37

# RACF update

*August 2004*

## In this issue

# 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**

Nicole Thomas
E-mail: nicole@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.

## 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.

## Sensitive commands and the ISPF/PDF log datasets

ISPF/PDF can be used to create printable listings of partitioned or sequential datasets and it can also maintain a log of activities. These items are kept in files called the list dataset and the log dataset, respectively. When needed, the two datasets are allocated automatically. They are temporary files named <prefix>.<userid>.SPF<n>.LIST and <prefix>.<userid>.SPFLOG<n>.LIST, where

- <prefix> is the dataset prefix in the TSO user profile.
- <userid> is the user identification used to connect to TSO/E and kept in a RACF database or the UADS.
- <n> is a sequence number ranging from 1 to 9.

The ISPF/PDF system-wide exit number 16 allows an installation to change the <prefix> used. The exit routine can provide a prefix up to 26 characters long. This could be useful in a Parallel Sysplex environment with shared datasets and userids for different OS/390 images. ISPF/PDF reserves the remaining 18 characters of the dataset name for its own use – see *Interactive System Productivity Facility (ISPF) Planning and Customizing* (SC28-1298) for more details.

The list dataset is used for temporary storage for data to be printed at a later time. This data includes, for example, data written as a result of:

- Using the LIST service.
- Issuing the PRINT, PRINT-HI, PRINTL, or PRINTLHI commands (but not PRINTG).
- Using ISPF/PDF Option 3 utilities.

The log dataset is used to capture data that can be useful for diagnosing problems. Its contents include, for example, data captured as a result of:

- Using the LOG service.
- Test and trace data such as ISPF/PDF TRACE mode data.
- ISPF/PDF Dialog test Option 7.7: Dialog trace data.

A PASSWORD command issued from an ISPF panel (like the TSO/E CMD panel, ISPF/PDF Option 6) is clearly visible in the ISPF log, as can be seen below:

```
Time              *** ISPF transaction log ***                      Userid:

15:45  Start of ISPF Log - - - Session # 497 -------------------------
15:46  TSO - Command -  - ALU IBZZEX1 PASSWORD(TESTØØ)
15:54  TSO - Command -  - PASSWORD USER(IBZZEX1) PASSWORD(TESTØØ TESTØ1)
```

Often the ISPF log datasets are defined with a universal access of READ. This situation makes it easy for a person with bad intentions to find sensitive information that could easily be abused. This is especially true if there is some logic behind the password reset policy – like choosing a password of Xddmmyy (where *mmddyy* represents the date). One option is to educate the users to issue the PASSWORD command without a value for new password, in which case TSO/E (actually RACF) will prompt for it. Once TSO/E has taken control, ISPF/PDF no longer knows what happens, so it will not be logged. Another possibility is to set the dataset log option to DELETE LOG FILE WHEN LOGOFF in all ISPF/PDF sessions. This is, however, a customization that is, by default, left to the user unless ISPF/PDF is customized rather heavily. Furthermore, ISPF/PDF cannot guarantee a successful delete of the dataset in the case of an ABEND or a cancelled session. Finally, the log datasets could be protected by traditional RACF (or equivalent External Security Manager – ESM) means. This seems like the most logical thing to do and it won't hurt for the people who have the SPECIAL attribute. To set the universal (UACC) to NONE for all the ISPF log datasets for all users could create other problems. Debugging ISPF/PDF applications could become very troublesome. Because the developer in this situation has no access to the log datasets of the users, finding and resolving a problem could be unnecessarily complicated.

Our recommendation is two-fold. The first part is to remove a number of commands like PASSWORD/PW from the system altogether, an approach that we discussed in ‘Removing commands that have a security impact’, *RACF Update*, issue 2, November 1995, pp. 45–50. The second part is to keep the existing ISPF/PDF log dataset protection but to remove the logging of sensitive commands.

