                RELEASE=1. 9. 2,                              X
                WORKA=RACWORK, MF=(E, ROUTWRK2)
NOACEE  DS      ØH
        CLC    RETCODE(4), = F' Ø'          VERI FY OK?
        BNE   FAIL                          NO – NOT THE CORRECT PASSWORD
        B     SUCCESS                        THINGS ARE GOOD
.
.
.
FAIL    DS      ØH
*      Process password inval id si tuation.
.
.
.
SUCCESS DS      ØH
*      Process password val id si tuation.
.
.
.
*****
RACROUT1 RACROUTE REQUEST=EXTRACT,                                X
                TYPE=EXTRACT,                                    X
                CLASS=' USER' ,                                X
                RELEASE=1. 9. 2,                              X
                MF=L
ROUTLEN1 EQU    *-RACROUT1
*****
RACROUT2 RACROUTE REQUEST=VERI FY,                                X
                PASSCHK=YES,                                    X
                RELEASE=1. 9. 2,                              X
                MF=L
ROUTLEN2 EQU    *-RACROUT2
*****
FLDLI ST1 DC    F' 1'

```

```

          DC      C' PASSASIS'                MIXED CASE PASSWORD INDICATOR
*****
.
.
.
WORKAREA DSECT
USERIDLN DS      XL1
USERID   DS      CL8
OLDPWDLN DS      XL1
PASSWORD DS      CL8
RETCODE  DS      F
ROUTWRK1 DS      ØD, CL(ROUTLEN1)
ROUTWRK2 DS      ØD, CL(ROUTLEN2)
RACWORK  DS      ØD, CL256
ACEEADDR DS      F
PASSASIS DS      XL1
ASISLEN  DS      F
WORKLEN  EQU     *-WORKAREA
.
.
.
          IRRPRXTW ,

```

## A CHANGE password example

For a password change operation, we still need to check the upper-case status of the existing password for the user ID, but we must also determine whether or not RACF is accepting mixed-case passwords. The code snippet for the change operation would look as follows:

```

*****
* Assumption at this point in the code is that USERIDLN already *
* contains the length of the userid, USERID contains the uppercase *
* userid, OLDPWDLN contains the length of the existing password *
* value, PASSWORD contains the current password entered as is, *
* NEWPWDLN contains the length of the requested new password value, *
* NEWPWD contains the requested new password entered as is. *
*****
          MVI     PASSASIS, X' ØØ'                CLEAR THE PASSASIS FLAG BYTE
          MVC     ROUTWRK1(ROUTLEN1), RACROUT1  MOVE IN RACROUTE MODEL
          RACROUTE REQUEST=EXTRACT,
          TYPE=EXTRACT,
          ENTITY=USERID,
          FIELDS=FLDLIST1,
          SUBPOOL=1,
          RELEASE=1.9.2,
          WORKA=RACWORK, MF=(E, ROUTWRK1)

```

```

        LTR    R15, R15                EXTRACT SUCCESSFUL?
        BZ     XTRACTOK                YES - GO ON
*   If things fall through to here, the PASSASIS field has not been
*   extracted and it can be assumed that the current password value
*   is not a mixed case password.
        OC     PASSWORD(8), =8C' '    SET PASSWORD TO UPPER CASE
        B      DOCHANGE                GO CHANGE
XTRACTOK DS   ØH
        USING EXTWKEA, R1              EXTRACT WORKAREA ADDRESSABILITY
        XR     R15, R15                CLEAR R15
        ICM    R15, B' ØØ11' , EXTWOFF GET OFFSET OF DATA AREA
*****
        AR     R15, R1                POINT TO PASSASIS FLAG LEN
        ICM    R7, B' 1111' , Ø(R15)   GET LENGTH OF FLAG FIELD
        ST     R7, ASISLEN             SAVE THE LENGTH
        LTR    R7, R7                  ANY DATA?
        BZ     NOASIS                  NO - BYPASS
        MVC    PASSASIS(1), 4(R15)     SAVE THE FLAG BYTE
NOASIS  DS    ØH
*****
        XR     R7, R7                  CLEAR R7
        XR     R8, R8                  CLEAR R8
        ICM    R7, B' Ø111' , EXTWLN    INSERT STORAGE LENGTH
        ICM    R8, B' ØØØ1' , EXTWSP    INSERT STORAGE SUBPOOL
        STORAGE RELEASE, LENGTH=(R7), ADDR=(R1), SP=(R8)
        DROP   R1
*****
        CLI    PASSASIS, X' ØØ'        PWD NOT LOWER CASE?
        BNE    DOCHANGE                YES - GO DO CHANGE
        OC     PASSWORD(8), =8C' '    SET PASSWORD TO UPPER CASE
*****
*   At this point, the old password value in PASSWORD has been
*   appropriately upper cased if required or left as is if the
*   PASSASIS flag is non-zero.
*****
DOCHANGE DS   ØH
        L      R1, 16                  GET CVT ADDRESS
        USING CVT, R1                  SET ADDRESSABILITY
        L      R6, CVTRAC              GET RCVT ADDRESS
        USING RCVT, R6                SET ADDRESSABILITY
        DROP   R1
        TM     RCVTFLG3, RCVTPLC       LOWER CASE PASSWORDS OK?
        BO     LC_OK1                  YES - LEAVE NEW PWD AS IS
        OC     NEWPWD(8), =8C' '      SET NEW PASSWORD TO UPPER CASE
LC_OK1  DS    ØH
        XC     ACEEADDR(4), ACEEADDR    SANITIZE THE ADDR FIELD
        MVC    ROUTWRK2(ROUTLEN2), RACROUT2 MOVE IN RACROUTE MODEL
        RACROUTE REQUEST=VERIFY,
        ENVI R=CREATE,
        PASSCHK=YES,

```

```

X
X
X

```

```

          ENCRYPT=YES, X
          NEWPASS=NEWPWDLN, X
          PASSWRD=OLDPWDLN, X
          USERID=USERIDLN, X
          ACEE=ACEEADDR, X
          RELEASE=1.9.2, X
          MSGSUPP=YES, X
          WORKA=RACWORK, MF=(E, ROUTWRK2)
ST      R15, RETCODE          SAVE RETURN CODE
CLC     ACEEADDR(4), =F' Ø'   AN ACEE?
BE      NOACEE                NO - DON'T DELETE IT
MVC     ROUTWRK2(ROUTLEN2), RACROUT2 MOVE IN RACROUTE MODEL
RACROUTE REQUEST=VERIFY, X
          ENVI R=DELETE, X
          PASSCHK=NO, X
          ACEE=ACEEADDR, X
          RELEASE=1.9.2, X
          WORKA=RACWORK, MF=(E, ROUTWRK2)
NOACEE  DS      ØH
          CLC     RETCODE(4), = F' Ø'   CHANGE OK?
          BNE    FAIL                NO - NOT THE CORRECT PASSWORD
          B      SUCCESS             THINGS ARE GOOD
.
.
.
FAIL    DS      ØH
*      Process password invalid situation.
.
.
.
SUCCESS DS      ØH
*      Process password valid situation.
.
.
.
*****
RACROUT1 RACROUTE REQUEST=EXTRACT, X
          TYPE=EXTRACT, X
          CLASS=' USER' , X
          RELEASE=1.9.2, X
          MF=L
ROUTLEN1 EQU    *-RACROUT1
*****
RACROUT2 RACROUTE REQUEST=VERIFY, X
          PASSCHK=YES, X
          RELEASE=1.9.2, X
          MF=L
ROUTLEN2 EQU    *-RACROUT2
*****
FLDLIST1 DC    F' 1'

```

```

          DC      C' PASSASI S'                MI XED CASE PASSWORD I NDI CATOR
*****
.
.
.
WORKAREA DSECT
USERI DLN DS    XL1
USERI D   DS    CL8
OLDPVDLN DS    XL1
PASSWORD DS    CL8
NEWPVDLN DS    XL1
NEWPWD   DS    CL8
RETCODE  DS     F
ROUTWRK1 DS    ØD, CL(ROUTLEN1)
ROUTWRK2 DS    ØD, CL(ROUTLEN2)
RACWORK  DS    ØD, CL256
ACEEADDR DS     F
PASSASI S DS    XL1
ASISLEN  DS     F
WORKLEN  EQU   *-WORKAREA
.
.
.
          CVT    DSECT=YES
          I CHPRCVT ,
          I RRPRTW ,

```

## ADDITIONAL RACF PASSWORD RULE OPTIONS

In addition to being able to indicate that RACF is accepting mixed-case alpha characters in password values, four more options have been introduced for RACF password rules. Password rules can be set in RACF to indicate what characters are acceptable in given byte positions in the password value. The following new password character rules can be used:

- 1 **MIXEDCONSONANT** – this rule indicates that an upper or lower-case consonant is valid in this character position. If the RACF ISPF panels are used to set password rules, a **MIXEDCONSONANT** is indicated with a format value of lower-case 'c'. If the **SETROPTS** command is used to set the password rules, an example command specifying **MIXEDCONSONANT** would be similar to:

```
SETROPTS PASSWORD(RULE1(LENGTH(4: 6) MI XEDCONSONANT(1, 3: 4)))
```

- 2 MIXEDVOWEL – this rule indicates that an upper or lower-case vowel is valid in this character position. If RACF ISPF panels are used to set password rules, a MIXEDVOWEL is indicated with a format value of lower-case ‘v’. If the **SETROPTS** command is used to set the password rules, an example command specifying MIXEDVOWEL would be similar to:

```
SETROPTS PASSWORD(RULE1(LENGTH(3: 7) MIXEDVOWEL(1, 3, 5)))
```

- 3 MIXEDNUM – this rule indicates that this character position can have an upper or lower-case alpha character, a numeric character, or a national character. If RACF ISPF panels are used to set password rules, a MIXEDNUM is indicated with a format value of lower-case ‘m’. There are three extensions to this rule. If only one MIXEDNUM character position is occupied in a password value, that character can be an upper or lower-case alpha character, a numeric character, or a national character. If two MIXEDNUM character positions are occupied in a password value, the two occupied positions must contain two of the following three options:

- An upper-case alpha or national character
- A lower-case alpha character
- A numeric character.

If three (or more) MIXEDNUM character positions are occupied, all three of the above indicated character options must be represented.

If the **SETROPTS** command is used to set the password rules, an example command specifying MIXEDNUM would be similar to:

```
SETROPTS PASSWORD(RULE1(LENGTH(7) MIXEDNUM(1: 3, 5)))
```

- 4 NATIONAL – this rule indicates that this character position must be a national character, @ # \$. If RACF ISPF panels are used to set password rules, a NATIONAL is indicated with a format value of ‘\$’. If the **SETROPTS** command is



used to set the password rules, an example command specifying NATIONAL would be similar to:

```
SETROPTS PASSWORD(RULE1(LENGTH(4: 8) NATIONAL(2, 4: 6, 8)))
```

The specified required length of a password value indicates whether or not the rule of a given character position will need to be enforced. If a valid password value can range in length from four to eight characters, and a four-character password value is entered, the specified rules for character positions five to eight are not enforced. Multiple different rule values can be specified. For example, if a password value should include MIXEDVOWEL and MIXEDNUM characters, a password rule could be specified as follows:

