40

RACF update

May 2005

In this issue

A xephon PUBLICATION

RACF Update

Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

Editor

Trevor Eddolls
E-mail: trevore@xephon.com

Publisher

Colin Smith
E-mail: info@xephon.com

RACF Update on-line

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

Subscriptions and back-issues

A year's subscription to *RACF Update* (four quarterly issues) costs $290.00 in the USA and Canada; £190.00 in the UK; £196.00 in Europe; £202.00 in Australasia and Japan; and £200.50 elsewhere. The price includes postage. Individual issues, starting with the August 2000 issue, are available separately to subscribers for $72.75 (£48.50) each including postage.

Disclaimer

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

Contributions

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

Printed in England.

RACF 101 – using RACF in batch mode

RACF 101 is a regular column for newcomers to the RACF world. It presents basic RACF topics in a tutorial format. In this issue we will discuss how to run RACF commands in batch mode and see some of the advantages of using this method.

It is fine to enter RACF commands in ISPF, using option 6. Most of the time, this is all you want. There are times, however, when you want to do more than simply view the output on your screen.

When you run RACF commands in batch mode, you have the ability to save the results of your commands in a dataset for future reference. This is not only desirable in some cases, but is quite often required, even mandated, because of the nature of your commands.

So, when would you want to use batch mode, and how do you do it? Here are a few reasons. Each example shows a slightly different version of the skeleton JCL.

In the discussion, some knowledge of ISPF and TSO datasets is assumed.

WHEN YOU WANT TO CAPTURE RESULTS

You may want to capture the results of your RACF command in a dataset for review, reporting, or audit purposes. The ISPF RACF commands do not allow this. You can, of course, use cut-and-paste to save the results in a dataset: but this can not be only cumbersome, it can also be time consuming when your RACF command generates a lot of output.

Let's say we want to list all users in the RACF database. The RACF command for this is LISTUSER *, and the sample JCL to use is as follows:

```
//RACFJOB1  JOB 1,'YOUR NAME',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
//STEPØ1 EXEC PGM=IKJEFTØ1,REGION=1M
```

```
//SYSTSPRT DD DSN=YOUR.DATASET.NAME,DISP=SHR
//SYSTSIN  DD *
LISTUSER *
/*
```

The first statement is installation-specific, so you may want to contact the systems programmer at your installation for its specification.

The second statement is the one that runs the RACF commands in batch mode. It can be used as specified here; no changes are required.

The output will go to the dataset YOUR.DATASET.NAME. You can change this to whatever you like.

The RACF command that follows the statement //SYSTSIN is the one that will be run in batch mode. You can change it to any other RACF command.

YOU WANT TO PERFORM AN ACTIVITY REPEATEDLY

Say you want to list the RACF global options periodically and keep your results in a PDS. Using ISPF, you can view the global options on your screen, but you will not be able to save the results for later review. Batch mode provides the answer.

In this example, the output will go to the PDS YOUR.DATASET.NAME, in member name LIST1. All you need to do this is change the member name each time you submit the JCL, for example, LIST2, LIST3, etc.

```
//RACFJOB1  JOB 1,'YOUR NAME',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
//STEPØ1 EXEC PGM=IKJEFTØ1,REGION=1M
//SYSTSPRT DD DSN=YOUR.DATASET.NAME(LIST1),DISP=SHR
//SYSTSIN  DD *
SETROPTS LIST
/*
```

YOU WANT TO AUTOMATE A PROCESS

Let’s say you have decided as standard procedure to list all users in the RACF database once a week. You may not

remember to do this every week. Using batch mode, it is possible to automate this task by scheduling it using your scheduler software (whatever that may be). By automating the task you have not only removed the possibility of forgetting to do this, there is one less activity for you to do – you can now concentrate on other important tasks.

The JCL remains the same as in the first example. Simply let your scheduling team know that you want to do this once a week.

A ROUTINE TASK REQUIRES MORE THAN ONE RACF COMMAND

Let's say you have a task that requires more than one RACF command, and you need to save the result whenever you perform this task. Using batch mode will remove the need to remember a series of RACF commands.

An example of this would be to build a process to always list a userid before deleting it, and to save the result for future reference. Not only do you have the details of the userid should you ever need to restore it, but you also know which userid you deleted and when. This is standard practice at most installations, and one that the auditors love!

In this case, the commands you want to enter repeatedly for all deletions of userids are:

```
LISTUSER ABC123
DELUSER ABC123
```

The JCL you need is:

```
//RACFJOB1  JOB 1,'YOUR NAME',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
//STEPØ1 EXEC PGM=IKJEFTØ1,REGION=1M
//SYSTSPRT DD DSN=YOUR.DATASET.NAME(ABC123),DISP=SHR
//SYSTSIN  DD *
LISTUSER ABC123
DELUSER ABC123
/*
```

Here we see that you can have as many RACF commands as you wish, all of them following the statement //SYSTSIN.

WHEN BATCH MODE IS THE ONLY METHOD

Lastly, there are times when you *must* use batch mode to accomplish a task because it cannot be done using ISPF commands. An example of this is when you want to run the RACF Data Security Monitor (DSMON) report.

The JCL for this is as follows:

```
//RACFJOB1  JOB 1,'YOUR NAME',MSGCLASS=X,CLASS=A,NOTIFY=&SYSUID
//STEP1         EXEC  PGM=ICHDSM00
//SYSPRINT  DD     SYSOUT=A
//SYSUT2      DD     SYSOUT=A
//SYSIN        DD     *
    FUNCTION ALL
/*
```

IN CONCLUSION

The best way to learn batch mode is, of course, to practice. Find some uses at your installation, even though they may not be the best ones. The idea is to learn, and be productive at the same time. You can do this with RACF commands that don't make changes to the database, such as listing all RACF userids. Or you can run reports such as the Data Security Monitor.

Dinesh Dattani is an Independent Consultant specializing in mainframe security. He welcomes comments and feedback on this column. He can be contacted at dinesh123@rogers.com.

Dinesh Dattani
Independent Consultant
Toronto (Canada)

CICS command segregation in RACF

INTRODUCTION

This article will define the provisions for increased security of the CICS commands through proper segregation within the CCICSCMD/VCICSCMD General Resource profiles. Please note that a full article on CICS transaction and command segregation can be found in the March and April 2005 issues of *CICS Update*.

RELEASES/VERSIONS

For the purposes of this article, the following IBM versions and releases will be used as examples:

- z/OS Version 1.4
- RACF Version 1.4
- CICS/TS Version 2.2.

CICS COMMAND SECURITY

Everybody seems to focus on the CICS *transactions* in security set-ups, and pages can be written on the GCICSTRN/TCICSTRN General Resource profiles. Sadly, though, very little attention is paid to the ugly sister of the RACF GenRes settings, the CCICSCMD/VCICSCMD CICS command security profiles. Even IBM gives this subject short shrift in the *CICS RACF Security Guide*. I find this a bit disappointing, and somewhat disturbing, considering the immense power that some of these commands can wield.

Some organizations have virtually no segregation of the commands defined within RACF. Some segregate only the SECURITY command, normally at the request of the security

department, while lumping the rest together for virtually unlimited access by the systems programmers (and, indeed, some development programmers as well). I've always been disturbed by this, because some of the CICS commands are rather dangerous in their misapplication (either accidental or intentional).

So, since there was no real guidance on this issue, I decided to create my own command segregation. In general, it breaks down in the following manner:

- M001 – inquiry only commands
- M002 – DB2 commands
- M003 – terminal/monitor/TCPIP commands
- M004 – reserved for future use
- M005 – general use commands (segregated by RACF access level READ/UPDATE/ALTER)
- M006 – EXEC CICS-level commands
- M007 – CEMT-level commands
- M008 – high-end technical support commands only
- M009 – SECURITY command only.

As you can see, I've segregated the SECURITY command from all other functions – being a security guy, it's kind of expected. However, the other commands have been placed in more general groupings, ascending in risk from 1 (simple inquiry-only functions) up to 8 (the juicier tech-support functions) and 9 (the SECURITY function).

Access to these groups may seem somewhat random, and in some cases redundant. For example, you might want to give your technical support team access to all the functions except SECURITY, which is fine. However, there are some functions that you may wish to provide to users in some of the less technical areas. For example, there are instances where you

may want to provide development staff with access to the CEMT commands, but only on a READ basis (for CEMT dump commands, for example). Or there may be certain Web or VTAM inquiries your network staff may need (again at READ level) that you do not wish to provide to the developers.

For the permission levels you'll need to set in your PERMIT commands, use the following guide:

- Inquire = READ
- Set = UPDATE
- Create = ALTER
- Delete = ALTER.

The sample segregation gives you the opportunity to provide access to CICS commands on a needs-only basis. It may be changed or tweaked, as you feel appropriate for your own circumstances. Think of it as a starting point, one that helps you tailor your CICS command security more closely to your own requirements.

The full description of these commands can be found below, as can the JCL to create the General Resource profiles.

CICS COMMAND SEGREGATION

Lvl	*Resource name*	*Related CICS command(s)*
1	BEAN	INQUIRE BEAN
1	CFDTPOOL	INQUIRE CFDTPOOL
1	EXCI	INQUIRE EXCI
1	MVSTCB	COLLECT STATISTICS INQUIRE MVSTCB
1	RRMS	INQUIRE RRMS
1	STORAGE	INQUIRE STORAGE
1	STREAMNAME	INQUIRE STREAMNAME
1	SUBPOOL	INQUIRE SUBPOOL
1	UOWDSNFAIL	INQUIRE UOWDSNFAIL
1	UOWENQ	INQUIRE UOWENQ
2	DB2CONN	INQUIRE | SET | CREATE | DISCARD DB2CONN
2	DB2ENTRY	INQUIRE | SET | CREATE | DISCARD DB2ENTRY
2	DB2TRAN	INQUIRE | SET | CREATE | DISCARD DB2TRAN

3	CONNECTION	INQUIRE | SET | CREATE | DISCARD CONNECTION
3	IRC	INQUIRE | SET IRC
3	MONITOR	INQUIRE | SET MONITOR
3	TCPIP	INQUIRE | SET
3	TCPIPSERVICE	INQUIRE | SET | CREATE | DISCARD
3	TERMINAL	INQUIRE | SET | CREATE | DISCARD TERMINAL and INQUIRE | SET NETNAME
3	TSMODEL	CREATE | INQUIRE | SET | DISCARD
3	TSPOOL	INQUIRE
3	TSQNAME	INQUIRE | SET
3	VTAM	INQUIRE | SET VTAM
3	WEB	INQUIRE | SET
5	AUTINSTMODEL	INQUIRE | DISCARD AUTINSTMODEL
5	AUTOINSTALL	INQUIRE | SET AUTOINSTALL
5	BRFACILITY	INQUIRE | SET BRFACILITY
5	CORBASERVER	INQUIRE | SET | CREATE | DISCARD | PERFORM CORBASERVER
5	DELETSHIPPED	INQUIRE | SET | PERFORM DELETSHIPPED
5	DISPATCHER	INQUIRE | SET DISPATCHER
5	DJAR	INQUIRE | CREATE | DISCARD | PERFORM DJAR

Note: ALTER access to the associated DJAR resource is required for the PERFORM CORBASERVER SCAN command.