As can be seen from the example, all RACF commands are kept in the ISPF log files. In the example, a user’s profile is altered to set a new RACF password. In this case removing the ALTUSER/ALU RACF command is not a serious consideration. That’s why we propose to set the ISPF logging to OFF for all RACF commands. This can be done in the ISPTCM module (see SC28-1298). Bit X'10' of the flag byte tells ISPF/PDF not to write the command to the log dataset. Because all RACF commands are known only to the TSO authorized command facilities and not to ISPF/PDF, they should be copied from the TSO/E definitions in SYS1.PARMLIB(IKJTSO*xx*) or from the CSECT IKJTABLS (aliased by IKJEFTE2, IKJEFTE8, IKJEFTAP, and IKJEFTNS) to the ISPTCM module.

Because ISPF/PDF has the last word in the creation of the environment once a command is defined in its own table, care should be taken with the other settings too. If the aim is solely to turn off the ISPF/PDF logging, it would seem sufficient to define the RACF commands with an X'10' flag (see list below). This would lead, however, to X'047' abends in the RACF environment module (IRRENV00). Abend X'047' indicates that an unauthorized program issued a restricted Supervisor Call (SVC) instruction. In general an authorized program runs in supervisor state, holds PSW key 0-7, or resides in an APF-authorized library. Since we didn’t change the library, it is clear that ISPF intervenes and runs the command in an unauthorized (program problem state and user key) status once it is defined in the ISPTCM table.

The solution is to gather all TSO/E authorized RACF commands and to repeat the definitions in ISPTCM with a flag byte of

X'32'. This effectively turns off the ISPF logging without any other unwanted side-effects. The command is invoked as an authorized command (the X'20' bit) and ISPF/PDF knows that it is a program and not a CLIST or REXX EXEC (the X'02' flag). Other sensitive TSO/E authorized commands are not harmful, otherwise we could include them too. The TSO/E ACCOUNT command for instance is logged only when invoked. As with all TSO/E command processing under ISPF, subcommands are a matter for TSO/E to deal with. ISPF is involved only in setting up the requested environment.

The values of the ISPF flag byte for the current entry are shown below. The default is 02. The flag field is shown followed by its description:

- B'1.......' – reserved.
- B'.1......' – command requires a function pool. Set this bit on for a command processor program that issues dialog services.
- B'..1.....' – command requires an authorization check. Set this bit on for a command processor that must be invoked as an authorized command.
- B'...1....' – command is not to be logged. Set this bit on if the TSO command buffer should not be written to the ISPLOG dataset.
- B'....1...' – command is not supported by ISPF. Set this bit on for commands that cannot be invoked under ISPF.
- B'.....1..' – command is a command procedure (CLIST). Set this bit on if this is the name of a CLIST member.
- B'......1.' – command is a command processor. Set this bit on if this is the name of a command processor program module.
- B'.......1' – command requires a BLDL to be issued. Set this bit on if a BLDL is to be issued to determine whether this is a command processor module or a CLIST.

Since ISPTCM is known to SMP/E, changes should be applied with a user modification. The original ISPTCM Assembler source as delivered by IBM can be found in ISP.SISPSAMP(ISPTCMA). The member must be renamed to ISPTCM before assembly. In the example SMP/E job below, we assume that the site-specific ISPTCM is inserted. The added RACF commands can appear anywhere in between the ISPTCM HEADER and the ISPTCM END macro invocations. In the sample job, site-specific items include the JOBCARD, the UNIT NAME, the names of the system datasets, the name of the USERMOD, the FMID, the DISTLIB, the name of the CSI, and the ISPF/PDF target zone.

Notes:

1 Entries in ISPTCM do not have to be in alphabetical order but they are placed in this order in the load module. Do not be surprised if you check the result afterwards in the ISP.SISPLPA library and you cannot find your changes in the same place that you put them. This happened to us when we included all RACF commands at the start of the Assembler program and found them alphabetically ordered all over the load module.

2 The last step in the job includes a linkage-editor (or DFSMS Binder) statement to add an IDENTIFY record to the load module. IBM uses this method often when a PTF is applied and it seems a good habit to include an indicator of the SMP/E action that last changed the module at the start of it.