```
SETROPTS PASSWORD(RULE1(LENGTH(6) MIXEDVOWEL(4) NATIONAL(1: 3, 5)))
```

## BE CAREFUL!

When the ability to use mixed-case alpha characters in password values is activated, there are a couple of things that you need to be aware of.

The first issue comes into play if RACF is enabled for mixed-case alpha characters (**SETROPTS PASSWORD(MIXEDCASE)**) and then is subsequently disabled for mixed-case alpha characters (**SETROPTS PASSWORD(NOMIXEDCASE)**). Again, there are two possible undesired outcomes. The first of these has to do with password verification scenarios that involve traditional online z/OS system access such as through TSO or CICS – this was already alluded to earlier in the *HOW IT'S DONE* section. With mixed-case password support disabled in RACF, password values entered in TSO or CICS are passed through to RACF in upper case. If a user ID has set their password to contain mixed-case alpha characters, they will be unable to successfully verify their password if mixed-case password support is disabled in RACF. The second problem that can occur again has to do with password rules that are set requiring a lower-case alpha character. Any potential new password value entered, regardless

of entry technique, will be set to upper case prior to RACF performing password rule option checks. This will be a problem if all the specified password rules require the inclusion of a lower-case alpha character.

There is one other interesting anomaly. RACF commands that are issued through the TSO command from an ISPF panel command line are all upper-cased prior to execution. For example, consider the following command entered by a RACF administrator from the primary command line entry area of an ISPF panel:

```
TSO ALU USERØ1 PASSWORD(Mi Xed123) NOEXPI RE
```

By the time the ALU command is executed, the parameters will have been converted to be:

```
USERØ1 PASSWORD(MI XED123) NOEXPI RE
```

This behaviour can have two possible side effects. The first problem is that, if the password value is accepted by RACF (ie it passes all the password rules), there will be an expectation that the password valued is MiXed123 when, in fact, the password value is MIXED123. The second problem is that the current password rules may be enforcing a requirement for a lower-case alpha character. If that is the case, the **ALU** command will fail, indicating that the password rules aren't being met. It will not be immediately apparent to the issuer of the **ALU** command that the password value itself has been converted to upper case. This same command entered at the TSO READY prompt or through ISPF Option 6, with **SETROPTS PASSWORD(MIXEDCASE)** active, would not have the password value characters set to upper case prior to **ALU** command execution.

## CONCLUSION

RACF supporting mixed-case passwords is a welcome update to password capability in z/OS. Besides the effective increase in available password values, sites that use password synchronization techniques can benefit from their z/OS system

being able to support synchronized passwords from other platforms. I look forward to RACF V1.8 where passwords greater than eight characters will be supported.

---

*Rudy Douglas*  
*System Programmer (USA)*

© Xephon 2007

---

## Win a 'free pass' for your next audit

In our last issue (see 'Forget *Free Willy*, how about free RACF!', *RACF Update*, issue 47, February 2007) we looked at free tools for analysis of the RACF Unload file generated by the IRRDBU00 program: RACF reporting tools from Nigel Pentland, an MS-Access database from Cory Curtis, and the IBM RACF Goodies site MS-Excel database. While these tools are great for examining the structures in your RACF database, they're only part of the total picture.

In this issue we're going to explore tools for interpreting System Management Facility (SMF) data, the essential forensic audit trail of the mainframe environment. As luck would have it, all three contributors from our last article offer similar free tools for SMF reporting, so we will start with these.

### OBTAINING THE SMF DATA

It is worth briefly covering the processes by which most mainframe installations' SMF data is generated. SMF contains a multitude of information about all aspects of running your mainframe. Data is available for all job/user initiation, system hardware/software conditions and status, as well as the security activity records we are interested in.

By default z/OS supplies three datasets for 'live' SMF; these are: SYS1.MAN1, SYS1.MAN2, and SYS1.MAN3. Most likely at your installation these names have been customized in the

current system parmlib member SMFPRMxx. You can view the dataset names in use with the MVS operator command **D SMF**. Common variations are SYS1.lpar-or-jesnode.MANx and SMF.\*.MANx. Try using SYS1.\*\*.MAN\* or SMF.\*\* in ISPF Dslist (option 3.4) if you don't have access to the above mentioned definitive methods.

In any case, although the 'live' SMF datasets above are interesting, what most audit processes are concerned with is the SMF 'dump' datasets. Typically, SMF live datasets are 'dumped', or emptied, by a started task or batch job initiated via either a system exit or automated operations. The dump datasets created by this process are usually collected into a Generation Data Group (GDG) dataset and often grouped by date or period (eg SMF.DAILY, SMF.WEEKLY, etc) to provide a history of system management data for later analysis and processing. It is also common to see post-processing of the SMF data for special interest groups, eg SMF.DAILY.CICS, SMF.DAILY.DB2, and SMF.DAILY.RACF.

As a RACF administrator or auditor, you need to understand the typical flow of SMF data described above in order to know what information is available in the selection of security-related SMF records presented to you, the end user of this automated system process. You may have to request that the current SMF processing is altered to collect additional SMF record types into the subset generated for security analysis. A recommended set of SMF record types to cover most security-related events are:

- Dataset activity records (types 14, 15, 17, 18, 62, and 64).
- JES2 spool offload records (type 24).
- Catalog activity records (types 60, 61, 63, 65, 66, 67, and 68).
- RACF processing records (types 80, 81, and 83).
- HSM function statistics records (custom record type).

For the IRRADU processing, though, we require only a few

types of SMF records – types 30, 80, 81, and 83 are used. The IRRADU utility consists of user exits for IFASMFDP (the SMF dump program). The exits IRRADU00 and IRRADU86 (see the JCL example below) select and format the relevant SMF records from the dump job. This formatting is what allows the relatively easy creation of new security reports using DFSORT and ICETOOL.

```

//*-----
//*- CREATE NEW IRRADU00 OUTPUT
//*-----
//STEP020 EXEC PGM=IFASMFDP
//SYSPRINT DD SYSOUT=*
//ADUPRINT DD SYSOUT=*
//OUTDD DD DISP=(NEW,CATLG),DSN=userid.RACF.ICE.IRRADU00, <--CHANGE
//          SPACE=(CYL,(1000,500),RLSE), <--CHANGE
//          DCB=(DSORG=PS,RECFM=V)
//SMFDATA DD DISP=SHR,DSN=SYS1.SMF.DAILY.DUMP(-1) <--CHANGE
//SMFOUT DD DUMMY
//SYSIN DD *
          INDD(SMFDATA,OPTIONS(DUMP))
          OUTDD(SMFOUT,TYPE(000:255))
          ABEND(NORETRY)
          USER2(IRRADU00)
          USER3(IRRADU86)

```

The job above can be executed against both raw SMF datasets (ie the live SMF datasets) or SMF dump datasets previously unloaded during normal SMF processing. This job produces SMF records in a readable format that are then easily parsed by other tools. See below for example output from IRRADU SMF dump exits (records are truncated):

ACCESS	SUCCESS	10:20:51	1995-11-06	IM13	NO	YES	NO	I EESYSAS	JESXC
DEFINE	SUCCESS	10:47:25	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
DELRES	SUCCESS	10:47:51	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
RDEFINE	SUCCESS	11:14:08	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
PERMIT	SUCCESS	11:14:34	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
SETROPTS	SUCCESS	11:14:46	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
DELDS	SUCCESS	14:01:52	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
DELUSER	SUCCESS	14:01:54	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
DELGROUP	SUCCESS	14:01:59	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
ADDUSER	SUCCESS	14:02:00	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
ADDGROUP	SUCCESS	14:02:04	1995-11-06	IM13	NO	NO	NO	MARKN	SYS1
JOBINIT	INVP SWD	14:02:08	1995-11-06	IM13	YES	NO	NO	MARK	SYS1
CONNECT	SUCCESS	14:09:59	1995-10-04	IM13	NO	NO	NO	RRSFU2	SYS1
PASSWORD	INSAUTH	14:10:03	1995-10-04	IM13	YES	NO	NO	RRSFU2	SYS1

PERMIT	SUCCESS	14:10:07	1995-10-04	IM13	NO	NO	NO	RRSFU2	SYS1
DEFINE	ALRDEFD	14:10:16	1995-10-04	IM13	YES	NO	NO	RRSFU2	SYS1
SETROPTS	INSAUTH	14:10:20	1995-10-04	IM13	YES	NO	NO	RRSFU2	SYS1

So now we have some SMF data ready for analysis with the various tools at our disposal. For this article I am using the supplied sample IRRADU output from the RACFICE tools, available from the RACFICE ftp server in TSO XMIT format as filename 'racfice.sampadu.xmit'. Simply transfer this file to your z/OS system using binary FTP to a pre-allocated dataset with RECFM=FB and LRECL=80. The file occupied just under 140 tracks (3390) when I tested this. Unpack the file to a dataset of your choice using the **TSO RECEIVE** command:

```
TSO RECEIVE INDA('uploaded-file')
```

Respond to the prompt with DA('userid.RACFICE.IRRADU00') or your preferred dataset name.

I expect that at your installation you would be using 'real' SMF data rather than the sample data I've used here, the JCL in the first example should be customized to match your SMF input data, your chosen dataset name for the IRRADU output data, and an appropriate size for the output dataset. The IRRADU dataset (OUTDD in the example) is the input data for the reporting jobs covered in the remainder of this discussion.

## NIGEL PENTLAND'S RACFICE JOBS

Nigel provides three sets of SMF reports on his Web site at [www.racf.co.uk](http://www.racf.co.uk). Although some overlap exists between each set and also with the IBM-supplied default RACFICE reports, these reports provide a comprehensive overview of SMF activity requiring little or no customization to get working in your z/OS environment.

The three files provided are:

- racfsmf.txt – a set of RACF action reports.
- moresmf.txt – additional RACF action reports.

- audtrpts.txt – a comprehensive set of RACF reports incorporating the previous two sets.

These files contain JCL and DFSORT/ICETOOL control cards to produce the various reports. You can download the files from Nigel's site and copy them to any JCL dataset member on your z/OS system. You will need to adjust the jobcards as well as the IRRADU00 input DD to be that of the IRRADU output dataset we generated from raw SMF above. In audtrpts.txt is a sample IRRADU00 SMF dump job, which I removed for my testing because I already had valid IRRADU data available.

Nigel gives credit to the many contributors who assisted him in creating these reporting suites. The audtrpts.txt sample is a collection of reports from the other samples, re-written in such a manner as to eliminate multiple passes of the SMF data. After running the three sample report sets, I agree that the audtrpts.txt seems to be the most comprehensive and the most efficient.

The list of reports produced by audtrpts.txt is:

- ACCESS VIOLATIONS
- ADDSD COMMANDS
- ADDGROUP COMMANDS
- ADDUSER COMMANDS
- ALTDSD COMMANDS
- ALTGROUP COMMANDS
- ALTUSER COMMANDS
- CONNECT COMMANDS
- DELDSD COMMANDS
- DELGROUP COMMANDS
- DELUSER COMMANDS
- PASSWORD COMMANDS
- PERMIT COMMANDS
- RALTER COMMANDS
- RDEFINE COMMANDS
- RDELETE COMMANDS
- REMOVE COMMANDS
- RVARY COMMANDS
- SETROPTS COMMANDS

- USERIDS WITH EXCESSIVE INCORRECT PASSWORDS
- USERIDS WITH EXCESSIVE ACCESS VIOLATIONS
- RESOURCES WITH EXCESSIVE ACCESS VIOLATIONS
- RESOURCES WITH EXCESSIVE ACCESS RECORDS
- RESOURCES WITH EXCESSIVE ACCESS RECORDS BY USERID.

Slight differences exist between the audtrpts.txt report set and the report sets generated by the other two jobs, an example being the distinction between ADDSD and DEFINE type records in the IRRADU00 output. For most practical purposes these records contain the same data and reporting on both is redundant. There is provision for reporting on both in the audtrpts.txt job; however, the duplicate report is not executed by default in the current version.

If you don't already have some regular RACF activity reporting, this suite is comprehensive, very easy to implement, and provides a great start to monitoring security activity within your z/OS system.

### CORY CURTIS IRRADU00 DATABASE

The SMF database available from Cory's Web site (<http://racf.curtistree.com/>) must be initialized with definitions from SYS1.SAMPLIB(IRRADULD) before its first use. This samplib member contains DB2 load statements that are read to determine the correct structure for the MS-Access database. After initialization of the database, the output of the IRRADU SMF dump job is transferred to a PC where the MS-Access database can import it for offline analysis. These instructions are covered in detail on Cory's Web site and reproduced in brief here:

- 1 Download and unzip the racf.zip file to a convenient directory. This contains the Access database.
- 2 File transfer your current SYS1.SAMPLIB(RACDBULD) dataset to C:\temp\temp\irraduld.txt.
- 3 Open the SMF.mdb Access database and run the 'First time run' macro.



- 4 File transfer IRRADU SMF dump output to C:\temp\temp\Data.txt.
- 5 Again in the SMF.mdb database, run the 'One Step Import' macro.

This process is identical to that used in Cory's RACF unload file MS-Access database. The 'First time run' macro must be executed against a fresh copy of the IRRADULD samplib member whenever your z/OS system is upgraded and all file transfers must be done as ASCII text. The file names for the IRRADULD and the SMF data are coded within the MS-Access macros, but can be altered if desired. For my testing I used the sample IRRADU output from the IBM RACFICE package – after transferring this to the mainframe and unpacking the XMIT file, I simply FTPed this data as ASCII text back to my PC.

Following these steps I had a working MS-Access copy of the SMF data within minutes. Once again Cory has created a very useful analysis tool for those comfortable with using MS-Access. There are a large number of tables available that represent all kinds of security-related events, including many OMVS-related event types such as filesystem mounts/unmounts, use of Extended Access Lists, and UID/GID use within the Unix filesystem. This detailed information is not readily available in the other SMF processing tools discussed here.

Unfortunately, many of these tables are not populated by the RACFICE provided sample data. The tables that are populated, though, interpret the entire IRRADU record and include many fields that are not displayed in the reports produced by the DFSORT- and ICETOOL-based utilities.

If you need to create custom reports using potentially all the available SMF fields, this relational database utility would be very useful. Similarly, it is far easier to create a truly relational query incorporating the results from more than one table (ie ADDSD and DEFINE can be 'merged' into one comprehensive record) using MS-Access than DFSORT and ICETOOL.

The availability of record types not currently processed by the

other utilities reviewed here makes Cory's MS-Access database a highly-attractive free auditing tool.

## IBM RACF GOODIES' SITE RACFICE REPORTS

IBM's RACFICE is the new RACF Report Writer. There is even a direct comparison of RACF Report Writer functions against equivalent RACFICE reports within the RACFICE documentation. IBM's vision is to use generic report writing tools already available at all z/OS installations rather than continue to enhance the RACF Report Writer to process the regularly updated list of SMF records and sub-types – hence the SMF dump exits IRRADUxx were created.

RACFICE can be downloaded directly from the IBM RACF FTP server at: <ftp://ftp.software.ibm.com/eserver/zseries/zos/racf/racfice/> – the RACFICE Web site recommends using the copy provided in SYS1.SAMPLIB(IRRICE) because this is guaranteed current with your installed version of z/OS. However, on examining the supplied SAMPLIB member and discovering that it contains IEBUPDTE statements to build the final jobstream(s), I elected to upload the racfice.xmit PDS supplied on the IBM RACF Web site instead. More years ago than I care to remember, I forgot how to use IEBUPDTE, and nowadays always have to look up this cryptic, albeit extremely useful, tool whenever it is needed. Being a person who prefers the path of least resistance, I decided it was prudent to at least check out the xmit file from the RACFICE pages first.

Transfer this dataset as usual using binary FTP to an FB(80) pre-allocated dataset, then use the **TSO RECEIVE** command to unpack to a destination PDS of your choice. On examining the output PDS you will find over 70 members, the most important of which is the \$\$CNTL\$\$ member. Edit this member to have a valid jobcard and ensure that the three JCL SET statements point to the IRRADU00, IRRDBU00, and the JCL library just created respectively. Similarly, point the JCLLIB statement to the just-created JCL library.

With these minor changes I was able to produce the complete

set of RACFICE reports in only a few minutes – just submit the \$\$\$CNTL\$\$\$ member and browse its output. However, after examining the reports generated, I realized that the RACFICE version available on the RACF Web site is not as current as the version in my SYS1.SAMPLIB(IRRICE) member – so back we go to IEBUPDTE.

A quick reference to the *DFSMSdfp Utilities* guide gave me a rough IEBUPDTE job to work with. The JCL below can be used to unpack the jobstream supplied in SYS1.SAMPLIB(IRRICE) to a new JCL library of your choice:

```
//USERØ1 JOB ( ), ' ', CLASS=A, MSGCLASS=X,  
//          NOTIFY=&SYSUID  
//STEP1 EXEC PGM=IEBUPDTE, PARM=NEW  
//SYSPRINT DD SYSOUT=A  
//SYSUT2 DD DSNNAME=userid.RACFICE.CNTL, DISP=(,CATLG), <---CHANGE  
//          SPACE=(TRK, (2Ø, 1Ø, 1Ø)),  
//          DCB=(RECFM=FB, LRECL=8Ø)  
//SYSIN DD DSN=SYS1.SAMPLIB(IRRICE), DISP=SHR
```

Once this job is complete, edit the library created and apply exactly the same changes mentioned above to the \$\$\$CNTL\$\$\$ member. You can then submit the \$\$\$CNTL\$\$\$ JCL and generate your reports – there are 38 reports provided by default with the version I tested on a z/OS 1.4 system:

- ALDS – discrete dataset profiles that have IDs on the standard access list with ALTER authority.
- ASOC – users who have explicit associations defined.
- BGGR – discrete general resource profiles with generic characters in their name.
- CCON – count of user connections, flagging those with more than x connections.
- CGEN – count of general resource profiles.
- CPRO – count of profiles.
- CONN – user IDs with group privileges above use.
- GIDS – shared Unix System Services GIDs.

- IDSC – dataset conditional access lists with ID(\*) of other than NONE.
- IDSS – dataset standard access lists with ID(\*) of other than NONE.
- IGRC – general resource conditional access lists with ID(\*) of other than NONE.
- IGRS – general resource standard access lists with ID(\*) of other than NONE.
- OMVS – user IDs that have Unix System Services (OMVS) segments.
- PCAM – PROGRAM class specific profiles with MAIN or BASICAPPLDATA.
- SUPU – Unix System Services super users (UID of zero).
- UGLB – user IDs with extraordinary system-level authorities.
- UGRP – user IDs with extraordinary RACF group authorities.
- UIDS – shared Unix System Services UIDs.
- URVK – user IDs that are currently revoked.
- UADS – dataset profiles with UACCs of other than NONE.
- UAGR – general resource profiles with UACCs of other than NONE.
- WNDS – dataset profiles in WARNING mode.
- WNGR – general resource profiles in WARNING mode.
- ACD\$ – users who are using automatic command direction.
- CADU – count of IRRADU00 events.
- CCMD – count of commands issued (by user).
- ECD\$ – users who are directing commands explicitly.
- LOGB – users who log on with LOGON BY.
- LOGF – all users with excessive incorrect passwords.

- OPER – accesses allowed because the user has OPERATIONS authority.
- PWD\$ – users who are using password synchronization.
- RACL – RACLINK audit records.
- RINC – RACF class initialization records.
- SELU – all audit records for a specific user.
- SPEC – events that succeeded because the user has SPECIAL authority.
- TRMF – excessive incorrect passwords from terminals.
- VIOL – access violations.
- WARN – accesses allowed due to WARNING mode profiles.

Remember that many of these reports use the RACF unload file (IRRDBU00) as well as the SMF data (IRRADU00), so for the best results it's important to have both sources of security information up-to-date before you run these reports.

Once you've mastered IEBUPDTE again, RACFICE is easy to use and produces a good range of reports. Comparing the reports produced by IRRICE against the list produced using Nigel's audtrpts.txt example reveals that there is only a slight overlap between these two sets of reports, mainly in the access violations and excessive passwords derived reports. Of course, RACFICE also analyses the RACF flat file IRRDBU00 and produces many reports from this data that are not available in the alternatives reviewed here. Overall, I would be tempted to use both sets of reports for a comprehensive overview of security.

## CONCLUSIONS

Similar to the utilities reviewed in the last issue, I find the table names created by utilities that rely on IBM's DB2 Load statements – Cory's SMF.MDB in this case – to be a little cryptic and difficult to understand without a good knowledge of the source

data and record and field definitions. The format of records produced by the SMF unload utility IRRADU00 is thoroughly documented in the current *RACF Macros and Interfaces* manual, but there is a lot of information to absorb in there, and to me seems a shame that more 'intuitive' table and field names cannot easily be used.

Despite this, of the three utilities reviewed here, my personal preference has to go with Cory Curtis' MS-Access database. Although it supplies no canned reports that can be run immediately against your SMF data, it does produce the most comprehensive coverage of SMF events by virtue of using the IBM-supplied (ie current) record and field definitions in order to create its table definitions. If you need a comprehensive review of the contents of your SMF data this is a great tool.

The main advantage of the other two tools looked at here are the supplied reports, getting you off to a fine start towards a set of audit reports relevant to your installation. As stated previously, I would tend to use a combination of Nigel's supplied reports and the base IRRICE reports that IBM supplies.

In the next issue we will continue to find new sources of data to increase our awareness of security-related activity and our ability to report on this. Specifically we will take a look at methods of analysing the Unix System Services Hierarchical File System.

---

*Michael Cairns (mike.cairns@arialgroup.com)*  
*Responsible for mainframe products and operations*  
*Arial Group International (Australia)*

© Xephon 2007

---

## **Overview of RACF enhancements in z/OS V1R8**

### **INTRODUCTION**

No matter how good RACF is, there's always room for improvement. And there are numerous changes made in each

version that make RACF more secure and more usable. In z/OS V1R8 there are a number of changes to increase the security and usability of RACF, which we shall see in this article.

In z/OS Version 1 Release 8, z/OS continues to deliver industry leadership for security. Improvements that are all intended to help deliver the kind of security-rich environment that has made z/OS an industry leader include:

- Significant improvements to Identrus-certified support for digital certificates, including SCEP and multiple-CA support.
- Support for defining Intrusion Detection Services (IDS) policies in a policy agent configuration file as well as an LDAP server. This solution provides an IDS policy solution that is consistent with other policy types for those installations that do not have an LDAP infrastructure in place or that prefer using configuration files instead of LDAP.
- RACF support for password phrases from 14 to 100 characters in length, in addition to the current support for passwords. Password phrases allow for an exponentially greater number of possible combinations of characters and numbers than do passwords.
- Public Key Infrastructure (PKI) Services support for multiple certificate authorities (CA), including the ability to establish multiple certificate authorities on a single image; and Simple Certificate Enrollment Protocol (SCEP) support.
- New options for securing tape datasets using the System Authorization Facility (SAF), to allow you to define profiles to protect datasets on tape using the DATASET class without the need to activate the TAPEDSN option or the TAPEVOL class, to allow you to specify that all datasets on a tape volume should have common authorization, and to allow you to specify whether users are authorized to overwrite existing files on a tape volume.
- Support for the Advanced Encryption Standard (AES) algorithm for IP security with a 128-bit key length.

- Support for SAF identity tokens. The support for SAF Identity Tokens provides exploiters with increased user accountability and auditability of resources by providing end-to-end auditing that tracks identities used for initial authentication and those used on the current platform.
- There is RACF support for virtual key rings. This support treats the collection of all the certificates owned by one user ID, including the SITE and CERTAUTH reserved user IDs, as an independent key ring. The use of the CERTAUTH virtual key ring is intended to help eliminate the need to manually create multiple real key rings for SSL-enabled z/OS client applications such as FTP.

## PASSWORD PHRASE SUPPORT

In z/OS V1R8, profile extension focuses on improving or making more flexible and relevant the security rules as implemented by RACF today. Password phrase support and custom fields support both lead toward this goal. Password phrase provides an alternative to traditional passwords. This alternative, a password phrase, allows a much longer value than a password as well as a larger character set. A password phrase is a character string made up of mixed-case letters, numbers, and special characters (including blanks). The intention of the password phrase is to have something that is long enough to provide security, but easy enough to remember so it won't be written down. It is unlikely that a random string of characters longer than eight characters can be memorized easily, therefore, a password phrase is more likely to be made up of words. Adding numbers and special characters makes the phrase more secure. RACF now supports password phrases from 14 to 100 characters in length, and enforces a basic set of rules to increase the strength of the password phrase. Since a user ID can have both a password and a password phrase, the same user ID can be used both for traditional applications that accept a password and for new applications that take advantage of the password phrase infrastructure.



## NEW RACF HEALTH CHECKS

RACF provides the following new Health Checks:

- Examine additional key security controls on the system, such as the current link list libraries, the current parmlib datasets, and key general resources.
- Examine the IBMUSER user ID and ensure that it is revoked.
- Examine key classes (such as TAPEVOL) and ensure that the classes are active.
- Support the verbose flag, which when set to ON causes the RACF\_FRS\_RNL check to list all of the ENQ names (just like the DEBUG option does currently).
- Allow installations to define their own resources to a RACF-supplied check.

## IMPROVEMENTS TO AUTOMATIC REVOKING OF USERIDS

Did you ever set the SETROPTS INACTIVE option and then find that an old unused user ID was still active? That happened because RACF used to set the last access date to UNKNOWN when it created a new user ID. And, as long as the user ID was not used, it would never be revoked. In the new version, RACF sets a new user ID's last access date to its creation date. Now, those old, unused user IDs will be automatically revoked if they aren't used before their revoke dates.

## AUTOMATIC SWITCH TO BACK-UP DATABASE WHEN I/O ERRORS OCCUR

There's an I/O error on your RACF database, and suddenly the console is flooded with error messages. The operator needs to do an **RVARY SWITCH** to activate the back-up database, but it's difficult to enter the command with all the messages coming in and to identify where the **RVARY** password was put. It can take valuable time to find the password and enter the command.

Now you don't need to; in the new version, RACF will automatically switch to the active back-up database when an I/O error is detected on the primary RACF database and the UCB indicates that the device is varied offline – and nobody needs to enter a password.

## ENHANCEMENTS TO IRRUT200 AND IRRUT400

RACF has made availability improvements in z/OS V1R8 that can help to identify and correct problems that have resulted in past system outages. The problems addressed in this release are:

- Clients needing to create a consistent back-up copy of the primary RACF dataset and activate the back-up without loss of synchronization between the two datasets. In the past, IRRUT200 did not allow this; the primary RACF database could be updated between the copy process and the activation of the back-up database, thus resulting in an error. IRRUT400 with the LOCKINPUT option could create a consistent RACF back-up database. However, it could cause applications attempting to access the locked database to fail.
- Simple errors in the DD statements for IRRUT200 or IRRUT400 could result in an active RACF dataset being an output dataset for these utilities. The utilities should detect this when possible and prevent the overwriting of an active RACF dataset.

To solve these problems, RACF has made the following availability improvements:

- A new parameter on the IRRUT200 utility will tell the utility to activate the back-up dataset pointed to as output. This will be accomplished by the utility internally issuing an **RVARY ACTIVE** for the back-up dataset after the copy is complete.
- IRRUT200 and IRRUT400 utilities will check whether their

output datasets are active primary or back-up RACF datasets on this system. If so, the utility will fail with an error message.

## LDAP CHANGE LOG

RACF currently supports event notification if a USER profile changes using the z/OS LDAP change log. Group change logging extends this support to GROUP profiles, including group connection information. This closes a functional gap such that all RACF information, which can be managed using the LDAP SDBM back-end, will not participate in the LDAP change logging function. In addition, two minor functional gaps were also closed with regard to the related password enveloping function, as follows:

- First, LDAP change log entries are now created for any password change, not just those that were enveloped.
- Also, the LISTUSER command has been enhanced to report on the presence of a password envelope for users. Previously, an administrator had to run the RACF database unload utility (IRRDBU00) to obtain this information.

## PKI ENHANCEMENTS

In z/OS R8, PKI has the following enhancements:

- Virtual key rings – in the past, SSL-enabled applications required the creation of a key ring for each unique user ID along with any CA certificates. This situation propagated, when in most cases the same set of CA certificates would be used for each user ID. Thus these key rings would all be replicas.

To solve this problem, RACF now treats all the certificates installed under a given user ID as a virtual key ring. This key ring is created when the user ID is added and destroyed when the user ID is deleted.

- Enabling multiple CA support for PKI services – this function now lifts the restriction preventing more than one instance of the PKI services daemon from being started simultaneously on a single MVS image. This allows PKI services customers to establish multiple certificate authorities on a single MVS image.
- Adding SCEP support to PKI services – Simple Certificate Enrolment Protocol (SCEP) allows SCEP-enabled clients to request certificates by sending messages to a certificate authority using the HTTP protocol. Adding SCEP support allows PKI services to accept, decrypt, and respond to the various SCEP messages and supports both the manual and automatic enrolment modes as defined in the standard. Manual enrolment requires the CA administrator to manually approve requests. With automatic enrolment, certificate requests are auto-approved and fulfilled synchronously, based on the requestor's knowledge of a predetermined secret, the challenge password.

## SAF IDENTITY TOKEN

The SAF identity token now provides increased user accountability and audit resources by providing end-to-end auditing that tracks the identity initially used for authentication as well as the identity on the current platform. This support is especially valuable to customers maintaining heterogeneous environments, where requests and entry points to network resources come from a variety of platforms.

## z/OS DB2 VERSION 8 SUPPORT

RACF supplies a security exit which enables SAF/RACF services to be utilized for DB2 security, effectively allowing DB2 security to be centralized within the scope of the system security administrator. RACF now extends the existing support to provide conditional access support, which enables the use of the DB2 role to authorize a user to a RACF-protected DB2 object. Further, in support of DB2's trusted context, the identity, which

is conveyed to z/OS DB2, may not be a z/OS RACF-defined user. RACF provides an identity-mapping plug, which enables z/OS DB2 foreign users to a z/OS RACF-defined identity that is needed for the resolution of access control decisions. This function provides RACF security support to the DB2 V8 community by using the new trusted channel role.

## REMOTE AUTHORIZATION AND AUDITING

RACF now provides the ability to process remote authorization and auditing requests at the z/OS security server. They will employ a standard LDAP protocol to enable requests from various system platforms, ensure an auditable level of trust between the z/OS security server and requesting applications, and use the familiar SMF logging capability for recording audit data, and support the unloading of that data. Customers increasingly look for value in a centralized directory such as z/OS LDAP to locate and manage people, objects, and services for the enterprise. Middleware application hosting environments WebSphere and Tivoli have embraced LDAP for that purpose. Leveraging these new z/OS LDAP extended operations, middleware and customer applications can achieve distributed authorization and auditing function that can be managed by the z/OS security administrator.

## IRRSDA00 ENHANCEMENTS

RACF provides an enhancement to IRRSDA00 to support using RACROUTE REQUEST=FASTAUTH when the RACF profile is RACLISTed. The current CIM Server uses `_check_resource_auth_np` to make sure that the CIM client can access CIM Server. The existing `_check_resource_auth_np` logic translates to USS invoking IRRSDA00 using the RACROUTE REQUEST=AUTH function.

The RACROUTE REQUEST=AUTH function is a long pathlength function. Since the CIM Server RACF profile is RACLISTed, changing IRRSDA00 to recognize the RACLISTed profile to do

a FASTAUTH check will have performance benefits for all `_check_resource_auth_np()` invokers checking on RACLISTed profiles.

The IRRSDA00 has added logic to create an ACEE for the input user ID for the FASTAUTH check on the RACLISTed profiles. This change gives better performance on `_check_resource_auth_np` when the RACF profile is RACLISTed.

## MINIMUM PASSWORD CHANGE INTERVAL

RACF allows a security administrator to set a maximum password interval, requiring users to change their passwords periodically. However, some users want to use the same password all the time. Previously, to circumvent the password history, when a favourite password expires, these users change that password multiple times, until the original password is no longer in the password history, and then change back to the original password. Security administrators have asked for the ability to set a minimum password interval, to prevent this circumvention. The new MINCHANGE option on the **SETROPTS PASSWORD** command allows a security administrator the ability to specify the minimum number of days between password changes. The minimum change interval is enforced when a user changes his or her own password with the **PASSWORD** command, **ALTUSER** command, or RACROUTE REQUEST=VERIFY macro. Authorized users can set another user's password before the interval has passed, but they must have CONTROL authority if they are authorized based on the IRR.PASSWORD.RESET resource in the FACILITY class.

## SMF LOGGING OF PASSWORD CHANGES

With SETROPTS AUDIT(USER) in effect, password changes made by RACROUTE REQUEST=VERIFY are now audited, in addition to password changes made using commands. Now auditors and security administrators can always find out when and how a user's password was changed.

## PASSTICKET ENHANCEMENTS

A callable service interface has been added that supports problem state callers for PassTicket generation and evaluation services. With the `r_ticketserv` interface, you can now use PassTicket functions for 31-bit callers. The `r_gensec` callable service now supports 64-bit callers. Also added is a new Java interface, using a Java Native Interface (JNI), which calls the updated `r_ticketserv` and `r_gensec` callable services, allowing Java code to easily access PassTicket services running on z/OS. These changes make it easier to use PassTicket.

## NESTED ACEES AND DELEGATED RESOURCES

A daemon is a highly-privileged Unix program that processes requests on behalf of clients. The daemon creates a new address space in which the security environment is that of the client. Once the new address space has been created, there is no longer a relationship between the daemon and the client. This means that the client needs to have authorization to all resources that the daemon uses. Enter the nested ACEE. An Access Control Environment Element (ACEE) is a control block describing the user's security environment. A nested ACEE associates a client identity with the daemon that spawned it by 'nesting' the daemon identity within the security environment created for the client. When a nested ACEE is used in an authorization check, if the client check fails, RACF checks the daemon's authorization. Applications can create nested ACEEs using a new keyword, `NESTED`, on the `RACROUTE REQUEST=VERIFY, ENVIR=CREATE` macro. By using nested ACEEs, applications can remove the need to permit large numbers of users to highly-sensitive RACF-protected resources. Unconditionally honouring a nested ACEE might inappropriately grant client access to an unintended resource. Therefore, only certain resources honour nested ACEEs in authorization checks. These resources are referred to as delegated resources. The security administrator designates a resource as delegated by placing the string '`RACFDELEGATED`' in the `APPLDATA` field of the RACF profile protecting the resource.

## SUMMARY

This release of RACF in z/OS V1.8 continues to deliver industry leadership for security. Improvements are all intended to help deliver the kind of security-rich environment that has made z/OS an industry leader. Some of the enhancements include RACF support for virtual key rings, and RACF template extensions that allow templates to expand beyond their current 4K sizes. RACF now supports the use of passwords longer than eight characters, now called password phrases. The RACF access control module exit, DSNXRAC, has changed substantially with DB2 Version 8. RACF administrators can now define a security rule before an object is created and preserve the rule for a dropped object. In addition, RACF general resources for member/grouping profiles can be utilized by an installation to protect multiple DB2 resources with a single RACF profile. A new parameter on the IRRUT200 utility tells the utility to activate the back-up dataset printed to as output. This is accomplished by the utility internally issuing an **RVARY ACTIVE** for the back-up dataset after the copy is complete. IRRUT200 and IRRUT400 utilities now check whether their output datasets are active primary or back-up RACF datasets on this system. With z/OS V1R8, RACF introduces new health checks.

---

*Laxminarayan Sriram*  
*Systems Programmer (USA)*

© Xephon 2007

---

## Automatic assignment of Unix UIDs or GIDs

### CONTROL UNIX SHARED IDENTITIES

Before you use automatic ID assignment, you must activate the control of shared Unix identities.



## IDs or GIDs shared

RACF permits the sharing of UIDs and GIDs among any number of users or groups. However, by defining a profile SHARED.IDS in the UNIXPRIV class, the unique Unix identifiers can be controlled.

In order to perform this operation, Application Identity Mapping (AIM) must be implemented. If the RACF database is not at least at AIM stage 2, advance it using the IRRIRA00 utility.

Refer to the *z/OS Security Server RACF System Programmer's Guide* for more information on using the IRRIRA00 utility to advance to AIM stage 2.

## Remarks

If shared UIDs and GIDs already exist in your RACF database, make an effort to minimize their use.

To use shared IDs (GIDs) means, for example, that user1, user2, user3, and user*n* have the same identity when they use the Unix environment.

For example, from the RACF events report we discover that the ID 12345 has cancelled the filesystem */Fpname*, but which of the users (user1, user2, or user*n*) executed this operation?

This way of working is counter to the first rule of the RACF, which says, 'one person, one user, only one identity'.

## DEFINE SHARED.IDS

The SHARED.IDS profile must be defined in the UNIXPRIV class, which must be RACLISTed. The administrator needs to refresh the class after defining or altering the SHARED.IDS profile.

The following commands define a SHARED.IDS profile:

```
RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

The first command defines the profile SHARED.IDS in the UNIXPRIV class. The second command raclists the UNIXPRIV class, and the third refreshes the class.

All RACF commands can be activated online or in batch.

### Online

Go to TSO option 6 (command) and type the three RACF commands:

```
RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
```

then:

```
SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)
```

then:

```
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

### Batch

You can use the following JCL:

```
//UseridCM jobcard parameters
//STEP011 EXEC PGM=IKJEFT01
//SYSLBC DD DSN=SYS1.BROADCAST,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
  PROFILE NOPREFIX
  RDEFINE UNIXPRIV SHARED.IDS UACC(NONE)
  SETROPTS CLASSACT(UNIXPRIV) RACLIST(UNIXPRIV)
  SETROPTS RACLIST(UNIXPRIV) REFRESH
/*
```

Once you have defined this profile, any attempt to assign an ID already in use fails. Now, in our RACF database, we can't have two users with the same UID or GID, and if we wish to make an exception, the SHARED operand must be used.

### USING THE SHARED OPERAND

If we wish to make an exception and create a shared UID, the SHARED operand must be used.

For example:

```
ADDUSER SUPERUS OMVS(UID(30) SHARED HOME(/))  
ADDGROUP MYGROUP OMVS(GID(77) SHARED)
```

The first command adds to the RACF database the SUPERUS user and assigns it the UID number 30. The second command adds the group MYGROUP and assigns it the group ID number 77.

To specify the SHARED operand, it is necessary to have the SPECIAL attribute or at least READ authority to the SHARED.IDS profile in the UNIXPRIV class.

For example:

```
PERMIT SHARED.IDS CLASS(UNIXPRIV) ID(USSUADM) ACCESS(READ)  
SETROPTS RACLIST(UNIXPRIV) REFRESH
```

The **PERMIT** command adds the user USSUADM with READ access to the SHARED.IDS profile. Now the user is authorized to specify the SHARED parameter during OMVS segment creation.

If specified, the SHARED operand is ignored when:

- The SHARED.IDS profile is not RACLISTed.
- The UID or GID operand is omitted.
- The specified UID or GID value is unique.
- The specified UID or GID value is identical to the current UID or GID value.

## ENABLING AUTOMATIC ASSIGNMENT OF UNIX IDENTITIES

RACF can automatically generate a unique ID value in the OMVS segment by defining a profile BPX.NEXT.USER in the FACILITY class and then specifying:

- The AUTOUID operand of the **ADDUSER** and **ALTUSER** commands.
- The AUTOGID operand of the **ADDGROUP** and **ALTGROUP** commands.

For example:

```
ADDUSER USER1 OMVS(HOME(/u/user1) PROGRAM(/bin/sh) AUTOUID)
ALTUSER USER2 OMVS(AUTOUID)
ADDGROUP USER3 OMVS(AUTOGID)
```

After the command completes, an informational message is issued to indicate the assigned value.

For example:

```
IRR52177I User USER1 was assigned an OMVS UID value of 6233.
```

For the **ALTUSER** and **ALTGROUP** commands, **AUTOUID** and **AUTOGID** cannot be used to change the ID value if one exists for the user.

It is not possible to use automatic assignment (**AUTOUID**) with a list of users or groups; the command will fail.

For example, the following command is wrong and will not work:

```
ADDUSER (USER1 USER2 USER3) OMVS(AUTOUID)
```

Remarks:

- Implementing **SHARED.IDS** and **BPX.NEXT.USER** is a prerequisite to completing successful automatic ID assignment.
- The **AUTOUID** and **AUTOGID** operands cannot be specified with the **SHARED** operand.
- **AUTOUID** is ignored if **UID** or **NOUID** is specified.
- **AUTOGID** is ignored if **GID** or **NOGID** is specified.

## DEFINE THE BPX.NEXT.USER PROFILE

**BPX.NEXT.USER** is a **FACILITY** class profile that is used by **RACF** to calculate unused **UID** and **GID** values. The **FACILITY** class does not have to be active for **RACF** to use **BPX.NEXT.USER**.

The **APPLDATA** field contains the starting **UID** or **GID** value or range of values separated by a forward slash (/). The starting

value is the value RACF attempts to use in ID assignment, having determined that the ID is not in use. If it is in use, the value is incremented until an appropriate value is found.

For example, if in your RACF database you start now to assign the user UID or GID, to have RACF start automatic assignment with a UID value of 1 and a GID value of 0, execute this command:

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(' 1/0' )
```

The maximum value is 2,147,483,647.

The starting value used is your choice. For example, if UID values 0–3000 are already in use, and GID values 0–300 are already in use, you should use a UID starting value of 3001 and a GID starting value of 301:

```
RALTER FACILITY BPX.NEXT.USER APPLDATA(' 3001/301' )
```

Specifying NOAUTO as a qualifier in the APPLDATA, or omitting the qualifier, prevents automatic ID assignment.

For example, if you don't want to use automatic UIDs assignment, but only the automatic GID assignment starting at 4000, use the following command:

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(' NOAUTO/4000' )
```

RACF also accepts range values. If you want to include a range of values, specify a start and end value separated by a dash (-). Both values must be valid UID or GID values and the second must be greater than the first.

Ranges can be specified independently for UIDs or GIDs. For example:

```
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(' 60000-70000/4000-20000' )  
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(' 60000/4000-20000' )  
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(' 60000-90000/4000' )  
RDEFINE FACILITY BPX.NEXT.USER APPLDATA(' NOAUTO/4000-20000' )
```

The first command activates the automatic UID creation starting from value 60000 to value 70000 and the automatic GID creation from value 4000 to value 20000.

The second command activates the automatic UID creation starting from value 60000 without an end value and the automatic GID creation from value 4000 to value 20000.

The third command activates the automatic UID creation starting from value 60000 to value 90000 and without the automatic GID creation.

The fourth command activates only the automatic GID creation, but the UID must be specified later to utilize the first UID number available.

Remarks:

- You cannot specify blanks in the APPLDATA string.
- Syntax checking of APPLDATA does not occur until AUTOUID and AUTOGID operands are specified on the **ADDUSER**, **ALTUSER**, **ADDGROUP**, and **ALTGROUP** commands.
- If you have defined BPX.NEXT.USER with incorrect APPLDATA, issuing AUTOUID or AUTOGID fails with message IRR52187I being issued.
- You can change the APPLDATA values at any time.
- After successful automatic ID assignment, RACF updates the APPLDATA starting value with either the next potential value or end of range.

---

*Magni Mauro*  
*System Engineer (Italy)*

© Xephon 2007

---

## **Pentland Utilities V2.0 – an update**

*Editor's note: RACF Update ran a two-part series in February/May 2003 that examined the Pentland Utilities in detail. Doc Farmer takes a look at some major improvements and additions made to this suite of free utilities.*

Four years is a long time for any software package, but *a fortiori* for a program linked to the growth and idiosyncrasies of a mainframe security suite like RACF. Such are the Pentland Utilities. I know that Mike Cairns wrote about some of the utilities in the last issue, but as luck, timing, and Murphy's Law would have it, Nigel Pentland released a new version of his suite of PC-based RACF utilities just after the February 2007 *RACF Update* ended up in your inboxes. Not just a new version, but a major release.

When last we reviewed this package, Nigel was on release 1.10. However, on 11 February 2007, the Pentland Utilities took a paradigm shift to Version 2.00. (By the way, there is a lot of confusion about the definition of 'paradigm'. It's an American term meaning 20 cents...). At this writing, Nigel has performed two more updates to Version 2.02. When I wrote the two-parter in 2003, there were 72 utilities. Now there are 106 – an increase of almost 50%. In the intervening quadrennial, IBM came up with much more varied functionality in the areas of USS, cryptography, and certificate handling. The Pentland Utilities address these, especially the latter, with 13 very detailed report and JCL generators in place.

The documentation for the Pentland Utilities also made a leap forward (in more ways than one) with the development of a new Adobe Acrobat-based manual. It has a table of contents, an index, keywords, bookmarks, verbs, all that fancy stuff. When 2.00 came out, however, there was one thing that seemed to be missing. Since the Table of Contents was arranged by grouping the utilities by function (Miscellaneous, Groups, Users, Datasets, General Resources, and Certificates) it was a bit difficult to find utilities in numerical order. So, I helped Nigel develop a table of one-line summaries to give users a brief list of all the programs, with iconic descriptors of the function and the output types generated by the program (HTML, text, or JCL). This is shown in Figure 1.

The table also includes page numbers and bookmark jumps to the specific location in the manual, which most people find

Utility	PAGE	Description	H T M L	J C L	T E X T	U S E R	G R P	D S N	G E N R	C E R T	M I S C
<a href="#">RACF00</a>	<a href="#">12</a>	IRRDPU00 Flatfile Pre-Processor (Removes Binary Chars)									<input checked="" type="checkbox"/>
<a href="#">RACF01</a>	<a href="#">12</a>	Summary of Unloaded Records (Totals By Record Type)	<input checked="" type="checkbox"/>								<input checked="" type="checkbox"/>
<a href="#">RACF02</a>	<a href="#">12</a>	Profiles Owned By Non-Existent IDs or Groups	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>							<input checked="" type="checkbox"/>
<a href="#">RACF03</a>	<a href="#">19</a>	Group Tree Text Report	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>				
<a href="#">RACF04</a>	<a href="#">19</a>	List Groups With Installation Data And Create Date	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>				
<a href="#">RACF05</a>	<a href="#">27</a>	List Expired User IDs - Generate JCL To Delete	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>					
<a href="#">RACF06</a>	<a href="#">19</a>	List Group = ??? (With JCL To Remove Users)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>				
<a href="#">RACF07</a>	<a href="#">35</a>	Recursive HLQ Access Report With DSN Text File	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>			
<a href="#">RACF08</a>	<a href="#">27</a>	Search For User ID Using Specified String			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
<a href="#">RACF09</a>	<a href="#">27</a>	List All User IDs Starting With...	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>					
<a href="#">RACF10</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF11</a>	<a href="#">13</a>	Enhanced User ID Cross-Reference With JCL To Recreate	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>
<a href="#">RACF12</a>	<a href="#">40</a>	Access With User ID & Name for General Resource Profile	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF13</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF14</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF15</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF16</a>	<a href="#">40</a>	List Profiles In Member/Group Class Pair	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF17</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF18</a>	<a href="#">28</a>	Generate Data File For Fast Searches Of User IDs			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					
<a href="#">RACF19</a>	<a href="#">40</a>	List General Resource Profiles With Access Lists	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF20</a>	<a href="#">41</a>	List General Resource Profiles With Access Lists/Prefix (JCL)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>		
<a href="#">RACF21</a>	<a href="#">13</a>	Enhanced User ID Cross-Reference With JCL To Remove	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>
<a href="#">RACF22</a>	<a href="#">41</a>	List Class Profiles With Installation Data	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF23</a>	<a href="#">35</a>	List Dataset Profiles With Prefix = ???	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>			
<a href="#">RACF24</a>	<a href="#">28</a>	List Revoked User IDs	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>					
<a href="#">RACF25</a>	<a href="#">41</a>	List General Resource Profiles With Access Lists/Prefix (JCL to Delete)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>		
<a href="#">RACF26</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF27</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF28</a>	<a href="#">42</a>	List CICS Profiles With Transaction Prefix	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF29</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF30</a>	<a href="#">42</a>	List All STARTED Profiles With STDATA	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF31</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF32</a>	<a href="#">14</a>	List All Profiles With WARNING Attribute	<input checked="" type="checkbox"/>								<input checked="" type="checkbox"/>
<a href="#">RACF33</a>	<a href="#">42</a>	List All CICS Profiles For Default CICS Class TCICSTRN	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF34</a>	<a href="#">43</a>	List Duplicate Transaction Entries for Class TCICSTRN	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF35</a>	<a href="#">43</a>	Compare General Resource Profiles With Specified Prefix	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>		
<a href="#">RACF36</a>	<a href="#">20</a>	Compare Two Groups And List User IDs Found In Both	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>				
<a href="#">RACF37</a>	<a href="#">43</a>	List General Resource Members With Specified ID/Group in Access List	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>		
<a href="#">RACF38</a>	<a href="#">28</a>	Generate Audit Report On Higher User IDs	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>					
<a href="#">RACF39</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF40</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF41</a>		~~~~~EMPTY~~~~~									
<a href="#">RACF42</a>	<a href="#">14</a>	List All Discrete Profiles with ALTER	<input checked="" type="checkbox"/>								<input checked="" type="checkbox"/>
<a href="#">RACF43</a>		~~~~~EMPTY~~~~~									

Figure 1: One-line summary of Pentland Utilities (continues)



Utility	PAG	Description	H	T	U	G	D	G	C	M
			T	J	E	S	R	S	N	R
			M	C	X	R	S			S
RACF44		~~~~~EMPTY~~~~~								
RACF45		~~~~~EMPTY~~~~~								
RACF46	29	Generate JCL To Remove List Of User IDs			<input checked="" type="checkbox"/>					
RACF47	20	List Group With JCL To Change Owner/DFLTGRP To New Group				<input checked="" type="checkbox"/>				
RACF48	44	List General Resource Profiles With Access Lists/Non-Prefix (JCL)						<input checked="" type="checkbox"/>		
RACF49	44	List General Resource Profiles With Access Lists/Non-Prefix (DEL)						<input checked="" type="checkbox"/>		
RACF50	20	List User IDs In Group With DFLTGRP & All Connected Groups				<input checked="" type="checkbox"/>				
RACF51	45	List Prefixed General Resource Profile Pairs In Friendly Format						<input checked="" type="checkbox"/>		
RACF52	14	List All Profiles/Access Lists For OwnerID								<input checked="" type="checkbox"/>
RACF53	21	List Group With CICS Details				<input checked="" type="checkbox"/>				
RACF54		~~~~~EMPTY~~~~~								
RACF55		~~~~~EMPTY~~~~~								
RACF56	35	List Datasets/JESSPOOLS with UACC > NONE						<input checked="" type="checkbox"/>		
RACF57		~~~~~EMPTY~~~~~								
RACF58	15	List All Profiles With NOTIFY Set (With JCL)								<input checked="" type="checkbox"/>
RACF59	15	List Profiles With Non-Default Audit Attribute								<input checked="" type="checkbox"/>
RACF60		~~~~~EMPTY~~~~~								
RACF61	36	Grant Access To Some Dataset Profiles With Prefix & User ID						<input checked="" type="checkbox"/>		
RACF62	45	Grant Access To Some General Resource Profiles With Prefix & User ID						<input checked="" type="checkbox"/>		
RACF63		~~~~~EMPTY~~~~~								
RACF64	45	List General Resource Profiles With Access Lists/Prefix (JCL)						<input checked="" type="checkbox"/>		
RACF65	46	List General Resource Class						<input checked="" type="checkbox"/>		