5	DOCTEMPLATE	INQUIRE | SET
5	DSNAME	INQUIRE | SET DSNAME
5	DUMPDS	INQUIRE | SET DUMPDS
5	ENQMODEL	INQUIRE | CREATE | SET
5	FILE	INQUIRE | SET | CREATE | DISCARD FILE
5	JOURNAL	INQUIRE | SET JOURNALNAME
5	JVMPOOL	INQUIRE | SET JVMPOOL
5	MODENAME	INQUIRE | SET MODENAME
5	PARTNER	INQUIRE | CREATE | DISCARD PARTNER
5	PROFILE	INQUIRE | CREATE | DISCARD PROFILE
5	PROGRAM	INQUIRE | SET | CREATE | DISCARD PROGRAM
5	REQUESTMODEL	INQUIRE | SET
5	SYSDUMPCODE	INQUIRE | SET SYSDUMPCODE (see note 4)
5	SYSTEM	INQUIRE | SET SYSTEM
5	TASK	INQUIRE | SET TASK and TASK LIST
5	TCLASS	INQUIRE | SET | DISCARD TCLASS and INQUIRE | SET | CREATE | DISCARD TRANCLASS
5	TDQUEUE	INQUIRE | SET | CREATE | DISCARD TDQUEUE
5	TRANDUMPCODE	INQUIRE | SET TRANDUMPCODE (see note 4)

5	TRANSACTION	INQUIRE | SET | DISCARD | CREATE TRANSACTION
5	UOW	INQUIRE | SET UOW
6	EXITPROGRAM	EXEC CICS ENABLE PROGRAM EXEC CICS DISABLE PROGRAM EXEC CICS EXTRACT EXIT EXEC CICS RESYNC ENTRYNAME INQUIRE EXITPROGRAM
6	REQID	EXEC CICS INQUIRE REQID
6	STATISTICS	INQUIRE | SET STATISTICS EXEC CICS COLLECT STATISTICS, and PERFORM STATISTICS RECORD
6	TRACEDEST	EXEC CICS INQUIRE | SET TRACEDEST
6	TRACEFLAG	EXEC CICS INQUIRE | SET TRACEFLAG
6	TRACETYPE	EXEC CICS INQUIRE | SET TRACETYPE
6	TSQUEUE	EXEC CICS INQUIRE TSQUEUE
7	DUMP	PERFORM DUMP CEMT PERFORM SNAP
7	JOURNALMODEL	EXEC CICS INQUIRE | CREATE | DISCARD JOURNALMODEL CEMT INQUIRE JMODEL
7	LINE	CEMT INQUIRE | SET LINE
7	PROCESSTYPE	CEMT DEFINE PROCESSTYPE EXEC CICS CREATE PROCESSTYPE EXEC CICS DISCARD PROCESSTYPE CEMT INQUIRE PROCESSTYPE CEMT SET PROCESSTYPE
7	UOWLINK	INQUIRE UOWLINK EXEC CICS SET UOWLINK
8	FEPIRESOURCE	Certain EXEC CICS FEPI commands (see note 3)
8	LSRPOOL	CREATE LSRPOOL
8	MAPSET	CREATE | DISCARD MAPSET
8	PARTITIONSET	CREATE | DISCARD PARTITIONSET
8	RESETTIME	PERFORM RESETTIME (see note 4)
8	SESSIONS	CREATE | DISCARD SESSIONS
8	SHUTDOWN	PERFORM SHUTDOWN (see note 2)
8	TYPETERM	CREATE | DISCARD TYPETERM
9	SECURITY	PERFORM SECURITY REBUILD

Notes:

1 If you are using prefixing, the CICS region user ID must be prefixed to the command resource name.

2 Be particularly cautious when authorizing access to these and any other CICS commands that include a SHUTDOWN option.

3 For more information about FEPI security, see the *CICS Front End Programming Interface User's Guide*.

4 See *Resource names for CEMT*.

CICS COMMAND SEGREGATION – JCL

```
ADDGROUP zzzzGRPS                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(CIX#PRD)                  -
NAME(RRRRRRRR - PRIME GRP)         -
DATA('CICS PRODUCTION REGION - RRRRRRRR - PRIME GRP-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
TOP GROUP FOR CICS REGION  - TREE STARTS HERE-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD  444-444-1414-
PRIMARY GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGRPS)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØØ                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGRPS)                 -
NAME(RRRRRRRR - IBM CMDS)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD  444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØØ)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ1                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ1)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS       INQUIRY ONLY COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD  444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ1)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ2                  -
```

```
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ2)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS                DB2 COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD 444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ2)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ3                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ3)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS TERM-MONITOR-TCPIP COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD 444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ3)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ5                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ5)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS        GENERAL USE COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD 444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ5)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ6                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ6)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS    EXEC CICS-LEVEL COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD 444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
```

```
        GROUP(zzzzGMØ6)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ7                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ7)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS         CEMT-LEVEL COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD  444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ7)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ8                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ8)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS HIGH-END TECH SUPP COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD  444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ8)            -
        AUTHORITY(USE)
ADDGROUP zzzzGMØ9                  -
OWNER(RRRRRRRR)                    -
SUPGROUP(zzzzGMØØ)                 -
NAME(RRRRRRRR - IBM GMØ9)          -
DATA('CICS PRODUCTION REGION - RRRRRRRR -  IBM CMDS-
DESCRIBE REGION HERE DESCRIBE REGION HERE DES-
IBM CICS COMMANDS  HIGH-END SECURITY COMMANDS-
OWNER=CCCCCCCCCCCCCCCCCCCCCCCCC  555-555-1515-
PRGMR=DDDDDDDDDDDDDDDDDDDDDDDDD  444-444-1414-
COMMAND GROUP   RRRRRRRR %%%%%')
CONNECT RRRRRRRR                   -
        GROUP(zzzzGMØ9)            -
        AUTHORITY(USE)
RDEFINE VCICSCMD xxxxMØØ1     -
OWNER(zzzzGMØ1)               -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 1 - INQUIRY-
 ONLY COMMANDS - TECHNICAL SUPPORT USER-
AUTH1:MMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNN 888-888-1818-
```

```
RRRRRRRR     &&&&&')                    -
AUDIT(FAILURES(READ))          -
UACC(NONE)                     -
ADDMEM(RRRRRRRR.BEAN           -
       RRRRRRRR.CFDTPOOL       -
       RRRRRRRR.EXCI           -
       RRRRRRRR.MVSTCB         -
       RRRRRRRR.RRMS           -
       RRRRRRRR.STORAGE        -
       RRRRRRRR.STREAMNAME     -
       RRRRRRRR.SUBPOOL        -
       RRRRRRRR.UOWDSNFAIL     -
       RRRRRRRR.UOWENQ)
PERMIT xxxxMØØ1                -
       CLASS(VCICSCMD)         -
       ACCESS(READ)            -
       ID(RRRRRRRR)
PERMIT xxxxMØØ1                -
       CLASS(VCICSCMD)         -
       ACCESS(READ)            -
       ID(xxxxGMØ1)
RDEFINE VCICSCMD xxxxMØØ2      -
OWNER(zzzzGMØ2)                -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 2 - DB2 COM-
MANDS - DATABASE TECHNICAL SUPPORT USER-
AUTH1:MMMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                    -
AUDIT(FAILURES(READ)SUCCESS(UPDATE))   -
UACC(NONE)                     -
ADDMEM(RRRRRRRR.DB2CONN        -
       RRRRRRRR.DB2ENTRY       -
       RRRRRRRR.DB2TRAN)
PERMIT xxxxMØØ2                -
       CLASS(VCICSCMD)         -
       ACCESS(READ)            -
       ID(RRRRRRRR)
PERMIT xxxxMØØ2                -
       CLASS(VCICSCMD)         -
       ACCESS(READ)            -
       ID(xxxxGMØ2)
RDEFINE VCICSCMD xxxxMØØ3      -
OWNER(zzzzGMØ3)                -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 3 - TERMINA-
L-MONITOR-TCPIP COMMANDS - TECH SUPPORT-
```

```
AUTH1:MMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                    -
AUDIT(FAILURES(READ)SUCCESS(UPDATE))   -
UACC(NONE)                      -
ADDMEM(RRRRRRRR.CONNECTION     -
       RRRRRRRR.IRC            -
       RRRRRRRR.MONITOR        -
       RRRRRRRR.TCPIP          -
       RRRRRRRR.TCPIPSERVICE   -
       RRRRRRRR.TERMINAL       -
       RRRRRRRR.TSMODEL        -
       RRRRRRRR.TSPOOL         -
       RRRRRRRR.TSQNAME        -
       RRRRRRRR.VTAM           -
       RRRRRRRR.WEB)
PERMIT xxxxMØØ3                 -
       CLASS(VCICSCMD)          -
       ACCESS(READ)             -
       ID(RRRRRRRR)
PERMIT xxxxMØØ3                 -
       CLASS(VCICSCMD)          -
       ACCESS(READ)             -
       ID(xxxxGMØ3)
RDEFINE VCICSCMD xxxxMØØ5      -
OWNER(zzzzGMØ5)                 -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 5 - GENERAL-
 USAGE COMMANDS -     TECHNICAL SUPPORT-
AUTH1:MMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                     -
AUDIT(FAILURES(READ)SUCCESS(UPDATE))   -
UACC(NONE)                      -
ADDMEM(RRRRRRRR.AUTINSTMODEL   -
       RRRRRRRR.AUTOINSTALL    -
       RRRRRRRR.BRFACILITY     -
       RRRRRRRR.CORBASERVER    -
       RRRRRRRR.DELETSHIPPED   -
       RRRRRRRR.DISPATCHER     -
       RRRRRRRR.DJAR           -
       RRRRRRRR.DOCTEMPLATE    -
       RRRRRRRR.DSNAME         -
       RRRRRRRR.DUMPDS         -
       RRRRRRRR.ENQMODEL       -
       RRRRRRRR.FILE           -
       RRRRRRRR.JOURNAL        -
       RRRRRRRR.JVMPOOL        -
```

```
       RRRRRRRR.MODENAME      -
       RRRRRRRR.PARTNER       -
       RRRRRRRR.PROFILE       -
       RRRRRRRR.PROGRAM       -
       RRRRRRRR.REQUESTMODEL  -
       RRRRRRRR.SYSDUMPCODE   -
       RRRRRRRR.SYSTEM        -
       RRRRRRRR.TASK          -
       RRRRRRRR.TCLASS        -
       RRRRRRRR.TDQUEUE       -
       RRRRRRRR.TRANDUMPCODE  -
       RRRRRRRR.TRANSACTION   -
       RRRRRRRR.UOW)
PERMIT xxxxMØØ5               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(RRRRRRRR)
PERMIT xxxxMØØ5               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(xxxxGMØ5)
RDEFINE VCICSCMD xxxxMØØ6     -
OWNER(zzzzGMØ6)               -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 6 - EXEC CI-
CS-LEVEL COMMANDS -   TECHNICAL SUPPORT-
AUTH1:MMMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                   -
AUDIT(FAILURES(READ)SUCCESS(READ))    -
UACC(NONE)                    -
ADDMEM(RRRRRRRR.EXITPROGRAM   -
       RRRRRRRR.REQID         -
       RRRRRRRR.STATISTICS    -
       RRRRRRRR.TRACEDEST     -
       RRRRRRRR.TRACEFLAG     -
       RRRRRRRR.TRACETYPE     -
       RRRRRRRR.TSQUEUE)
PERMIT xxxxMØØ6               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(RRRRRRRR)
PERMIT xxxxMØØ6               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(xxxxGMØ6)
RDEFINE VCICSCMD xxxxMØØ7     -
OWNER(zzzzGMØ7)               -
```

```
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 7 - CEMT CI-
CS-LEVELCOMMANDS -    TECHNICAL SUPPORT-
AUTH1:MMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                   -
AUDIT(FAILURES(READ)SUCCESS(READ))    -
UACC(NONE)                    -
ADDMEM(RRRRRRRR.DUMP          -
       RRRRRRRR.JOURNALMODEL  -
       RRRRRRRR.LINE          -
       RRRRRRRR.PROCESSTYPE   -
       RRRRRRRR.UOWLINK)
PERMIT xxxxM00Ø               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(RRRRRRRR)
PERMIT xxxxM00Ø               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(xxxxGM0Ø)
RDEFINE VCICSCMD xxxxM00Ø     -
OWNER(zzzzGM0Ø)               -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 8 - HIGH-EN-
D TECH SUPPORT ONLY COMMANDS - TECH SUP-
AUTH1:MMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                   -
AUDIT(FAILURES(READ)SUCCESS(READ))    -
UACC(NONE)                    -
ADDMEM(RRRRRRRR.FEPIRESOURCE  -
       RRRRRRRR.LSRPOOL       -
       RRRRRRRR.MAPSET        -
       RRRRRRRR.PARTITIONSET  -
       RRRRRRRR.RESETTIME     -
       RRRRRRRR.SESSIONS      -
       RRRRRRRR.SHUTDOWN      -
       RRRRRRRR.TYPETERM)
PERMIT xxxxM00Ø               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(RRRRRRRR)
PERMIT xxxxM00Ø               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(xxxxGM0Ø)
```

```
RDEFINE VCICSCMD xxxxMØØ9     -
OWNER(zzzzGMØ9)               -
DATA('FULL NAME AND DESCRIPTION OF CICS REGION-
 HERE - SHOULD BE SAME FOR ALL PROFILES-
CICS SUPPLIED COMMANDS - LVL 9 - SECURIT-
Y ONLY COMMANDS -    SECURITY DEPT ONLY-
AUTH1:MMMMMMMMMMMMMMMMMMM 999-999-1919-
 AUTH2:NNNNNNNNNNNNNNNNNNN 888-888-1818-
RRRRRRRR     &&&&&')                   -
AUDIT(FAILURES(READ)SUCCESS(READ))    -
UACC(NONE)                    -
ADDMEM(RRRRRRRR.SECURITY)
PERMIT xxxxMØØ9               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(RRRRRRRR)
PERMIT xxxxMØØ9               -
       CLASS(VCICSCMD)        -
       ACCESS(READ)           -
       ID(xxxxGMØ9)
```