Start of job:

```
//IBZZEX4X JOB (JAN),'JAN DE DECKER',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,
//             REGION=ØM,COND=(Ø,NE)
//*
//* ISPF/PDF TSO/E AUTHORIZED COMMAND TABLE
//*
//SØ       EXEC PGM=ASMA9Ø,REGION=ØM,
//             PARM='DECK,NOOBJECT'
//SYSLIB    DD  DSN=SYS1.MACLIB,DISP=SHR
//          DD  DSN=SYS1.MODGEN,DISP=SHR
//          DD  DSN=ISP.SISPMACS,DISP=SHR
```

```
//SYSUT1   DD  UNIT=VIO,SPACE=(17ØØ,(6ØØ,1ØØ))
//SYSUT2   DD  UNIT=VIO,SPACE=(17ØØ,(6ØØ,1ØØ))
//SYSUT3   DD  UNIT=VIO,SPACE=(17ØØ,(6ØØ,1ØØ))
//SYSPRINT DD  SYSOUT=*
//SYSPUNCH DD  DSN=&&LOADSET,
//             DISP=(,PASS),
//             SPACE=(CYL,(5,1)),UNIT=SYSALLDA
//SYSIN    DD  *
         PUNCH '++ USERMOD (IUMODØ5).'
         PUNCH '++ VER (ZØ38) FMID(HIF44Ø2).'
         PUNCH '++ MOD (ISPTCM) DISTLIB(AISPMOD1) LMOD(ISPTCM).'
```

Insert your old ISPTCM header here.

```
*
* ALL RACF COMMANDS FROM IKJTSOØØ
*
        ISPMTCM  FLAG=32,ENTNAME=AD       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ADDGROUP NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ADDSD    NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ADDUSER  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=AG       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ALD      NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ALG      NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ALTDSD   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ALTGROUP NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ALTUSER  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=ALU      NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=AU       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=BLKUPD   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=CO       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=CONNECT  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=DD       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=DELDSD   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=DELGROUP NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=DELUSER  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=DG       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=DU       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=IRRDPIØØ NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=LD       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=LG       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=LISTDSD  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=LISTGRP  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=LISTUSER NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=LU       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=PASSWORD NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=PE       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=PERMIT   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=PW       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RALT     NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RALTER   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RDEF     NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RDEFINE  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RDEL     NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RDELETE  NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RE       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=REMOVE   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RL       NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RLIST    NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=RVARY    NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=SEARCH   NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=SETR     NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=SETROPTS NOLOG, TSO CMD, CMD PROCESSOR
        ISPMTCM  FLAG=32,ENTNAME=SR       NOLOG, TSO CMD, CMD PROCESSOR
*
* END OF ADDED RACF COMMANDS
*
```

Insert your old commands and ISPTCM END statement here.

```
//S1       EXEC PGM=GIMSMP
//SMPCSI   DD  DSN=SPB1.MVS.V24Ø.GLOBAL.CSI,DISP=SHR
//SMPHOLD  DD  DUMMY
//SMPCNTL  DD  *
  SET BDY(MVST1ØØ).
     RESTORE S(IUMODØ5)         /* REMOVE USERMOD IF APPLIED  */ .
  RESETRC                       /* RESET RETURN CODE          */ .
  SET BDY(GLOBAL) .
    REJECT  S(IUMODØ5) BYPASS(APPLYCHECK) .
  RESETRC                       /* RESET RETURN CODE          */ .
  SET BDY(GLOBAL) .
    RECEIVE S(IUMODØ5)          /* RECEIVE USERMOD            */ .
  SET BDY(MVST1ØØ) .
    APPLY S(IUMODØ5)            /* APPLY USERMOD              */ .
    LIST  SYSMOD(IUMODØ5)       /* LIST USERMOD               */ .
/*
//SMPPTFIN DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//         DD *
 IDENTIFY ISPTCM('IUMODØ5')
/*
```

*Jan De Decker*
*Senior Systems Engineer JED:SP NV (Belgium)*

# RACF in focus – understanding WebSphere MQ security

*This is a regular column focusing on specific aspects of RACF. In this issue, we will discuss security issues related to the IBM middleware known as WebSphere MQ, and see how to implement security for it.*

## BACKGROUND

WebSphere MQ is what was previously known as MQSeries. This is the middleware that can run on multiple platforms, and its aim is to provide user-friendly ways for applications on different platforms to communicate with each other.