RACF66	29	List User Command In Print For Specific ID			<input checked="" type="checkbox"/>					
RACF67	46	Set Or Clear Notify On All Profiles In Prefixed General Resource						<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
RACF68	36	List APF Libraries Without Fully Qualified DSN Profiles						<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
RACF69	21	List All Revoked Group Connections (With JCL)				<input checked="" type="checkbox"/>				
RACF70	22	Count The Number Of Users In A Group				<input checked="" type="checkbox"/>				
RACF71	37	List DATASET Access But Without Names - Faster Version of RACF07						<input checked="" type="checkbox"/>		
RACF72	46	Summary Breakdown of General Resource Profiles						<input checked="" type="checkbox"/>		
RACF73	22	List Group With Names And All Other Group Connections				<input checked="" type="checkbox"/>				
RACF74		~~~~~EMPTY~~~~~								
RACF75	22	List Group With Only User ID & Group On Report				<input checked="" type="checkbox"/>				
RACF76	23	List Group With User ID, Name, Connect Owner, Authority				<input checked="" type="checkbox"/>				
RACF77	23	List All Groups Where Connect Not Owned				<input checked="" type="checkbox"/>				
RACF78		~~~~~EMPTY~~~~~								
RACF79	30	List User Where OWNER Not Equal To DFLTGRP			<input checked="" type="checkbox"/>					
RACF80	37	List Discrete Dataset Profiles With JCL To Make Generic						<input checked="" type="checkbox"/>		
RACF81		~~~~~EMPTY~~~~~								
RACF82	30	Annotate List Of User IDs			<input checked="" type="checkbox"/>					
RACF83		~~~~~EMPTY~~~~~								
RACF84	30	List User IDs Which Are All Numeric			<input checked="" type="checkbox"/>					
RACF85	47	List General Resource Profiles With Access Lists/Prefix (JCL)						<input checked="" type="checkbox"/>		
RACF86	31	Create JCL To Delete User IDs Still Owning DSN Profiles			<input checked="" type="checkbox"/>					
RACF87	23	GID Group Listing				<input checked="" type="checkbox"/>				
RACF88	31	UID User Listing			<input checked="" type="checkbox"/>					

Figure 1: One-line summary of Pentland Utilities (continues)

Utility	PAG	Description	H T M	J C	T E X	U S E	G R S	D S N	G E N	C E R	M I S
<a href="#">RACF89</a>	<a href="#">15</a>	List All Profiles With SECURITY In Installation Data									<input checked="" type="checkbox"/>
<a href="#">RACF90</a>	<a href="#">37</a>	List All Dataset Profiles With JCL To Rebuild						<input checked="" type="checkbox"/>			
<a href="#">RACF91</a>	<a href="#">47</a>	List All General Resource Profiles With Access Lists.						<input checked="" type="checkbox"/>			
<a href="#">RACF92</a>	<a href="#">31</a>	List All User IDs With Revoked & Other Data				<input checked="" type="checkbox"/>					
<a href="#">RACF93</a>	<a href="#">32</a>	List CICS Users Showing Any OPIDs				<input checked="" type="checkbox"/>					
<a href="#">RACF94</a>	<a href="#">50</a>	List Digital Certificates By User ID & Label								<input checked="" type="checkbox"/>	
<a href="#">RACF95</a>	<a href="#">50</a>	List All Digital Certificates (Unsorted)								<input checked="" type="checkbox"/>	
<a href="#">RACF96</a>	<a href="#">50</a>	List All Digital Certificate Key Rings								<input checked="" type="checkbox"/>	
<a href="#">RACF97</a>	<a href="#">50</a>	List All Digital Certificate Mappings								<input checked="" type="checkbox"/>	
<a href="#">RACF98</a>	<a href="#">51</a>	List All Digital Certificate Trusts								<input checked="" type="checkbox"/>	
<a href="#">RACF99</a>	<a href="#">51</a>	List All Digital Certificates Sorted By Expire Date								<input checked="" type="checkbox"/>	
<a href="#">RACF100</a>	<a href="#">51</a>	List All Digital Certificates Sorted With Cut-off Date								<input checked="" type="checkbox"/>	
<a href="#">RACF101</a>	<a href="#">52</a>	List All Digital Certificates Sorted By Expire Month								<input checked="" type="checkbox"/>	
<a href="#">RACF102</a>	<a href="#">32</a>	Annotate List Of User IDs				<input checked="" type="checkbox"/>					
<a href="#">RACF103</a>	<a href="#">32</a>	List User IDs With Duplicate Name Data				<input checked="" type="checkbox"/>					
<a href="#">RACF104</a>	<a href="#">24</a>	Group Tree Text Output for Visio Input					<input checked="" type="checkbox"/>				
<a href="#">RACF105</a>	<a href="#">33</a>	List STARTED User IDs & Check PROTECTED				<input checked="" type="checkbox"/>					
<a href="#">RACF106</a>	<a href="#">33</a>	List UID(0) User IDs & Check PROTECTED				<input checked="" type="checkbox"/>					
<a href="#">RACF107</a>	<a href="#">38</a>	Find Catalogues Without Dataset Profiles						<input checked="" type="checkbox"/>			
<a href="#">RACF108</a>	<a href="#">53</a>	List of Digital Certificates By Owner User With JCL								<input checked="" type="checkbox"/>	
<a href="#">RACF109</a>	<a href="#">53</a>	Digital Certificate Search (Up To Nine Search Strings)								<input checked="" type="checkbox"/>	
<a href="#">RACF110</a>	<a href="#">34</a>	List All User IDs With TSO Segment by TSOPROC				<input checked="" type="checkbox"/>					
<a href="#">RACF111</a>	<a href="#">24</a>	List Group for Sarbanes-Oxley Auditors					<input checked="" type="checkbox"/>				
<a href="#">RACF112</a>	<a href="#">54</a>	List Digital Certificate Sorted By Certificate Auth								<input checked="" type="checkbox"/>	
<a href="#">RACF113</a>	<a href="#">24</a>	Annotate A List Of Groups					<input checked="" type="checkbox"/>				
<a href="#">RACF114</a>	<a href="#">47</a>	Reports On CDT Class						<input checked="" type="checkbox"/>			
<a href="#">RACF115</a>	<a href="#">38</a>	List Dataset Profiles For Supplied Text File						<input checked="" type="checkbox"/>			
<a href="#">RACF116</a>	<a href="#">48</a>	List General Resource Profiles For Supplied Text File						<input checked="" type="checkbox"/>			
<a href="#">RACF117</a>	<a href="#">25</a>	Produce Summary Statistics For Groups					<input checked="" type="checkbox"/>				
<a href="#">RACF118</a>	<a href="#">16</a>	List Discrete Profile With Generic Characters									<input checked="" type="checkbox"/>
<a href="#">RACF119</a>	<a href="#">55</a>	Generate Digital Certificate JCL using RADCERT								<input checked="" type="checkbox"/>	
<a href="#">RACF120</a>	<a href="#">56</a>	Report Digital Certificate Common Names From RACF119								<input checked="" type="checkbox"/>	
<a href="#">RACF121</a>	<a href="#">25</a>	List Groups Showing Users That Never Logged On					<input checked="" type="checkbox"/>				
<a href="#">RACF122</a>	<a href="#">34</a>	List Users Never Signed On & Not PROTECTED				<input checked="" type="checkbox"/>					
<a href="#">RACF123</a>	<a href="#">48</a>	List General Resource Profiles/Members For Supplied Text File						<input checked="" type="checkbox"/>			
<a href="#">RACF124</a>	<a href="#">16</a>	Generate Text Files of User IDs, Groups, DSN Profiles, GenRes Profiles				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
<a href="#">RACF125</a>	<a href="#">48</a>	Generate Text File Listing of Profiles in Class Pair With UACC Not NONE						<input checked="" type="checkbox"/>			
<a href="#">RACF126</a>	<a href="#">25</a>	List Details of Groups Referenced As DFLTGRPs in Base User Segments					<input checked="" type="checkbox"/>				
<a href="#">RACFAWK</a>	<a href="#">17</a>	General Purpose Extract Tool									<input checked="" type="checkbox"/>
<a href="#">RACFDIAG</a>	<a href="#">17</a>	IRRDBU00 Flatfile Diagnostic (Check For File Errors)									<input checked="" type="checkbox"/>
<a href="#">RACFIJCL</a>	<a href="#">17</a>	Build JCL Around RACF Commands (Similar To CLIST Option)									<input checked="" type="checkbox"/>

Figure 1: One-line summary of Pentland Utilities (concluded)

helpful. One fly in the ointment there, however: for some reason the original Word document will jump directly to the specific utility description (even at the bottom of the page). When the Word document converts from .doc to .pdf, though, the Acrobat document only jumps to the page instead of the bookmark!

In the manual, Nigel was very gracious and gave acknowledgements to several people who had helped over the years in the development of the programs (including ideas and suggestions) – Mike Cairns, Jeffery Loewenstein, Mark Wilson, Ulrich Boche, and your humble correspondent.

Nigel also created a ‘catch-all’ batch process, which would allow users to run a specific set of utilities, and then create an HTML index to allow you to ‘jump’ to the report you wanted. This is especially handy when you want to get a specific set of reports over to the auditors in such a way as to satisfy their system review requirements and get them out of your hair.

## RUNNING THE UTILITIES

To go over a bit of old information, running the Pentland Utilities takes a little preparatory work. First, you need to download a few things from Nigel’s site (<http://www.racf.co.uk>):

- Pentland Utilities executables – <http://www.racf.co.uk/racf.zip>
- Pentland Utilities ini file – <http://www.racf.co.uk/racf.ini>
- Pentland Utilities manual – <http://www.racf.co.uk/racf.pdf>.

Create a folder on your C:\ drive with the name RACF, and unzip the executables to there. Then, create a subfolder (or subfolders) with the prefix-name of the LPAR you are going to examine. For example, we’ll use SMFX as a name in our discussion here. Copy the racf.ini file into C:\RACF\SMFX. Beneath that folder, create three subfolders – HTM, JCL, and TXT. They’ll be your repository for the reports the software will generate. And don’t forget to edit the racf.ini file with the dates and file names you’ll

be needing for the execution of the programs. The manual gives you all the gory details for that.

For the reports to generate, you'll need to provide a couple of things from the mainframe:

- Flat file of the RACF database – IRRDBU00
- Flat file of the DSMON report – ICHDSM00
- Flat file of all dataset names – IKJEFT01.

That last one is necessary only if you want to look for any datasets that are not protected by RACF profiles (RACF107). This is for the truly adventurous, but more about that later.

Download the output from your mainframe to your PC (in text format, not binary!) and pop them into the C:\RACF\SMFX folder.

Now, to actually run the programs, you need to do a few more steps. First, open a cmd.exe window and make a few changes to the settings so you can scroll back further and see all of the messages, ie:

- Right-click on the top bar of the command prompt screen and select *Properties*.
- Select the *Options* tab:
  - increase the *Buffer Size* to 100
  - increase the *Number of Buffers* to 20.
- Select the *Layout* tab:
  - increase *Screen Buffer Size*:
    - o increase the *Width* to 100
    - o increase the *Height* to 5000.
  - increase *Window Size*:
    - o increase the *Width* to 100
    - o increase the *Height* to 50.

This also comes in handy if you want to do a cut-and-paste of all the run-time messages as an audit trail. I've certainly found that to be of value, especially if you're doing a large number of utility commands or running the processes through a batch file (such as my MEGA-BAT.bat process – see next issue).

When you have that done, you will need to do a bit of Windows %path% file maintenance, ie:


```
cd \  
PATH %PATH%; c: /RACF; c: /RACF/SMFX; c: /RACF/SMFX/HTM; c: /RACF/SMFX/JCL; c: /  
RACF/SMFX/TXT  
cd RACF  
cd SMFX
```






Just remember that this page is giving you what appears to be multiple lines for the PATH command. That is because of word-wrap. For your purposes, that PATH command is a single line of text in the cmd.exe window. I've included the other subdirectories because you may wish to use some of Nigel's other utilities from his DOSUTILS collection after the RACF utilities have been completed (you don't need that many in the batch process, though). Once you are in the C:\RACF\SMFX folder, you are ready to execute. I'd advise that you always run two of the utilities for the first time you're going to process the RACF flat file – RACFDIAG and RACF00 (in that order). RACFDIAG checks the flat file to ensure that there are no structural problems that might prevent the other programs from running correctly. It is, admittedly, rare, but it can happen – especially if you're dealing with RACF databases in countries using code sets other than the standard English or American ones. I ran into this when doing a RACF analysis in Tokyo last year, so it does bear checking out. RACF00 fixes a little bit of binary hanky-panky that sometimes creeps into any downloaded RACF flat file.

## ARE YOU CERTAIN?

One of the major developments in RACF over the past few years has been its handling of digital certificates. Certificates, as we are all aware, are an excellent security tool when handled

properly, and an operational nightmare when the contrary is true. Maintaining the certificates, the key rings, the mapping, the expiry dates, the trust relationships, etc, can be a major migraine. Sadly (but not unexpectedly), the IBM screens and reporting tool for RACF digital certificates are not as good as they could be.

Enter Pentland Utilities with the Digital Certificate Collection (not available in any store!). Thirteen rather nifty little programs that can take care of your digital certificate reporting and JCL generation needs quite nicely. The little  icon denotes which of the utilities below actually generate JCL.

- RACF94 – list digital certificates by user ID and label
- RACF95 – list all digital certificates (unsorted)
- RACF96 – list all digital certificates key rings
- RACF97 – list all digital certificates mappings
- RACF98 – list all digital certificates trusts
- RACF99 – list all digital certificates sorted by expiry date
- RACF100 – list all digital certificates sorted by cut-off date
- RACF101 – list all digital certificates sorted by expiry month
- RACF108 – list of digital certificates by owner user
- RACF109 – digital certificates search (up to nine search strings)
- RACF112 – list digital certificate sorted by certificate authority
- RACF119 – generate digital certificate JCL using RADCERT
- RACF120 – report digital certificate common names from RACF119.

I am particularly impressed by RACF101. This creates 13 HTML output files. One is the 'index' page, which then provides links to each expiry month – very slick. RACF109 is also an excellent search tool if you have a lot of digital certificates to deal with or to sort through in order to create or recreate certificates via JCL. Since the native RACF command **SR** doesn't search for digital certificates, this is a very handy tool. Just remember that if you don't enter a search argument, RACF109 can run for quite a while. RACF119 is a good tool for generating JCL to create **RADCERT** commands, which can then be uploaded and run on the mainframe to provide a separate report, which can be used as input to RACF120. RACF120 is also excellent, but it takes a bit of additional handling, ie:





- Run RACF119 to generate JCL (SMFX-RACF119.jcl).
- Transfer the SMFX-RACF119.jcl file to mainframe.
- Run that JCL on the mainframe.
- Download the output from that job back to the PC.
- Run RACF120 against that output file.

The nice part of this is that if you use an old output file from the results of RACF119 against RACF120, it'll give you a warning that the list doesn't match the certificate list in the IRRDBU00 flat file.

## OTHER NEW THINGS

When I wrote the review back in 2003, the last name on the utility list was RACF92. Now, it's RACF126 – a total of 34 utilities. Subtracting the 13 utilities described in the previous section, that leaves 21 additional programs:

- RACF93 – list CICS users showing any OPIDs
- RACF102 – annotate list of user IDs
- RACF103 – list user IDs with duplicate name data

- RACF104 – group tree text output for Visio input
- RACF105 – list STARTED user IDs and check PROTECTED
- RACF106 – list UID(0) user IDs and check PROTECTED
- RACF107  – find catalogs without dataset profiles
- RACF110  – list all user IDs with TSO segment by TSOPROC
- RACF111 – list group for Sarbanes-Oxley auditors
- RACF113 – annotate a list of groups
- RACF114 – reports on CDT class
- RACF115 – list dataset profiles for supplied text file
- RACF116 – list general resource profiles for supplied text file
- RACF117 – produce summary statistics for groups
- RACF118  – list discrete profile with generic characters
- RACF121 – list groups showing users that never logged on
- RACF122 – list users never signed on and not PROTECTED
- RACF123  – list general resource profiles/members for supplied text file
- RACF124 – generate text files of user IDs, groups, DSN profiles, GenRes profiles
- RACF125 – generate text file listing of profiles in class pair with UACC not NONE
- RACF126 – list details of groups referenced as DFLTGRPs in base user segments.

Because of space considerations, we can't go through each and every item. So, I'll give you the highlights.

If (like me) you have an issue with multiple user IDs for a single



user you'll enjoy RACF103. It checks the user name field for duplicates. Granted, even a single letter difference will mean you might miss a few duplicate IDs, but this should catch plenty of them.

RACF104 is a personal favourite of mine, partly because I suggested it, and partly because it makes an excellent tool for mapping group structures in a more graphic fashion. Literally. It provides input to Visio's Organizational Chart 'wizard', so that you can make a full (and rather large) chart of your group tree. If your tree is more like a ground shrub (low and wide) that's a disadvantage for accurate and adequate printing unless:

- 1 You have access to a plotter with a very long roll of paper
- 2 You've got lots and lots of wall space.

However, Visio's wizard is designed to allow you to control what level (and even what group) you wish to use as your highest level on the page, and how many levels down you wish to go. Granted, it's not perfect, but it's a lot easier to follow than the linear listing in the DSMON report.

RACF107 was mentioned earlier. When you use the JCL mentioned in the manual, you can create a full listing of all datasets in the LPAR. RACF107 then compares the dataset list with the dataset profiles in the flat file. The result is a listing of all datasets left unprotected by any RACF profile. This comes in handy when you've got PROTECTALL turned off.

Have you got Sarbanes-Oxley auditors breathing down your neck, asking you for the same stuff four or five times? – stuff you don't understand the necessity for. Don't worry. They don't understand it either. However, they're the SOX auditors, so they get what they want. And what they want, apparently, is RACF111. It lists user IDs, names, last access date, creation date, and whether the revoked attribute is on.


Finally, a number of the utilities listed here need input files beyond the IRRDBU00 flat file or the DSMON report – user ID lists, general resource class combinations, group lists, etc. The

problem is, extracting those lists can be a bit of a pain. For that, let me introduce you to RACF124 – the handy list generator. It outputs simple text files, which can be used as input for a wide variety of utilities.


## FIXES AND UPDATES FOR THE GOLDEN OLDIES

‘New and improved’ – usually when you see that on a packet of corn flakes or a box of washing powder, you know that the term refers not to the contents but to the packaging. Normally, it’s the same old veQ inside (sorry, but since we’re IT professionals, at least one Klingon term must by law be included in this article). The nice thing about Pentland Utilities is that when something gets improved, it is an actual improvement! It becomes easier to read or more functional or corrects an odd error.

A few examples:

- RACF00 – enhanced to allow for European characters with accents. This also resulted in minor updates to the certificate reports, and refined the filtering in relation to dollar and pound signs. The main place where this was manifesting itself was in the digital certificate reports (RACF94 through RACF101).
- RACF01 – updated to include new record types added in z/OS V1.6.
- RACF07 – addition of output file containing list of datasets as plain text.
- R A C F 1 1  – enhanced to include command line options to allow better control on the granularity of reporting. Allows reporting on the ACL entries for just a user ID, the user ID plus selected connected groups, or the user ID and all connected groups. It also reports on attributes (both user and connection), class authorizations, whether a password is set to non-expiring, and whether UID(0) is defined for user. Furthermore, if a report is being generated on a user, it prompts for whether the report should include enumeration

of all connected groups. This runs very slowly if you've got a highly over-connected group in the mix.

- RACF38 – revamped to include new options in racf.ini to enable the suppression of sections. There is now a variable to define a threshold at which reporting occurs on password expiry of users. Also updated to include a new section listing users with the UAUDIT attribute.
- RACF52 – minor enhancement to display certificate names properly with spaces rather than dollar signs. Has also been improved in relation to handling, ie displaying, digital certificate information. This can be used to report on certificates owned by a user ID (note: all irrcerta certificates are actually owned by IBMUSER).
- RACF58  – fixed error in syntax of JCL relating to generic general resource class profiles.
- RACF82 – enhanced to show in bold any user ID that could not be annotated if not found in the flat file. Previously it was just dropped with no indication.
- RACF87 – enhanced to include hex values.
- RACF88 – enhanced to include hex values. Another column was added to show default group of each user listed. There are further extensions in functionality to accept a run-time command line argument of a group. If a group is specified it will list all members of that group whether or not they have an OMVS segment.
- RACF92 – added another column (by request) with the last password date showing the date the password was last changed.
- RACFDIAG – significant development to generate far more diagnostic information, especially relating to unexpected characters found in the unload.

## MEGA-BAT (THANKFULLY, NOT ANOTHER BATMAN SEQUEL...)

As the renowned existentialist philosopher Steve Martin once said (with a banjo in his hand and an arrow through his head), “I’m a ramblin’ guy”. When you’re a consultant, you bounce around from town to town (or continent to continent) armed only with your experience, your wealth of RACF knowledge, and enough prescription sleeping pills to get you through the long flights so as to escape the boredom, the noise, the airline food, and the inevitable torture of in-flight Paulie Shore movies. [*Editor’s note: Paulie Shore is an American actor and comedian with a very mixed reputation.*]

However, you also take a toolkit with you whenever possible. Not, thanks to the US Transportation Security Agency, one that consists of dangerous implements like screwdrivers, hammers, wrenches, tweezers, or nail clippers. Some clients don’t have products like Vanguard, some don’t have Consul (and some barely have RACF, quite frankly), and while ICETOOLS is a good source of SMF reporting (when CARLa’s not available, anyway), a pocket full of Pentland Utilities is a great and comforting resource. Even when the client has both Vanguard and Consul, I still like to use the Pentland Utilities. While both products are excellent, the only drawback is that it is a bit cumbersome to get the reports from the mainframe into a format that looks good in a report. Since most managers are PC-literate (while being mainly mainframe a-cephalic), it is nice to have a way to provide data in an HTML format that can be imported/converted into Word documents.

Building the commands every time for every different client gets a bit bothersome after a while. So, I decided to expand on Nigel’s idea of a batch file that creates a set of auditor reports. Granted, mine doesn’t come with an HTML front-end, but this batch file allows me to take the standard dataset prefixes, the SYS1 group, the IBMUSER, and a bunch of standard general resources classes, and build a rather massive ‘starting point’ from which to build on.

Once I’ve identified the tailored member-class/group-class combinations for their various CICS regions, I can expand the

sections for RACF33 and RACF34. Any additional GenRes classes can be plugged into programs like RACF12, RACF16, RACF22, and RACF65.

The same goes for the 'critical' dataset HLQs for each individual company, or the more dangerous user IDs, and the Pentland Utilities that cover them. Furthermore, once you get past the RACF11 jobs, no further operator intervention is required, and you can let your PC or laptop chug away merrily at the processing while you are doing something else.

Even if you're not a world-traveller, but a stay-at-home security bod or internal auditor, this batch file can provide you with a wealth of information about your system. Furthermore, it can be run at regular intervals (once you've tailored it to your needs) so that you can have a standard set of reports from which to make comparisons over time. The JCL is especially valuable if you want to have a 'drop-dead' point from which you can recover things like user IDs or dataset profiles if your RACF database (and all of your back-ups) get corrupted. Granted, that's a very very slim chance, but if you're a belt-and-braces kind of security/audit person, this fits quite nicely into your mindset.

*Editor's note: The code was too long to fit into this issue and will be published in the next issue of RACF Update.*

## CONCLUSION

If you've not yet used these utilities, you're missing out on a great reporting and recovery opportunity.

Many thanks to Nigel Pentland for taking an already good thing and making it even better!

*Editor's note: Doc Farmer describes himself as a security consultant, humourist, genius, and part-time curmudgeon living in America's heartland. He can be contacted at DocFarmer9999@yahoo.co.uk.*

---

*Doc Farmer*  
*Independent Security Consultant (USA)*

© Xephon 2007

---

## August 2004–May 2007 index

Items below are references to articles that have appeared in *RACF Update* since issue 37, August 2004. References show the issue number followed by the page number(s). Subscribers can download copies of all issues in Acrobat PDF format from Xephon's Web site.

ACCESS	44.27-47	ICSF	41.10-45, 43.3-4
Access checking	37.51-58	Identity mapping	45.3-7
Adding users	48.7-14	IFTHEN	45.18-24
Add-on products	40.51-56	IND\$FILE	37.34-50
ADDUSER	42.16-40	IRRDBU00	41.50-70, 43.16-26, 43.26-48, 44.12-20, 44.47-60, 44.60-63, 42.16-40
ALTUSER	42.16-40	IRRIRA00	45.3-7
Assigning Unix GIDs	48.46-52	ISPF/PDF	37.3-9
Attack	38.8-36	LDAP	40.37-50, 41.3-4, 43.6-7
Audit	43.48-59, 48.26-46	Mandatory Access Control (MAC)	44.9-12
Batch mode	40.3-6	Message Authentication Code (MAC)	41.11
C/C++	38.41-63	Mixed case passwords	48.15-26
Certificate checking	46.16-19	Monitoring	45.39-58
CICS	40.7-19, 47.7	Multilevel security (MLS)	44.7-12
Compare databases	46.9-15	Network Authentication Service	43.7
Cory Curtis	47.12-21, 48.26-46	OCEP	43.7
Courses	37.58-59, 39.12-22	OCSF	43.4
Cryptography	41.71	OpenSSH	43.8
Database	39.22-32	OPERATIONS	46.56-67
Database unload utility	41.50-70, 43.16-26, 43.26-48, 44.12-20, 44.47-60, 44.60-63	Orphans	42.45-59
DB2	39.36-67, 43.26-48, 44.47-60	Passphrase	48.3-7
DBSYNC	46.9-15	Password	38.8-36
DCE	43.7	Password copy	44.12-23
Deletion	38.3-7, 33.11-32, 36.34-53	PCI	41.10-12
DFSORT	45.18-24	PDAS	43.8
Discretionary Access Control (DAC)	44.8-12	Pentland	47.12-21, 48.26-46, 48.52-66
EIM	43.8	PERMIT	44.27-47
File access restrictions	46.19-38	Permits	42.9-15
Firewall	43.6	PKI	43.5-6
Free space	39.22-32	Profiles	42.45-59
Global Access Checking (GAC) Table	38.37-40	Questions & Answers	42.41-44, 43.9-15, 45.58-63, 46.3-9, 47.3-7
Goodies	47.12-21, 48.26-46	Quiz	41.47-49
Group	42.9-15	RACF Remote Sharing Facility (RRSF)	45.11-18
ICHRIX	37.14-33		

Redundant groups	41.5-9	SSL	43.4-5
Revoked users	37.14-33, 40.20-36	Surrogat class	45.7-11
RMF	41.12-14	Tape Security Options	47.8-11
Scrollable output	44.24-27	TASA	45.33-39
SDBM	41.3-4	Terminology	38.63-67, 39.7-12
SEARCH	44.3-7	UADS	39.32-35
Security	40.57-58, 42.3-9	Unload database	43.59-63
Security labels	45.24-33	User removal	47.41-59
SETROPTS	40.20	USS	46.19-38
SMF	41.14	Utilities	46.39-56, 47.12-21, 47.22-40, 48.52-66
SMF records	39.3-7	WebSphere MQ	37.9-13
SPLICE	43.16-26		

IBM has announced Version 1.8 Consul zSecure Suite, which is part of the newly acquired Consul security portfolio.

Consul zSecure Suite V1.8 gives z/OS customers an easier way to generate and review XML security audit reports on their mainframe data. It is also meant to ensure a more secure operating environment for users migrating to z/OS 1.8.

The Consul software is currently being integrated into the Tivoli Software portfolio.

For further information contact:

URL: <http://www.consul.com/PressReleaseDetail.asp?prid=123&pid=23&PageLoc=Home>.

\* \* \*

DataDirect Technologies, part of Progress Software, has announced new capabilities for its DataDirect Shadow RTE mainframe integration suite for optimizing Web services security, testing, and development across a broad range of mainframe databases, business logic, and screen-based applications. With the increase in SOA applications, the security protocols protecting mainframe assets need to be updated enabling them to work in the stateless environment of loosely-coupled Web services.

Originally designed to support individual sign-on, RACF is now required to process thousands of Web services logins per hour, creating an authentication bottleneck that slows performance and consumes expensive mainframe CPU cycles needed to support the security manager.

The DataDirect Shadow product addresses this problem with Security Optimization and

Management (SOM), which works in conjunction with the established client and host security protocols to eliminate redundant authentication requests by caching security credentials. Additionally, the Shadow SOM feature maintains the integrity of the security infrastructure by subscribing to updates in the security manager database. The Shadow SOM feature supports RACF.

For further information contact:

URL: [www.datadirect.com/company/news/press/pressitem/pressrelease\\_939821/index.ssp](http://www.datadirect.com/company/news/press/pressitem/pressrelease_939821/index.ssp).

\* \* \*

Sun Microsystems has announced Version 7.0 of the Sun Java System Identity Manager, its ID management solution, which unifies its user provisioning and auditing products. It allows a large number of ID management tasks to be performed from a single console, including role delegation, password synchronization, automated provisioning, and compliance auditing.

Policy and audit are strong points for Identity Manager. By integrating fully-functional auditing capabilities into the standard interface, it allows users to provision a new user for Active Directory, RACF, and Oracle, and compare the access given to current policies. If there are any violations, provisioning is automatically escalated for approval based on a process defined by the user. Existing identities can be periodically audited for policy violations.

For further information contact:

URL: [www.sun.com/software/products/identity\\_mgr/index.jsp](http://www.sun.com/software/products/identity_mgr/index.jsp).

\* \* \*