Doc Farmer
Independent Security Consultant (USA)

RACF Update on the Web

Code from individual articles of *RACF Update*, and complete issues in PDF format, can be accessed on our Web site, at:

www.xephon.com/racf

You will be asked to enter a word from the printed issue.

Revoking unused userids a flexible way

When I got a request to revoke userids that hadn't been used for the last 90 days, I thought it would be easy. When I looked in the RACF manual I found the following SETROPTS parameter:

```
INACTIVE(unused-userid-interval)
```

This specifies the number of days (1 to 255) that a userid can remain unused and still be considered valid. RACF user verification checks the number of days since the last time the user successfully accessed the system against the INACTIVE value and, if the former is larger, revokes the user's right to use the system. If you specify INACTIVE, INITSTATS must be in effect.

INITSTATS specifies that statistics available during RACF user verification are to be recorded. These statistics include the date and time the user was verified by RACF, the number of user verifications that specified a particular group, and the date and time of the user's last requested verification with a particular group. If you specify INACTIVE, REVOKE, HISTORY, or WARNING, INITSTATS must be in effect. INITSTATS is in effect when RACF is using a newly-initialized database.

The parameter INACTIVE takes effect system wide. This works fine if *all* userids should be handled the same way. Unfortunately we have different classes of userids that have to be handled in different ways: some should be revoked after expiration of the unused time value, some should only be recommended to be revoked, and others should never be revoked.

What do you do with the revoked userids? Perhaps they should be recommended for deletion from the system after a further 90 days.

To handle my revoke request a bit more flexibly, I have written

a little REXX procedure named RACFPRF, which is parameter driven and performs all the necessary tasks, like checking the revoke limit, checking the delete limit, checking the userids against exception tables for revoke and delete, writing protocol files for revoke and delete recommendations, and sending the delete recommendations by e-mail to the RACF administrator.

Input to the RACFPRF procedure consists of two files:

1 The output from the **LU *** command, which is directed to a sequential file via batch TSO (see attached job RACFLU).

2 The parameter file to control execution.

The parameter file consists of three different kinds of entry:

1 Comments

2 Parameters

3 Tables.

Lines with an asterisk (*) as the first character are treated as comment.

The general syntax of the parameters is:

```
parameter : value.
```

For example:

- The file where the job for mailing the delete recommendation is stored:

  ```
  DEL_MSG_DSN: filename
  ```

- The file where the executed revoke commands are logged. This file can be used as an interface to other systems:

  ```
  REVOKE_PROT_DSN:filename
  ```

- The volume where the revoke protocol file resides:

  ```
  REVOKE_PROT_VOL:volid
  ```

- The file where the delete recommendations are logged.

This file can be used as an interface to other systems:

```
DELETE_PROT_DSN:filename
```

- Volume where the revoke protocol file resides:

```
DELETE_PROT_VOL:volid
```

- The number of days for which a userid must be unused before it is revoked:

```
DAYS_NOT_USED:n
```

- The number of days for which a userid must be revoked before it is recommended for deletion:

```
DAYS_REVOKED:n
```

The general syntax of tables is Tablename or 1 to *n* table entries.

The syntax of a table entry is:

```
typ name
```

where *typ* can be G for all users of a group, or U so a user name can be given completely or generically (eg UDB*AD).

Table examples include:

- A table for excluding userids from execution. All entries defined here are excluded from revoke and delete checking respectively:

```
TAB_EXCLUDE:
```

- A table for excluding userids from revoke processing. All entries defined here are checked against the DAYS_NOT_USED parameter, but there is only a warning, not a revoke:

```
TAB_NO_REVOKE:
```

- A table for excluding userids from delete recommendation. All entries defined here are checked against the DAYS_REVOKED parameter, but instead of the userids

being put in the delete recommendation file, a warning is given.

TAB_NO_DELETE:

- A table for mail addresses of administrators who should receive the delete recommendations:

TAB_MAIL:

JCL EXAMPLES

```
//A001D010 JOB ,'U802259',
//            MSGLEVEL=(1,1),MSGCLASS=Z,
//            CLASS=A,REGION=4M,
//            NOTIFY=&SYSUID,
//            USER=,GROUP=,PASSWORD=
//*
//* LIB: PBASLB.SYST.JCLLIB(A001D010)
//* GDE: MVS POST INSTALLATION
//* DOC: DELETE AUSGABE FILE VON RACF LU COMMAND
//*
//*
//DELLIST  EXEC PGM=IDCAMS
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
  DELETE PBASLB.RACF.USER.LIST

//A001D020 JOB ,'U802259',
//            MSGLEVEL=(1,1),MSGCLASS=Z,
//            CLASS=A,REGION=4M,
//            NOTIFY=&SYSUID,
//            USER=URACFAD,GROUP=,PASSWORD=
//*
//* LIB: PBASLB.SYST.JCLLIB(A001D020)
//* GDE: MVS POST INSTALLATION
//* DOC: ERSTELLEN AUSGABE RACF COMMAND LISTUSER ALLER USERIDS
//*
//*
//RACFBAT EXEC TSOBATCH
//T.SYSTSPRT DD  DSN=PBASLB.RACF.USER.LIST,DISP=(NEW,CATLG)
//T.SYSTSIN  DD  *
  LU *

//A001D030 JOB ,'U802259',
//            MSGLEVEL=(1,1),MSGCLASS=Z,
//            CLASS=A,REGION=4M,
//            NOTIFY=&SYSUID,
```

```
//            USER=URACFAD,GROUP=,PASSWORD=
//*
//* LIB: PBASLB.SYST.JCLLIB(A001D030)
//* GDE: MVS POST INSTALLATION
//* DOC: REVOKE USERIDS NOT USED FOR MORE THEN 90 DAYS
//*
//*
//RACFBAT EXEC TSOBATCH
//T.SYSTSIN  DD  *
  %RACFPRF
//T.RACFLU   DD DSN=PBASLB.RACF.USER.LIST,DISP=SHR
//T.PARMLI   DD DSN=SYNPU1.SYST.PARMLIB(RACFPRF),DISP=SHR

//TSOBATCH PROC MBR=IKJEFT1B
//*
//ALLOC    EXEC PGM=IEFBR14
//ISPPROF  DD  DSN=&ISPPROF(DUMMY),DISP=(NEW,PASS),
//             LIKE=PBASLB.ISPFBAT.ISPPROF,DSNTYPE=LIBRARY
//T        EXEC PGM=&MBR,
//            DYNAMNBR=35,
//            REGION=512K
//SYMDEF   INCLUDE MEMBER=SYMDEF
//ISPPROF  DD  DSN=&ISPPROF,DISP=(OLD,DELETE,DELETE)
//ISPPLIB  DD  DSN=ISP.SISPPENU,DISP=SHR
//ISPTLIB  DD  DSN=ISP.SISPTENU,DISP=SHR
//ISPMLIB  DD  DSN=ISP.SISPMENU,DISP=SHR
//ISPSLIB  DD  DSN=ISP.SISPSLIB,DISP=SHR
//SYSIN    DD  TERM=TS,SYSOUT=*
//SYSPRINT DD  TERM=TS,SYSOUT=*
//SYSPROC  DD  DSN=SYNPU1.&SYSVNRN..CLIST,DISP=SHR
//         DD  DSN=SYNPU1.SYST.CLIST,DISP=SHR
//         DD  DSN=ISP.SISPCLIB,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
```