It is important to understand that, since the MQ software can reside on different platforms, different security mechanisms are involved.

For IBM mainframe (OS/390 and z/OS) platforms, security is implemented via SAF calls to RACF. And this is what we will be discussing here. For distributed platforms, MQ security is implemented via 'Callable Services' and 'Object Authority Manager'. We shall not discuss that here. The mainframe MQ security is more robust and granular than the MQ security found on other platforms.

Even though we will not discuss MQ security on other platforms, protecting WebSphere MQ resources on the mainframe poses some peculiar challenges, since the application connections can be from any other platform. Also, there are some things peculiar to the security aspects of MQ itself. For example, there is something called 'switch profiles', which are unique to this product. The concept of switch profiles is not found anywhere else in RACF.

## PROTECTING MQ RESOURCES

WebSphere MQ security is achieved in RACF by activating classes for MQ. These are MQADMIN, MQCONN, MQQUEUE, MQCMDS, MQPROC, and MQNLIST. You need to activate these classes and build appropriate profiles.

The MQ profiles are similar to DB2 profiles in one respect – they are prefixed by the subsystem name, followed by other relevant information about the profile. To simplify, we will use MQPP as the subsystem id in all the examples below.

The good thing about MQ security is that it can be implemented gradually – you do not have to protect all MQ resources at once. A phased approach is desirable because it allows you to minimize the impact on the user community, and also to learn from your mistakes.

## MQADMIN CLASS

We will first consider the MQADMIN class because it is the one that allows you to gradually implement MQ security.

Profiles in this class define administrative controls, alternative user profiles, context control profiles, resource-level control profiles, and switch profiles.

Among the administrative controls at your disposal is the ability to switch off some of the security checking for MQ. And this is where switch profiles come in.

For example, if you have a profile:

```
MQPP.NO.COMMAND.CHECKS
```

then security for MQ commands will not be checked by RACF. The mere presence of this profile in the MQADMIN class is the 'switch' that turns command-level security OFF, hence the term 'switch profiles'. All profiles in the MQCMDS class described below will be ignored. To start protecting MQ commands, you need to remove this profile.

Similarly you can define switch profiles for the following, to bypass MQ security:

- MQPP.NO.QUEUE.CHECKS – bypasses profiles in MQQUEUE class.

- MQPP.NO.CONNECT.CHECKS – bypasses profiles in MQCONN class.
- MQPP.NO.ALTERNATE.USER.CHECKS – bypasses alternate user checking.
- MQPP.NO.NLIST.CHECKS – bypasses profiles in MQNLIST class.
- MQPP.NO.PROCESS.CHECKS – bypasses profiles in MQPROC class.

There is even the ability to switch off the security for an entire subsystem, for all classes. For example, the profile MQPP.NO.SUBSYS.SECURITY will turn off security checking for the entire MQPP subsystem!

It follows that if it is your job to ensure that full MQ security has been implemented at your site, one of the things you have to do is to verify that there are no SUBSYS.NO profiles in the MQADMIN class. If there are no switch profiles, by default, MQ security checking will be done if the class is active.

By the way, there are no userids or groups in the access lists of switch profiles. Neither is UACC used.

MQ allows you to specify an alternative userid to do your work, if you are so authorized. The concept is similar to that of the SURROGAT class.

For example, the profile MQPP.USER1.ALTUSER specifies that userid USER1 can do work under the alternative userid ALTUSER.

## MQCONN CLASS

The MQCONN class contains MQ connection profiles. These profiles define who can connect to MQ, and by what means.

Here are some examples:

- MQPP.BATCH specifies who can access MQ applications from batch and TSO.

- MQPP.CICS specifies who can access MQ applications from CICS transactions.
- MQPP.IMS specifies who can access MQ applications from IMS transactions.
- MQPP.CHIN specifies who can access MQ applications via channel initiator programs.