LOG FILE

```
  %RACFPRF
*************************************************************
*
* RACFPRF was executed with the following Properties:'
* Rundate      : 18/03/03--21:09:08
* Parameterfile: SYNPU1.SYST.PARMLIB
* Inputfile    : PBASLB.RACF.USER.LIST
* Protokolfile : SYNPU1.RACF.REVOKE.DATAF
* Deleteliste : SYNPU1.RACF.DELETE.DATAF
* Mailfile    : PBASLB.SYST.JCLLIB(A001D040)
* Number of days notused befor User is revoked      : 90
* Number of days revoked befor Deleterecommendation : 90
```

```
* From Checking and Revoke excluded:
*               Group ONTOP
*               User  IRR*
*               User  OEDFLTU
*               Group OMVSGRP
*               Group IMWEB
*               User  PUBLIC
*               User  INTERNAL
*               User  PRIVATE
*               User  FWKERN
*               User  UDB*AD
*               User  UDB*OP
*               User  UDB*RW
* After Check no Revoke (Warning only):
*               Group SYS1
* After Check no Delete recommendation (Warning only):
*               User  P88888*
*               User  U999999
*               User  UØØ7616
*               Group MAKLER
*               User  P11*
*               User  P21*
*               User  P31*
*               User  P41*
*               User  P51*
*               User  P61*
*               User  P71*
*               User  P81*
*               User  P8Ø9814
*               User  P8Ø9817
*
****************************************************************
IRRCERTA CERTAUTH ANCHOR        excluded by condition U IRR*
IRRMULTI CRITERIA ANCHOR        excluded by condition U IRR*
IRRSITEC SITE ANCHOR            excluded by condition U IRR*
AØ5CDP2  wasn't used since 2Ø7 days - Check if Userid is needed further
on
BPXROOT BPXROOT                excluded by condition G OMVSGRP
DBT1USR  wasn't used since 173 days - Check if Userid is needed further
on
DBT4USR  wasn't used since 383 days - Check if Userid is needed further
on
EXPLAIN  wasn't used since 13Ø days - Check if Userid is needed further
on
FWKERN UNKNOWN   excluded by condition U FWKERN
IMT1USR  wasn't used since 12Ø days - Check if Userid is needed further
on
INTERNAL UNKNOWN   excluded by condition U INTERNAL
IRRDPTAB IRRDPTAB               excluded durch Bedingung U IRR*
```

```
M002002  wasn't used since 130 days - Deleterequest suppressed
M002005  wasn't used since 130 days - Deleterequest suppressed
REVOKE P001070  Test User           18/03/03 21:10:05    91
```

RACF_PRF

```
/* REXX                                                                 */
/* Extract the User information and the Last Access date out
        of the Racflist */
/* from command  'TSO LU *'                                             */
trace o
arg ludsn parmdsn .
ludsn = strip(ludsn,'B','"')
ludsn = strip(ludsn,'B',"'")
parmdsn = strip(parmdsn,'B','"')
parmdsn = strip(parmdsn,'B',"'")
lu_rc = listdsi('RACFLU FILE')
if lu_rc = 16   /* File not allocated */
   then "ALLOC FI(RACFLU) DA('"ludsn"') SHR REUSE"
   else ludsn = sysdsname
pa_rc = listdsi('PARMLI FILE')
if pa_rc = 16   /* File not allocated */
   then "ALLOC FI(PARMLI) DA('"parmdsn"') SHR REUSE"
   else parmdsn = sysdsname
call read_parms
call header
lines = 500
start = 1
ret = 0
j = 0
k1 = 0
dc = 0
dcp= 0
do while ret = 0
   'EXECIO 'lines' DISKR RACFLU 'start '( STEM EIN.'
   ret = rc
   do i = 1 to ein.0
if i > 100 then trace o
   if pos('USER=',ein.i) > 0
      then do
           parse var ein.i . 'USER='user . 'NAME=' name,
              'OWNER=' owner . 'CREATED=' created .
           aus = user';'name';'created';'
           attrib = ''
           iterate
           end
   pos_dflt = pos('DEFAULT-GROUP=',ein.i)
   if pos_dflt > 0
```

```
      then do
           dfltgrp = substr(ein.i,pos_dflt+14,8)
           dfltgrp = strip(dfltgrp,'B')
           aus = aus||dfltgrp';'
           iterate
           end
   if pos('ATTRIBUTES=',ein.i) > 0
      then do
           parse var ein.i . 'ATTRIBUTES=' attrib1
           attrib = attrib ||strip(attrib1,'T')' '
           iterate
           end
   if pos('LAST-ACCESS=',ein.i) > 0
      then do
           parse var ein.i . 'LAST-ACCESS=' lacc .
           aus = aus||attrib';'lacc';'
           call prf_revoke aus
           iterate
           end
   end
   start = start + lines
   if start > 1000 then trace o
end
'EXECIO 0 DISKR RACFLU ( FINIS'
if k1 > 0      /* There are revoked Userids  */
   then do
     if sysdsn("'"rvk_prot"'") = 'OK'
      then "ALLOC FI(USEROUT) DA('"rvk_prot"') MOD"
      else "ALLOC FI(USEROUT) DA('"rvk_prot"') NEW VOLUME("rvk_vol")",
           "RECFM(F,B) LRECL(80)"
     'EXECIO 'k1' DISKW USEROUT ( STEM REVOKE. FINIS'
     'FREE FI(USEROUT)'
     end
if dc > 0 then call sendmail_deleteuser
          else call sendmail_iefbr14
exit

PRF_REVOKE:
/*-------------------------------------------------------------*/
/*                                                             */
/* Display all Userids that weren't used for the last n days   */
/*                                                             */
/*-------------------------------------------------------------*/
arg parmlist
parse var parmlist user';'name';'def';'dfltgrp';'attr';'lacc';'
do ix = 1 to anz_excl
   parse var excl.ix excl_typ excl_obj
   pos1 = pos('*',excl_obj)
   if pos1 > 0 then l = pos1 - 1
```

```
            else l = length(excl_obj)
   pos2 = lastpos('*',excl_obj)
   if pos2 > Ø then lr = length(excl_obj) - pos2
            else lr = length(excl_obj)
   select
     when excl_typ = 'G' & left(excl_obj,l) = left(dfltgrp,l),
                         & right(excl_obj,lr) = right(dfltgrp,lr)
       then do
            say user name 'excluded by condition 'excl.ix
            return
            end
     when excl_typ = 'U' & left(excl_obj,l) = left(user,l),
                         & right(excl_obj,lr) = right(user,lr)
       then do
            say user name 'excluded by condition 'excl.ix
            return
            end
     otherwise nop
   end
end
warnkz = Ø
do ix = 1 to anz_warn
   parse var warn.ix warn_typ warn_obj
   pos1 = pos('*',warn_obj)
   if pos1 > Ø then l = pos1 - 1
            else l = length(warn_obj)
   pos2 = lastpos('*',warn_obj)
   if pos2 > Ø then lr = length(warn_obj) - pos2
            else lr = length(warn_obj)
   select
     when warn_typ = 'G' & left(warn_obj,l) = left(dfltgrp,l),
                         & right(warn_obj,lr) = right(dfltgrp,lr)
       then warnkz = 1
     when warn_typ = 'U' & left(warn_obj,l) = left(user,l),
                         & right(warn_obj,lr) = right(user,lr)
       then warnkz = 1
     otherwise nop
   end
end
ndelkz = Ø
do ix = 1 to anz_ndel
   parse var ndel.ix ndel_typ ndel_obj
   pos1 = pos('*',ndel_obj)
   if pos1 > Ø then l = pos1 - 1
            else l = length(ndel_obj)
   pos2 = lastpos('*',ndel_obj)
   if pos2 > Ø then lr = length(ndel_obj) - pos2
            else lr = length(ndel_obj)
   select
```

```
     when ndel_typ = 'G' & left(ndel_obj,l) = left(dfltgrp,l),
                         & right(ndel_obj,lr) = right(dfltgrp,lr)
       then ndelkz = 1
     when ndel_typ = 'U' & left(ndel_obj,l) = left(user,l),
                         & right(ndel_obj,lr) = right(user,lr)
       then ndelkz = 1
     otherwise nop
   end
end
/*-------------------------------------------------------------------*/
/* if last access date unknown use definition date instead           */
/*-------------------------------------------------------------------*/
if lacc = 'UNKNOWN' then do
                         parse var def jj'.'ttt
                         jd = jj||ttt
                         ldays = date('B',jd,'J')
                         ein.i = ein.i'        '
                         end
                    else do
                         parse var lacc jd'/'.
                         parse var jd jj'.'ttt
                         jd = jj||ttt
                         ldays = date('B',jd,'J')
                         end
delta = date('B') - ldays

  if  delta > n
    then do
         if find(attr,'REVOKED') = 0
            then do
                 if warnkz = 1
                    then do
                      say left(user,8)' wasn't used since 'delta' days',
                          ' - Check if Userid is needed further on'
                      end
                    else do
                 if date(s) < 20020930
              then say 'ALU ('user') REVOKE /*' delta name dfltgrp
              else do
                       'ALU ('user') REVOKE'
                       k1 = k1 + 1
                       revoke.k1 = 'REVOKE' left(user,8)
                       revoke.k1 = revoke.k1' 'left(name,20) date('E')
                       revoke.k1 = revoke.k1' 'time()' 'right(delta,5)
                       say revoke.k1 /* Protokoll in SYSOUT */
                 end
                       end
                 end
         end
```

```
  if  delta > n+nd
    then do
         if find(attr,'REVOKED') > Ø
            then do
                 if ndelkz = 1
                    then do
                      say left(user,8)' wasn't used since 'delta' days',
                          ' - Deleterequest suppressed'
                     end
                    else do
              dc = dc + 1
              dtxt.dc = left(user,8)' 'left(name,2Ø)' 'right(delta,5),
                      date('E',jd,'J')
              pnum = substr(user,2)
              if datatype(pnum) = 'NUM'
                 then do
                    dcp = dcp + 1
                    dptxt.dcp = user';'name';'delta';'date('E',jd,'J')';'
                     end
                         end
                end
         end
return

HEADER:
    say '************************************************************'
   say '*'
   say '* RACFPRF was executed with the following Properties:'
   say '* Rundate      :' date('E')'--'time()
   say '* Parameterfile:' parmdsn
   say '* Inputfile    :' ludsn
   say '* Protokolfile :' rvk_prot
   if del_prot ¬= ''
      then say '* Deletelist  :' del_prot
   say '* Mailfile    :' deldsn
   say '* Number of days notused befor User is revoked      :' n
   say '* Number of days revoked befor Deleterecommendation :' nd
   say '* From Checking and Revoke excluded:'
   do i = 1 to anz_excl
      parse var excl.i typ name
      if typ = 'G' then typ = 'Group'
                   else typ = 'User '
      say '*               'typ' 'name
      end
   say '* After Check no Revoke (Warning only):'
   do i = 1 to anz_warn
      parse var warn.i typ name
      if typ = 'G' then typ = 'Group'
                   else typ = 'User '
```

```
      say '*               'typ' 'name
      end
   say '* After Check no Delete recommendation (Warning only):'
   do i = 1 to anz_ndel
      parse var ndel.i typ name
      if typ = 'G' then typ = 'Group'
                   else typ = 'User '
      say '*               'typ' 'name
      end
   say '*'
    say '*****************************************************************'
return

SENDMAIL_DELETEUSER:
   "ALLOC FI(DELMAIL) DA('"deldsn"') SHR REUSE"
   parse var deldsn . '('jn')'.
   head.1  = "//"jn" JOB ,'U8Ø2259',"
   head.2  = "//             MSGLEVEL=(1,1),MSGCLASS=Z, "
   head.3  = "//             CLASS=G,REGION=4M, "
   head.4  = "//             NOTIFY=&SYSUID, "
   head.5  = "//             USER=,GROUP=,PASSWORD= "
   head.6  = "//* "
   head.7  = "//* LIB: "deldsn
   head.8  = "//* JOB GENERIERT DURCH PROZEDUR RACFPRF"
   head.9  = "//JES2MAIL EXEC JES2MAIL,PARMS=/CONFIGW2 "
   head.1Ø = "//SYSIN    DD * "
   head.11 = "FILE=DD:MESSAGE "
   head.12 = "END "
   head.13 = "//MESSAGE  DD * "
   do hc = 1 to anz_mail
      h = hc + 13
      head.h  = "TO: "mail.hc
      end
   h = h + 1
   head.h  = "FROM: ROMAN.HAWLITSCHEK@XXXX.DE "
   h = h + 1
   head.h  = "SUBJECT: Delete recommendation for revoked Userids"
   h = h + 1
   head.h  = "  "
   h = h + 1
   head.h  = "Following User's are revoked since "nd" days"
   h = h + 1
   head.h  = " and should be deleted:"
   h = h + 1
   head.h  = "                                Days"
   h = h + 1
   head.h  = "Userid   Name                n.used   Last Access "
   h = h + 1
   head.h  = "================================================="
```

```
   "EXECIO "h" DISKW DELMAIL ( STEM HEAD."
   "EXECIO "dc" DISKW DELMAIL ( STEM DTXT."
   bot.1 = "                        "
   bot.2 = "MfG.   DV/Systemtechnik"
   bot.3 = "."
   "EXECIO 3 DISKW DELMAIL ( STEM BOT. FINIS"
   "FREE FI(DELMAIL)"
   if del_prot ¬= ''
      then do
           if sysdsn("'"del_prot"'") = 'OK'
              then "ALLOC FI(DELEOUT) DA('"del_prot"') SHR REUSE"
              else "ALLOC FI(DELEOUT) DA('"del_prot"')",
                   "NEW VOLUME("del_vol") RECFM(F,B) LRECL(8Ø)"
           "EXECIO "dcp" DISKW DELEOUT ( STEM DPTXT. FINIS"
           'FREE FI(DELEOUT)'
           end
return

SENDMAIL_IEFBR14:
   "ALLOC FI(DELMAIL) DA('"deldsn"') SHR REUSE"
   parse var deldsn . '('jn')'.
   head.1  = "//"jn" JOB ,'U8Ø2259',"
   head.2  = "//            MSGLEVEL=(1,1),MSGCLASS=Z, "
   head.3  = "//            CLASS=G,REGION=4M, "
   head.4  = "//            NOTIFY=&SYSUID, "
   head.5  = "//            USER=,GROUP=,PASSWORD= "
   head.6  = "//* "
   head.7  = "//* LIB: "deldsn
   head.8  = "//* JOB GENERIERT DURCH PROZEDUR RACFPRF"
   head.9  = "//DUMMY    EXEC PGM=IEFBR14 "
   "EXECIO 9 DISKW DELMAIL ( STEM HEAD. FINIS"
   "FREE FI(DELMAIL)"
return

READ_PARMS:
   'EXECIO * DISKR PARMLI ( STEM INPPARM. FINIS'
   anz_excl = Ø
   anz_warn = Ø
   anz_ndel = Ø
   anz_mail = Ø
   del_prot = ''
   do pp = 1 to inpparm.Ø
      select
        when left(inpparm.pp,1) = '*' then iterate
        when pos('DEL_MSG_DSN:',inpparm.pp) > Ø
         then do
              parse var inpparm.pp . 'DEL_MSG_DSN:' deldsn .
              deldsn = strip(deldsn,'B')
              tabtyp = ''
```

```
      iterate
      end
when pos('REVOKE_PROT_DSN:',inpparm.pp) > 0
 then do
      parse var inpparm.pp . 'REVOKE_PROT_DSN:' rvk_prot .
      rvk_prot = strip(rvk_prot,'B')
      tabtyp = ''
      iterate
      end
when pos('REVOKE_PROT_VOL:',inpparm.pp) > 0
 then do
      parse var inpparm.pp . 'REVOKE_PROT_VOL:' rvk_vol .
      rvk_vol = strip(rvk_vol,'B')
      tabtyp = ''
      iterate
      end
when pos('DELETE_PROT_DSN:',inpparm.pp) > 0
 then do
      parse var inpparm.pp . 'DELETE_PROT_DSN:' del_prot .
      del_prot = strip(del_prot,'B')
      tabtyp = ''
      iterate
      end
when pos('DELETE_PROT_VOL:',inpparm.pp) > 0
 then do
      parse var inpparm.pp . 'DELETE_PROT_VOL:' del_vol .
      del_vol = strip(del_vol,'B')
      tabtyp = ''
      iterate
      end
when pos('DAYS_NOT_USED:',inpparm.pp) > 0
 then do
      parse var inpparm.pp . 'DAYS_NOT_USED:' n .
      n = strip(n,'B')
      tabtyp = ''
      iterate
      end
when pos('DAYS_REVOKED:',inpparm.pp) > 0
 then do
      parse var inpparm.pp . 'DAYS_REVOKED:' nd .
      nd = strip(nd,'B')
      tabtyp = ''
      iterate
      end
when pos('TAB_EXCLUDE:',inpparm.pp) > 0
 then do
      tabtyp = 'EXCLUDE'
      iterate
      end
```

```
         when pos('TAB_NO_REVOKE:',inpparm.pp) > 0
          then do
               tabtyp = 'NO_REVOKE'
               iterate
               end
         when pos('TAB_NO_DELETE:',inpparm.pp) > 0
          then do
               tabtyp = 'NO_DELETE'
               iterate
               end
         when pos('TAB_MAIL:',inpparm.pp) > 0
          then do
               tabtyp = 'MAIL'
               iterate
               end
         otherwise call load_tab tabtyp
       end
   end
return

LOAD_TAB:
   arg tabtyp .
   select
     when tabtyp = '' then return
     when tabtyp = 'EXCLUDE'
       then do
            anz_excl = anz_excl + 1
            hi = anz_excl
            excl.hi = strip(inpparm.pp,'B')
            end
     when tabtyp = 'NO_REVOKE'
       then do
            anz_warn = anz_warn + 1
            hi = anz_warn
            warn.hi = strip(inpparm.pp,'B')
            end
     when tabtyp = 'NO_DELETE'
       then do
            anz_ndel = anz_ndel + 1
            hi = anz_ndel
            ndel.hi = strip(inpparm.pp,'B')
            end
     when tabtyp = 'MAIL'
       then do
            anz_mail = anz_mail + 1
            hi = anz_mail
            mail.hi = strip(inpparm.pp,'B')
            end
     otherwise nop
```

```
   end
return
```