In all cases, users need READ access to these profiles.

And remember, you want to make sure you do not have the profile MQPP.NO.CONNECT.CHECKS in the MQADMIN class, otherwise security checking for the MQCONN class will be bypassed.

## MQQUEUE CLASS

Profiles in the MQQUEUE class determine who can use the MQ application queue resources. This is where you will have to interact with your MQ people and determine what queue names are being used at your installation.

Example:

```
MQPP.QUEUE.NAME
```

The level of access required in the access lists of these profiles depends on the type of operation required:

- Inquire and browse type functions require READ access.
- Input, output, and bind require UPDATE access.
- SET requires ALTER access.

If you want to protect MQ's queue resources, make sure the profile MQPP.NO.QUEUE.CHECKS in the MQADMIN class is absent, otherwise security checking for the MQQUEUE class will be bypassed.

## MQCMDS CLASS

Profiles in the MQCMDS class protect MQ commands.

For example, the profile MQPP.ARCHIVE.* specifies who can use the MQ ARCHIVE command.

Display commands require READ access. Commands such as ARCHIVE, PING, RECOVER, RESET, etc require CONTROL access. The commands DEFINE, ALTER, DELETE, etc require ALTER access.

If a command affects a resource, then access to that resource must also be allowed for the command to execute successfully.

## MQNLIST

Profiles in the MQNLIST class protect Namelist resources.

## MQPROC

Profiles in the MQPROC class control MQ Process resources.

## SUMMARY

We have seen how basic MQ security can be implemented in a phased manner. But please remember, you cannot do it alone. You need to involve your MQ support people, and, in some cases, the CICS people.

*Dinesh Dattani (dinesh123@rogers.com)*
*Independent Consultant*
*Toronto (Canada)*

# Differentiating RACF REVOKED userids with ICHRIX01 and ICHRIX02

Information available in the RACF database does not always make it clear why a RACF userid has gone into REVOKE status. The most likely reasons for a userid becoming REVOKEd are:

1 An administrator purposely set the userid into REVOKE status.

2 The real owner of the userid entered too many consecutive incorrect password values while attempting to log on to the system.

It would be nice to be able to distinguish between these two different causes.

This article discusses using the ICHRIX01/ICHRIX02 RACF exit pair to capture a userid state change (ie a userid going from nonREVOKEd to REVOKEd status while attempting to log on) and reflect that REVOKE reason into the RACF database. If ICHRIX02 determines that a userid has gone into REVOKE status (when the userid hadn't previously been in REVOKE status in the ICHRIX01 check), ICHRIX02 will set the X'04' bit in the FLAG4 field of the user base segment record for the userid in question.

## FUNCTIONAL OVERVIEW

ICHRIX01 and ICHRIX02 are the RACF VERIFY(X) pre- and post-exits respectively. They will be invoked whenever a RACROUTE VERIFY(X) operation is requested (eg TSO log-on, CICS log-on). The ICHRIX01 exit captures and maintains the REVOKE status for the userid in question. ICHRIX02 examines this saved status and compares it with the current status for the userid. If the userid has gone into REVOKE status (almost assuredly the result of too many invalid password attempts), ICHRIX02 will set the X'04' bit in the FLAG4 field of the base segment record for this userid (leaving the already set X'80' bit intact). The use of the additional bit setting in the FLAG4 field does not compromise the normal use of this field by RACF and when a subsequent ALTUSER command is used to RESUME a userid that has had its FLAG4 field modified as described here, the flag is properly reset to X'00'.

## USING THE FLAG4 INFORMATION

A post-processing program, RVKLST, can be used to display the current revoke status of either all userids defined to RACF or one specific userid as provided in a parameter to the RVKLST utility program. For example, the following JCL can be used to produce an output listing providing the revoke status of every userid defined to a RACF database:

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

Running the above job will produce sample output similar to the following:

```
Userid    Revoke Status
--------  -------------------------
ABARS     Not REVOKED
APPC      Not REVOKED
ASCH      REVOKED
BLSJPRMI  REVOKED
BPXOINIT  Not REVOKED
BPXROOT   Not REVOKED
USER1     Administrator REVOKED
USER2     Not REVOKED
USER3     Not REVOKED
USER4     Date REVOKED
USER5     Not REVOKED
USER6     Password violation REVOKED
```

The above is merely a representative example of the output. On even a modestly-sized RACF database, there would be many more userids than the sample output provides.

You can also use the RVKLST utility to display the REVOKE status of a single RACF userid. Use an EXEC statement PARM specification to display the REVOKE status of a single userid. For example, to display the REVOKE status for userid USER12, use the following job:

```
//RVKLST   EXEC PGM=RVKLST,PARM='USER12'
//STEPLIB  DD   DSN=auth.load.library,DISP=SHR
//SYSPRINT DD   SYSOUT=*
```

Running the RVKLST utility against a RACF database that has the specified userid REVOKEd because of too many password violations, and this article’s ICHRIX01/ICHRIX02 exits active, will show output similar to the following:

```
Userid   Revoke Status
-------- -------------------------
USER12   Password violation REVOKED
```

## PROGRAM LINKAGE AND EXIT MANAGEMENT

RVKLST, ICHRIX01, and ICHRIX02 require standard assembly with SYSLIB specifying SYS1.MACLIB and SYS1.MODGEN. The RVKLST program will need to be link-edited into an authorized load library using a linkedit job similar to the following:

```
//IEWL      EXEC   PGM=HEWLHØ96,PARM='XREF,LIST,MAP'
//SYSPRINT DD     SYSOUT=*
//SYSUT1   DD     UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT   DD     DSN=object.code.pds,DISP=SHR
//SYSLMOD  DD     DSN=auth.load.library,DISP=SHR
//SYSLIN   DD     *
   INCLUDE OBJECT(RVKLST)
   ENTRY   RVKLST
   SETCODE AC(1)
   NAME    RVKLST(R)
```

The ICHRIX01 and ICHRIX02 exits are recognized and activated by RACF at system IPL. Unless you have access to a dynamic RACF exit loader (see 'Dynamic RACF exits', *RACF Update*, issue 33, August 2003) you will need to link-edit ICHRIX01 and ICHRIX02 into a load library that is contained in your LPALST concatenation. Use a link-edit job similar to the following:

```
//IEWL      EXEC   PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'
//SYSPRINT DD     SYSOUT=*
//SYSUT1   DD     UNIT=SYSDA,SPACE=(CYL,(2,1))
//OBJECT   DD     DSN=object.code.pds,DISP=SHR
//SYSLMOD  DD     DSN=lpalst.load.library,DISP=SHR
//SYSLIN   DD     *
   INCLUDE OBJECT(ICHRIXØ1)
   ENTRY   ICHRIXØ1
   SETCODE AC(1)
   NAME    ICHRIXØ1(R)
   INCLUDE OBJECT(ICHRIXØ2)
   ENTRY   ICHRIXØ2
   SETCODE AC(1)
   NAME    ICHRIXØ2(R)
```

After the exits have been link-edited into an LPA dataset, IPL the system. RACF should automatically recognize their existence.

## CONCLUSION

Being able to determine the reason for a userid REVOKE condition can be important to a RACF administrator who is charged with the responsibility to RESUME a REVOKEd userid. Using the information captured and saved by ICHRIX01 and ICHRIX02 as discussed in this article can provide a useful extension to RACF administration. If you add this to the ability to capture RACF administrative REVOKE conditions (see article 'Using IRREVX01 and the RACF database to help differentiate REVOKED userids' in *RACF Update*, issue 35, February 2004), your RACF administrators will have even more information available to them when they are requested to RESUME a userid.

I'm confident that this information will be of valuable use to your RACF administrators.