RACFPRF.PARMS

```
DEL_MSG_DSN:PBASLB.SYST.JCLLIB(A001D040)
REVOKE_PROT_DSN:SYNPU1.RACF.REVOKE.DATAF
REVOKE_PROT_VOL:ONPPD1
RESET_PROT_DSN:SYNPU1.RACF.RESET.DATAF
RESET_PROT_VOL:ONPPD1
DELETE_PROT_DSN:SYNPU1.RACF.DELETE.DATAF
DELETE_PROT_VOL:ONPPD1
DAYS_NOT_USED:90
DAYS_REVOKED:90
*
*-- GENERAL STRUCTURE OF THE EXCLUDE TABLES     ----------------------*/
* STRUCTURE of the Element to be excluded : TYP NAME
* TYP:  G --> GROUP    All users of this group
*       U --> USER
* NAME: Complete or geberic name with *  for example UDB*AD
*
*-- EXCLUDE TABLES FOR REVOKE AND DELETE      ----------------------*/
TAB_EXCLUDE:
* IBM SERVIVE USER
   G ONTOP
* RACF INTERNAL DEFAULTUSER
   U IRR*
* OMVS DEFAULT USER DEFINITION MUST BE THERE
* FOR CICS AND FTP
   U OEDFLTU
* USER FOR OMVS-SERVICES
   G OMVSGRP
* USER FOR OMVS-WEBSERVER
   G IMWEB
* USER FOR OMVS-WEBSERVER
   U PUBLIC
* USER FOR OMVS-WEBSERVER
   U INTERNAL
* USER FOR OMVS-WEBSERVER
   U PRIVATE
* USER FOR OMVS-SERVICES
   U FWKERN
* SYSTEMDEFAULT ADMIN USER FOR DB2
   U UDB*AD
* SYSTEMDEFAULT OPER  USER FOR DB2
   U UDB*OP
* USERID FüR R/W ZUGRIFF BEI DYNAMISCHEN SQL
   U UDB*RW
```

```
*-- EXCLUDE TABLE FOR REVOKE OF USERID      ----------------------*/
TAB_NO_REVOKE:
   G SYS1
*-- EXCLUDE TABLE FOR DELETE OF USERID      ----------------------*/
TAB_NO_DELETE:
* TSO-USER WITH STANDARD ACCESS FOR PROCEDURE TESTING
   U U999999
* HERR BUSTOS IBM
   U U0070616
* ALLE USER OF GROUP MAKLER
   G MAKLER
* ALLE SECUNDARY USERIDS FOR SPECIAL TASKS
   U P11*
   U P21*
   U P31*
   U P41*
   U P51*
   U P61*
   U P71*
   U P81*
*-- MAILADDRESS TABLE FOR DELETE RECOMMENDATION --*/
TAB_MAIL:
   ROMAN.HAWLITSCHEK@XXXX.DE
```

Roman Hawlitschek
Systems Programmer (Germany)

If you have ever experienced any difficulties with RACF, or made an interesting discovery, you could receive a cash payment simply by telling us about it.

More information about contributing an article, plus an explanation of our terms and conditions, can be found at www.xephon.com/nfc.

If you have an idea for an article, please contact the editor, Trevor Eddolls, at trevore@xephon.com.

An LDAP client program for updating RACF userid attributes

LDAP servers are becoming prevalent repositories for corporate data. To that end, the OS/390 (z/OS) Security Server has been enhanced to include an LDAP server that can run as a traditional MVS started task, batch job, or even in a pseudo open system environment as a process under the OS/390 USS kernel. In all cases, a full range of LDAP server functions are provided. The OS/390 LDAP server is also capable of providing an interface into the RACF database.

The OS/390 LDAP server was first introduced with OS/390 2.5 and it required DB2 to maintain the directory information. This LDAP interface used an internal protocol known as RDBM for accessing directory data. OS/390 2.7 extended the LDAP server to interface with the RACF database – the RACF LDAP interface uses an internal protocol known as SDBM to handle the mapping of requests between LDAP and RACF. A second DB2 interface for the LDAP server was introduced with OS/390 2.10 – this LDAP interface uses an internal protocol known as TDBM. The TDBM protocol is more flexible and provides better performance than the RDBM interface, and TDBM is intended eventually to completely supersede RDBM. Although the RDBM and TDBM interfaces are useful for general-purpose LDAP access and customer schema definition, this article will focus on the use of the SDBM interface for the OS/390 LDAP server.

Use of LDAP falls into three broad operational categories:

1 Authentication – this would include operations such as bind and unbind, which are used to establish or disable connection to an LDAP server. Also included in this category would be abandon operations that are used to abandon asynchronous LDAP requests.

2 Query – this would include any operation involving a search or compare operation that can result in the retrieval of information from a directory.

3 Update – this would include operations that add, delete, or modify entries in the LDAP database.

FUNCTIONAL OVERVIEW

The LDAP client program provided with this article shows examples of all the above operational categories. The ATTRUPD C program included with this article is an LDAP client program that uses non-SSL communication with the OS/390 LDAP server. An ldap_bind_s() function call is used to establish the RACF authorization under which the subsequent search and update operations are to occur. Examples of query operations used in the ATTRUPD program include ldap_search_s(), ldap_count_entries(), ldap_first_attribute(), ldap_get_values(), and ldap_next_attribute(). The ATTRUPD program uses an ldap_modify_s() function call to update the specified RACF attribute. This would be an example of an update operation. To provide a clean termination, the ATTRUPD program ends its communication with the LDAP server with an ldap_unbind() function call. By convention, function calls suffixed with _s are performed synchronously. Appropriately, comparable asynchronous versions of these functions also exist – the asynchronous functions can be invoked without the _s suffix and outcomes can be checked later with the ldap_result() function call.

The ATTRUPD program, as coded, is set up to run on OS/390 or z/OS systems under USS (the OMVS environment), but can be modified to run on any system that supports the LDAP client APIs. What is required is an LDAP server running on OS/390 or z/OS that is configured to operate with the SDBM protocol (ie an LDAP server that is configured to use RACF as

a backend database). ATTRUPD accepts seven tag-defined arguments as follows:

1 –h *hostname* – specifies the hostname or IP address of the target LDAP server.

2 –p *port#* – specifies the port number the LDAP server is listening on. This is the only argument that has a default setting. Port 389 will be used if the –p argument is not specified.

3 –b *bind_userid* – specifies the RACF userid that is to be used on the ldap_bind_s() function call. If the bind is successful, this will be the userid that is used for the subsequent search and modify requests (ie this userid must have sufficient RACF authority to perform the search and modify operations or the requests will fail).

4 –c *bind_userid_pwd* – specifies the password value to be used for validating the bind userid on the ldap_bind_s() function call.

5 –u *target_userid* – specifies the userid to be used for the RACF attribute extract and update request.

6 –a *attribute_name* – specifies the LDAP server RACF attribute name that is to be updated. Chapter 17, *Accessing RACF Information* in the *z/OS Secure Way Security Server LDAP Server Administration and Usage Guide* discusses the appropriate LDAP-style attribute names that can be specified. Some examples include racfpassword, racfprogrammername, racfdefaultgroup, and racfomvsuid.

7 –v *attribute_value* – specifies the updated value that should be applied to the corresponding –a attribute name for attribute update.

Prior to compiling the program, you will need to decide whether you want to enable some minimal diagnostic output

or not. If you want to produce diagnostic output, the program code line containing:

```
//#define DEBUG 1
```

should be uncommented. With this code line uncommented, diagnostic messages will be produced for even normal program results. If the program is being compiled, prelinked, and linkedited on an OS/390 or z/OS system, the SYSLIB DD in the compile job should include your standard C header datasets as well as GLD.SGLDHDRC (for the LDAP header files) and the prelink job should include module GLDCLDPX from dataset GLD.SGLDEXPC.

RUNNING THE PROGRAM

Once the program has been compiled and linked on whichever client system you plan to use it, it can be run against an OS/390 or z/OS target system that is running an LDAP server with the SDBM protocol enabled. An example ATTRUPD program invocation might look something like:

```
attrupd -h 10.0.10.2 -p 389 -b admuser -c admpwd -u upduser -a
racfpassword -v newpwd
```

In this case, we have linked the program into an executable module named attrupd. The expected LDAP server is running at IP address 10.0.10.2 and is listening on port 389. The userid and password to be used for the bind operation are ADMUSER and ADMPWD respectively. If the bind operation is a success and ADMUSER has sufficient RACF authority to update RACF attributes for UPDUSER, UPDUSER's RACF password will be updated to NEWPWD. Status messages and error message will be produced by attrupd and directed to stdout. Some sample output from running the ATTRUPD program is shown below:

```
  Attribute: objectclass
  Attribute: racfid                          Values: UPDUSER
  Attribute: racfprogrammername              Values: LDAP TEST ID
  Attribute: racfowner                       Values:
racfid=RACFOWN,profiletype=USER,SYSPLEX=SYSPLEX1
```

```
  Attribute: racfauthorizationdate             Values: ØØ.329
  Attribute: racfdefaultgroup                  Values:
racfid=TESTGRP,profiletype=GROUP,SYSPLEX=SYSPLEX1
  Attribute: racfpasswordchangedate            Values: Ø4.156
  Attribute: racfpasswordinterval              Values: 18Ø
  Attribute: racfattributes                    Values: NONE
  Attribute: racfrevokedate                    Values: NONE
  Attribute: racfresumedate                    Values: NONE
  Attribute: racflastaccess                    Values: Ø4.156/Ø9:34:Ø8
  Attribute: racfclassname                     Values: NONE
  Attribute: racfinstallationdata              Values: NO-INSTALLATION-
DATA
  Attribute: racfdatasetmodel                  Values: NO-MODEL-NAME
  Attribute: racflogondays                     Values: ANYDAY
  Attribute: racflogontime                     Values: ANYTIME
  Attribute: racfconnectgroupname              Values:
racfid=TESTGRP,profiletype=GROUP,SYSPLEX=SYSPLEX1
                         racfid=ALTGRP1,profiletype=GROUP,SYSPLEX=SYSPLEX1
                         racfid=ALTGRP2,profiletype=GROUP,SYSPLEX=SYSPLEX1
  Attribute: racfsecuritylevel                 Values: NONE SPECIFIED
  Attribute: racfsecuritycategorylist          Values: NONE SPECIFIED
  Attribute: racfsecuritylabel                 Values: NONE SPECIFIED
  Attribute: racfomvsuid                       Values: ØØØØØØ1105
  Attribute: racfomvshome                      Values: /
  Attribute: racfomvsinitialprogram            Values: /bin/sh
  Attribute: racfomvsmaximumcputime            Values: NONE
  Attribute: racfomvsmaximumaddressspacesize   Values: NONE
  Attribute: racfomvsmaximumfilesperprocess    Values: NONE
  Attribute: racfomvsmaximumprocessesperuid    Values: NONE
  Attribute: racfomvsmaximumthreadsperprocess  Values: NONE
  Attribute: racfomvsmaximummemorymaparea      Values: NONE
  Attribute: safaccountnumber                  Values: ACCT#
  Attribute: safmessageclass                   Values: H
  Attribute: safdefaultloginproc               Values: ISPFPROC
  Attribute: saflogonsize                      Values: ØØØØ4Ø96
  Attribute: safmaximumregionsize              Values: ØØØØ8192
  Attribute: safuserdata                       Values: ØØØØ
  Attribute: safdefaultcommand                 Values:
  Attribute update successful for upduser
```

This shows the LDAP-style attribute names that are returned. Some attributes such as racfauthorizationdate, racfpasswordchangedate, racfpasswordinterval, and racflastaccess are not modifiable – they are display-only attributes. Also, notice from the output that attributes are displayed for the userid RACF BASE segment (attributes racfid through racfsecuritylabel) as well as the OMVS segment

(attributes racfomvsuid through racfomvsmaximummemorymaparea) and for the TSO segment (attributes safaccountnumber through safdefaultcommand).

If the display of the userid's current attributes is not desired, the program code line containing the following should be uncommented prior to compiling the program source code:

```
//goto NO_ATTR_LIST;
```

CONCLUSION

This program is a useful first attempt at creating an LDAP client program. It demonstrates many of the primary LDAP client function APIs. With only very minor changes, the program could be modified to act as a verification front-end to any Internet application so it has a very practical foundation. Perhaps you can find some uses in your own environment.

ATTRUPD C

```
/*
 * This program is an LDAP client program that can be used to update
 * RACF userid attributes on a system where the RACF OS/390 or z/OS
 * LDAP server is running with the SDBM database mode.
 *
 * The program accepts seven tag identified arguments as follows:
 *
 * -a Attribute name
 * -b Bind userid
 * -c Credentials (password) for bind userid
 * -h Target hostname or IP address
 * -p Port # for target LDAP server
 * -u Target userid for attribute extract/update
 * -v Value for attribute specified in arg_a
 *
 * A sample program invocation might look something like:
 *
 * attrupd -h 10.0.10.2 -p 389 -b admuser -c admpwd -u trgtusr
 *         -a racfpassword -v newpwd
 *
 * where the above command should be entered on one logical command line
 * with appropriate command wrap.  In the example, a system whose IP
 * address is 10.0.10.2 is running a RACF LDAP server that is listening
 * on port 389.  The ldap_bind will be attempted using a userid of
```

```
 * admuser and a password of admpwd.  If the ldap_bind is successful
 * and the admuser userid has sufficient RACF authority, the password
 * for userid trgtuser will be reset to newpwd.
 *
 * Chapter 17. Accessing RACF Information in the z/OS Secure Way
 * Security Server LDAP Server Administration and Usage Guide
 * discusses the appropriate LDAP-style attribute names that can be
 * used for the -a specified option.  The -v values must be consistent
 * with the specified -a attribute name or the attribute update request
 * will fail.  Any -v values containing blanks should be enclosed
 * in ".  For example, if you wanted to change the programmer name
 * associated with a RACF userid and the name contained blanks, you
 * could use the following arguments:
 *
 *   -a racfprogrammername -v "New Name"
 *
 * This program can be used on any system that supports a C compiler
 * and the LDAP client functions, but is specifically set up to run
 * under z/OS OMVS or through the BPXBATCH program interface.
 *
 * If this program is to be compiled on an OS/390 or z/OS system
 * with the IBM C/C++ compiler, remember to convert all occurrences
 * of '[' to x'AD' and ']' to x'BD'
 *
 */

#define mvs
#ifdef mvs
#pragma runopts("POSIX(ON)")
#define _OPEN_THREADS
#define MVS_PTHREADS
#define _OE_SOCKETS
#define _SHARE_EXT_VARS
#define LOCALCP_TRANSLATION
#define EBCDIC_PLATFORM
#define LONGMAP
#endif

//#define DEBUG 1    // Uncomment this #define to enable some diagnostic
msgs.

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <ldap.h>

static char *password = NULL;
static char *admpwd = NULL;
static char *binddn = NULL;
```

```
static char *binddn2 = NULL;
static char *ldaphost = NULL;
static int   ldapport = LDAP_PORT;
static int   ldapversion = LDAP_VERSION3;
static char *ATTR_NAME1 = "objectclass";
static char *VAL_HDR = "  Values: ";
char *attribute;
char **values;

main(int argc, char *argv[])
{
  int rc, authmethod;
  LDAP  *ld;
  char racfid[9];
  char passwd[9];
  char adm_passwd[9];
  int i, j;
  int racfid_len, passwd_len, prefix_len;

  LDAPMessage *res;
  LDAPMessage *entry;
  BerElement *ber;
  char filler[128];
  char binddn_area[128] = {0};
  char binddn_area2[128] = {0};
  char output_line[2048];
  char attr_name[128];
  char attr_val[128];

  char arg_a[128];    // -a Attribute name
  char arg_b[128];    // -b Bind userid
  char arg_c[128];    // -c Credentials (password) for bind userid
  char arg_h[128];    // -h Target hostname or IP address
  char arg_p[128];    // -p Port # for target LDAP server
  char arg_u[128];    // -u Target userid for attribute extract/update
  char arg_v[128];    // -v Value for attribute specified in arg_a
  int  flg_a = 0;
  int  flg_b = 0;
  int  flg_c = 0;
  int  flg_h = 0;
  int  flg_p = 1;     // Set up to default to port 389
  int  flg_u = 0;
  int  flg_v = 0;

  char *vals[2];
  LDAPMod mod;
  LDAPMod *mods[2];

/*
 * Parse the incoming arguments and ensure all necessary information
```

```
 * has been provided.  The only tag/value pair that is not required
 * is -p port#.  By default, port 389 will be used if no port # is
 * provided with the command invocation.
 */

  i = 1;
  while (i < argc)
  {
    if (i == 1 || i == 3 || i == 5 || i == 7 || i == 9 || i == 11 ||
        i == 13)
    {
/*
 * argv arguments with odd numbered indices must be tag (-) prefixed.
 */
      j = strncmp(argv[i],"-",1);
      if (j != 0)
      {
        printf("Invalid flag detected %s\n",argv[i]);
        return(16);
      }
      else if (strcmp(argv[i],"-a") == 0)  // Attribute name?
      {
        strcpy(arg_a,argv[i+1]);
        flg_a = 1;
        i = i + 2;
      }
      else if (strcmp(argv[i],"-b") == 0)  // Bind userid?
      {
        strcpy(arg_b,argv[i+1]);
        flg_b = 1;
        i = i + 2;
      }
      else if (strcmp(argv[i],"-c") == 0)  // Credentials (pwd) for bind
userid?
      {
        strcpy(arg_c,argv[i+1]);
        flg_c = 1;
        i = i + 2;
      }
      else if (strcmp(argv[i],"-h") == 0)  // Target hostname or IP
address?
      {
        strcpy(arg_h,argv[i+1]);
        flg_h = 1;
        i = i + 2;
      }
      else if (strcmp(argv[i],"-p") == 0)  // Port # for target LDAP
server?
      {
        strcpy(arg_p,argv[i+1]);
```

```
        ldapport = atoi(arg_p);
        flg_p = 1;
        i = i + 2;
      }
      else if (strcmp(argv[i],"-u") == Ø)  // Userid for attr update/
extract?
      {
        strcpy(arg_u,argv[i+1]);
        flg_u = 1;
        i = i + 2;
      }
      else if (strcmp(argv[i],"-v") == Ø)  // Value for specified
attribute?
      {
        strcpy(arg_v,argv[i+1]);
        flg_v = 1;
        i = i + 2;
      }
      else
      {
        printf("Unrecognized flag %s\n",argv[i]);
        return(2Ø);
      }
    }
  }

/*
 * The argument list has been parsed.  Determine whether all the
 * necessary information has been provided with the command
 * invocation.
 */

  if (flg_a != 1 || flg_b != 1 || flg_c != 1 || flg_h != 1 ||
      flg_p != 1 || flg_u != 1 || flg_v != 1)
  {
    if (flg_a != 1)
    {
      printf("Attribute name not specified\n");
    }
    if (flg_b != 1)
    {
      printf("Bind userid not specified\n");
    }
    if (flg_c != 1)
    {
      printf("Credentials (password) for bind userid not specified\n");
    }
    if (flg_h != 1)
    {
      printf("Target hostname or IP address not specified\n");
```

```
    }
    if (flg_p != 1)
    {
      printf("Target system LDAP server port # not specified\n");
    }
    if (flg_u != 1)
    {
      printf("Target user for attribute extract/update not
specified\n");
    }
    if (flg_v != 1)
    {
      printf("Value not detected for specified attribute\n");
    }
    return(24);
  }

#if defined(DEBUG)
  printf("Attribute name %s\n",arg_a);
  printf("Bind userid %s\n",arg_b);
  printf("Credentials (password) for bind userid %s\n",arg_c);
  printf("Target system hostname or IP address %s\n",arg_h);
  printf("Target system LDAP server port # %s\n",arg_p);
  printf("Userid for attribute extract/update %s\n",arg_u);
  printf("Value for attribute update %s\n",arg_v);
#endif

  ldaphost = strcat(arg_h,"");

/*
 * At this point, the syntax of the tag/value arguments has been
 * validated.  We can attempt to establish a connection to the
 * target LDAP server.
 */

  if ((ld = ldap_init(ldaphost, ldapport)) == NULL)
  {
    perror(ldaphost);
    printf("ldap_init(%s,%d)\n",ldaphost,ldapport);
    exit(1);
  }

/*
 * Depending on the configuration of the SDBM database definition on
 * the target LDAP server, you may need to customize the binddn and
 * binddn2 values below.  Specifically, the sysplex= value in the
 * SDBM database definition of the target LDAP server should match
 * the sysplex= value specified for binddn and binddn2.
 */
```

```
  strcpy(racfid,arg_u);
  strncat(binddn_area,"racfid=",7);
  binddn = strncat(binddn_area,racfid,8);
  binddn = strncat(binddn,",profiletype=USER,sysplex=sysplex1",34);
  authmethod = LDAP_AUTH_SIMPLE;
  strcpy(adm_passwd,arg_c);
  admpwd = strcat(adm_passwd,"");
  strncat(binddn_area2,"racfid=",7);
  binddn2 = strncat(binddn_area2,arg_b,strlen(arg_b));
  binddn2 = strncat(binddn2,",profiletype=USER,sysplex=sysplex1",34);

  if (ldap_bind_s(ld, binddn2, admpwd, authmethod) != LDAP_SUCCESS)
  {
    ldap_perror(ld, "ldap_bind");
    printf("\nracfid= %s",racfid);
    exit(2);
  }

/*
 * This section of code extracts and displays, to stdout, the
 * current RACF userid attributes for the userid specified by the
 * -u tag.  If you don't want to display the userid attributes
 * prior to attempting the attribute update, uncomment the following
 * goto instruction.
 */
//goto NO_ATTR_LIST;

  rc = ldap_search_s(ld, binddn, LDAP_SCOPE_BASE, "objectclass=*",
                     NULL, 0, &res);
#if defined(DEBUG)
  printf("ldap_search_s() rc %d\n",rc);
#endif

  if (rc == 0)
  {
    rc = ldap_count_entries(ld, res);

#if defined(DEBUG)
    printf("ldap_count_entries() returned %d\n",rc);
#endif

    if (rc > 0)
    {
      entry = ldap_first_entry(ld, res);
      if (entry != NULL)
      {

#if defined(DEBUG)
        printf("ldap_first_entry() successful\n");
#endif
```

```
attribute = ldap_first_attribute(ld, entry, &ber);
while (attribute != NULL)
{
  j = strcmp(attribute,ATTR_NAME1);
  sprintf(output_line,"Attribute: %s",attribute);
  if (j == 0)
  {
    printf("%s \n",output_line);
  }
  else
  {
    j = 32-strlen(attribute);
    for (i = 0; i < j; i++)
    {
      filler[i] = 64;
    }
    filler[i] = 0;
    strcat(output_line,filler);
    strcat(output_line,VAL_HDR);
    prefix_len = strlen(output_line);
    values = ldap_get_values(ld, entry, attribute);
    for (i = 0; values[i] != NULL; i++)
    {
      if (i > 0  &&
          strcmp(attribute,"racfconnectgroupname") != 0)
      {
        strncat(output_line, " ", 1);
      }
      if (strcmp(attribute,"racfconnectgroupname") != 0)
      {
        strcat(output_line, values[i]);
      }
      else if (i == 0)
      {
        strcat(output_line, values[i]);
        printf("%s \n",output_line);
      }
      else
      {
        for (j = 0; j < prefix_len; j++)
        {
          output_line[j] = 64;
        }
        output_line[j] = 0;
        strcat(output_line, values[i]);
        printf("%s \n",output_line);
      }
    }
    if (strcmp(attribute,"racfconnectgroupname") != 0)
```

```
            {
              printf("%s \n",output_line);
            }
            ldap_value_free(values);
          }
          ldap_memfree(attribute);
          attribute = ldap_next_attribute(ld, entry, ber);
        }
      }
    }
  }
  NO_ATTR_LIST:

/*
 * Build the data structures required to effect the update to the
 * specified attribute and issue the ldap_modify_s() to attempt
 * the update.
 */

  strcpy(attr_name,arg_a);
  strcpy(attr_val,arg_v);
  vals[0] = (char *)&attr_val;
  vals[1] = NULL;
  mod.mod_op = LDAP_MOD_REPLACE;
  mod.mod_type = (char *)&attr_name;
  mod.mod_values = vals;

  mods[0] = &mod;
  mods[1] = NULL;

  j = ldap_modify_s(ld, binddn, mods);

#if defined(DEBUG)
  printf("ldap_modify_s() rc=%d\n",j);
#endif

  if (j != 0)
  {
    ldap_perror(ld, "ldap_modify_s() ");
    return(5);
  }

  ldap_unbind(ld);

  printf("Attribute update successful for %s\n",racfid);

  exit(0);
}
```

Rudy Douglas
System Programmer (Canada)

RACF in focus – RACF 'add-on' products

This is a regular column focusing on specific aspects of RACF. In this issue, we will discuss RACF 'add-on' products, and the pros and cons of acquiring them.

From a security administrator's viewpoint, RACF is not very user friendly. What is desired is not only simplified administration, but also simplified violation monitoring and reporting. This void has created a market for the so-called RACF 'add-on' products. These are products that add value to RACF by simplifying many of the daily RACF tasks, and by providing functions not found in native RACF.

By comparison, there are hardly any add-on products available in the market for the other two security software packages used on IBM mainframes, CA-ACF2 and CA-Top Secret. This is because those products are user friendly and there is no need for additional aids.

RACF add-on products are marketed not by IBM, but by third-party software vendors. Chief among these are Vanguard Security Professionals (with their Vanguard suite of products) and Consul/RACF.

PROS AND CONS

Not all installations need RACF add-on products. Smaller sites certainly can do without them. But larger shops find it easier to justify the expense.

As with anything in IT, there are advantages and disadvantages to using these additional products.

First, the pros:

1 They simplify daily RACF administration functions.

If you want to remove some of the drudgery from the daily

security administration then you need to look at these products. Not only will you be able to automate and simplify routine tasks, but you will also reduce errors such as typographical ones.

For instance, when you use these products, instead of using a RACF command to delete a userid, you list the userids, see on the screen the name associated with the one you want to delete, and enter **D** beside the line. This method is much less error-prone than the other method, in which it is easy to accidentally delete, for example, userid ABC1243 instead of userid ABC1234.

2 New RACF administrators do not need to acquire in-depth knowledge of native RACF commands.

Quite often non-mainframe staff are being cross-trained to perform RACF administration. These people naturally do not have an in-depth knowledge of mainframe command syntax. It is quite possible that they may not even have the desire to acquire this knowledge. Yet they need to get the job done. For them, having a user-friendly interface to RACF is welcome. In the case of deleting a userid mentioned above, they simply enter **D** beside the userid, without having to know the RACF command syntax.

3 The add-on products are supported by the vendors for new RACF releases, so you do not have to worry about these products not working in the future.

Constant changes are made to the operating system and to RACF. When you purchase add-on products, you are guaranteed that they will work now and in the future, with the vendor providing upgrades.

4 Add-on products are upgraded continuously with enhancements and improvements based on user requirements.

Your needs may change tomorrow. These products will

most likely meet those needs by providing additional functionality.

Now, the cons:

1 The products are usually expensive.

The cost of acquiring these products can be steep, depending on what options you select. The vendor, of course, will tell you that you will recover the expense in increased productivity, fewer errors, and other efficiency savings.

2 They are not customized to your standards. You can do very little to change these products to your liking; you have to change your procedures to fit with the way the products work.

Each installation has its own needs, and it is unlikely that your specific needs will be addressed by these off-the-shelf products.

3 When you use these products, in-house skills remain mediocre.

The old adage applies here: ‘Give me a fish and you feed me for a day; show me how to fish and you feed me forever’.

You will soon build a reliance on these products, and will be lost without them! Conversely, if you do not purchase them, you may begin thinking in more depth about your needs and develop some in-house tools of your own.

4 You may not need the add-ons if you already have some ‘home-grown’ software.

If, over the years, you have already built some tools, you may already have a customized set of add-on products, and there will be no need to throw them away and then acquire something costing a lot of money.

GENERAL FEATURES OF ADD-ON PRODUCTS

Most of these products are ISPF-menu driven. And most of them work with a copy of the RACF database, usually taken the previous day, to reduce the overhead on the live RACF database. For example, listing all users whose last name is SMITH is possible using these products, but can slow down RACF, and therefore the entire machine. But if you use an 'extract' of the previous day's database, this problem does not arise.

And most of these products are easy to install. While you still need a systems programmer to do the installation, because these products require MVS authorized libraries to be added to the system parameters, they are not complex to install.

RACF add-on products come in many flavours. Most of them fall in one of the following categories.

Security administration

Most add-on products focus on providing better RACF security administration. This activity is the most important reason for installations to justify acquiring add-on products.

Add-on products simplify the daily security administration functions such as, listing, adding, changing, or deleting RACF userids and RACF profiles. In addition, they provide more powerful capabilities than native RACF. For example you can list userids by last name, delete a group of userids, and make a change to a whole group of userids at the same time.

You can also 'clone' userids very easily by using these products; something very desirable, but not found in native RACF. Quite often a security request comes in that simply says, 'I have a new employee and I want them to have the same RACF accesses and capabilities as user John Smith'. Without an add-on product, you would have to produce listings and reports to find out all the accesses and privileges of John Smith and create the new userid using several RACF commands.

Security monitoring

Security monitoring add-on products provide ways to summarize and report on RACF violations and loggings. You can even alert management via e-mail when a particular violation occurs.

RACF violations that occur on the system usually go unnoticed, simply because there are so many of them are produced daily. You need either in-house developed programs or an add-on tool to cope with the monitoring of this important aspect of security.

Another important monitoring activity involves invalid password attempts. Add-on products provide a means to summarize and even trend invalid password attempts, so the security administrator is better able to determine whether these are routine events or something more serious.

The same goes for RACF 'warning' loggings, created as a result of a profile being in WARN mode. There is usually a reason why a profile is in WARN mode, and you want to be able to see at a glance this activity at your installation.

Yet another example of their use is in monitoring the activities of userids with special RACF powers. A summary of their activity is helpful, and usually easy to get from these products.

Password resets

Password resets are usually performed by Help Desk staff. The issue here is not the password resets itself but the security surrounding it – how does the person doing the password reset know the true identity of the person at the other end of the telephone line?

There are RACF add-on products that mitigate the risk somewhat, by providing the Help Desk staff with some means of verifying the authenticity of the person by asking for some personal information, such as mother's maiden name, social insurance number, or some other personal detail.

Other add-on products go a step further by allowing users to reset their own passwords via the Internet after they successfully verify their identity. Thus they automate this function in a risk-free manner, and remove the need for intervention by Help Desk staff.

Security reporting

Native RACF provides some basic reporting capabilities. Add-on products can provide a whole lot more.

For example, they provide these powerful search capabilities:

1 All dataset and resource profiles that are in WARN mode.

2 All profiles having a Universal Access of UPDATE (or READ or ALTER).

3 All profiles where a userid is the OWNER.

4 All accesses to all profiles by a userid.

Security compliance and enforcement

Add-on products that monitor and enforce operating system standards are not very well known, and there are only a few of them in the market.

These products not only monitor deviations from installation standards, they can also be used to enforce them by removing (or reversing) any changes that may occur.

Examples of entities that can be monitored and whose compliance can be enforced include: APF libraries, linklist libraries, SVCs, and started tasks.

IN CONCLUSION

If you are a small installation (less than 1,000 userids) you may not need a RACF add-on product, and you may not be able to justify acquiring one.

If you are a large installation with more than 1,000 userids then it becomes easier to justify the products. But then again, a business case should carefully consider two important factors: in-house RACF skills and existing home-grown tools.

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

Dinesh Dattani
Security Consultant
Toronto (Canada)

Maintaining good security

Over time, I have put together (from various sources) a few useful, and sometimes fairly obvious, suggestions for maintaining good security at the sites I visit. They are:

- Keep the number of people who are given the attribute system SPECIAL as small as possible. It's also a good idea to ensure that they (and everyone else) know that all their actions will be audited.
- Always allocate resource profiles to RACF groups rather than to users. This will make your life much easier at some time in the future when people change departments and ownership has to be reassigned.
- Always authorize resource profiles to RACF groups, not to users. This makes the job of access authority administration simpler and also means that the number of times a RACLISTed resource profile (eg IMS and CICS resources) has to be refreshed is much lower.
- Use group-SPECIAL and CLAUTH (USER) when you want authority over groups of resources (or even just one),

and you want this based on the scope of the group and protection objectives. You'll find that using the group-SPECIAL attribute makes it easier to reset the password for user-ids within the group (as opposed to using JOIN).

- Keep in mind that when you use group-SPECIAL to delegate power, it also applies to all the other groups below it in the hierarchy.
- When specifying authority over a single group of resources based on protection objectives, use JOIN and CLAUTH (USER).
- There will come a day when all your best plans go wrong. For these occasions, have emergency SPECIAL and OPERATIONS user-ids with passwords. These should be in a secure location and under management control.

Keep reminding users that passwords help them. It's what protects their data from being compromised. It is important that the passwords they choose aren't easy to guess (like their dog's name or their favourite team, etc). It's also important to maintain a password history to prevent passwords being repeated. Users with greater access to the system should change their passwords more frequently than ordinary users.

Where users are to be able to reset their passwords, use group-SPECIAL. You must then define all RACF commands to the PROGRAM class and restrict access to them, except for the ALTUSER command and its aliases for individuals granted the group-SPECIAL attribute.

Christopher Chapman
Security Consultant (USA)

August 2002–May 2005 index

Items below are references to articles that have appeared in *RACF Update* since issue 29, August 2002. 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.

Vanguard Integrity Professionals has received RSA Secured RSA SecurID Ready Certification from RSA Security for its Vanguard ez/Token product.

Vanguard ez/Token is a two-factor RSA SecurID authentication solution that allows users to authenticate through RSA SecurID tokens to the zSeries Server or any other application currently using RACF authentication. The Vanguard ezToken provides an additional level of security during the authentication process. With ez/Token, users substitute a new, one-time passcode in place of a password. Passcodes are generated randomly every 60 seconds. For enhanced security, the passcode can be combined with a PIN number.

For further information contact:
URL: www.go2vanguard.com/docs/marketing/press_releases/PR_2005_RSA-ezToken.pdf.

***

Counterpane Internet Security has announced Version 2.0 of Enterprise Protection Suite, its security services package that enhances protection from Web and e-mail-based attacks. New enhancements include expanded e-mail scanning services, and strengthened protection services to shield users from Distributed Denial of Service Attacks (DDoS) at the network layer.

The product consolidates traditionally separate service functions, such as security monitoring, carrier-level protection, and e-mail scanning, to provide a more unified view of disparate network activity. This provides a more robust defence against multiple points of network attacks, alleviates complexities in compliance reporting, and reduces administrative overhead for customers.

Other enhancements in the new version include enhanced managed security monitoring support for AS/400, AIX, and RACF devices with expanded security event coverage.

For further information contact:
URL: www.counterpane.com/pr-20050215a.html.

***

Sites running a mixture of RACF and other security products might be interested to know that Computer Associates has announced new versions of CA-ACF2 and CA-Top Secret.

Both products now include integration with the Security Management Architecture, as well as improved support for LDAP Directory Service, including recovery processing and the capability to send installation data.

For further information contact:
URL: www3.ca.com/Solutions/ProductFamily.asp?ID=111.

***