## ICHRIX01 ASSEMBLER

```
*-------------------------------------------------------------------*
*                                                                   *
*   ICHRIXØ1 is the RACF VERIFY(X) pre exit that is invoked by RACF *
*   prior to userid verification.                                   *
*                                                                   *
*   This ICHRIXØ1 exit is designed to capture and save the current  *
*   REVOKE status (BASE segment FLAG4) for a user going through     *
*   RACROUTE VERIFY(X) processing.  Specifically, this exit captures *
*   this information so that it can be used by the ICHRIXØ1         *
*   companion exit, ICHRIXØ2 (VERIFY(X) post exit), to determine    *
*   whether a userid has gone into REVOKE status between the VERIFY *
*   pre exit and the VERIFY post exit.                              *
*                                                                   *
*   ICHRIXØ1 must be reentrant and is entered in supervisor state,  *
*   key Ø so be careful.                                            *
*                                                                   *
*   The following JCL provides a sample job to linkedit the         *
*   ICHRIXØ1 exit:                                                  *
*                                                                      *
*   //IEWL      EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'           *
*   //SYSPRINT DD    SYSOUT=*                                          *
*   //SYSUT1   DD    UNIT=SYSDA,SPACE=(CYL,(2,1))                      *
*   //OBJECT   DD    DSN=object.code.pds,DISP=SHR                      *
*   //SYSLMOD  DD    DSN=lpalst.load.library,DISP=SHR                  *
*   //SYSLIN   DD    *                                                 *
*      INCLUDE OBJECT(ICHRIXØ1)                                        *
*      ENTRY   ICHRIXØ1                                                *
*      SETCODE AC(1)                                                   *
*      NAME    ICHRIXØ1(R)                                             *
*                                                                      *
*   Under normal RACF operation, the existence of the ICHRIXØ1 exit   *
*   is determined at system IPL time with module ICHRIXØ1 residing    *
*   somewhere in the LPALSTxx concatenation of datasets.  RACF does   *
*   not honour this exit via dynamic LPA activation.  In the absence  *
*   of a dynamic RACF exit loader, it will be necessary to IPL your   *
*   system to activate this exit.                                      *
*                                                                      *
*   This ICHRIXØ1 exit dynamically acquires storage that is released  *
*   by the companion ICHRIXØ2 exit.                                    *
*                                                                      *
*----------------------------------------------------------------------*
ICHRIXØ1 CSECT
ICHRIXØ1 AMODE 31
ICHRIXØ1 RMODE ANY
         STM   R14,R12,12(R13)       Save incoming registers
         LR    R12,R15               Copy module base address
         USING ICHRIXØ1,R12          Set addressability
         LR    R3,R13                Copy savearea address
         LR    R2,R1                 Save parm address
         STORAGE OBTAIN,LENGTH=WORKLEN,LOC=ANY
         LR    RØ,R1                 Copy storage address
         LR    R14,R1                Again
         LR    R13,R1                And again
         L     R1,=A(WORKLEN)        Get length
         XR    R15,R15               Clear the fill byte
         MVCL  RØ,R14                Clear the storage
         USING WORKAREA,R13          Set addressability
         ST    R3,SAVEAREA+4         Save savearea address
*---------------------------------------------------------------------*
         USING RIXPL,R2              Set parameter addressability
*---------------------------------------------------------------------*
         L     R6,RIXFLAG            Get flag address
         TM    Ø(R6),RIXENVCH+RIXENVDE Change or delete flag set?
         BNZ   RETURN                Yes - return
         TM    Ø(R6),RIXPSCKN        PASSCHK=NO?
         BO    RETURN                Yes - return
*---------------------------------------------------------------------*
```

```
         XC    RACWORK(256),RACWORK  Clear the RACROUTE ...
         XC    RACWORK+256(256),RACWORK+256 work area
         MVC   ROUTWRK1(ROUTLEN1),RACROUT1 Copy the RACROUTE model
         L     R6,RIXUID             Get userid area address
         LA    R6,1(,R6)             Skip past length
         RACROUTE REQUEST=EXTRACT,                                    x
               TYPE=EXTRACT,                                          x
               ENTITY=(R6),                                           x
               RELEASE=1.9.2,                                         x
               FIELDS=FLDLIST1,                                       x
               SUBPOOL=1,                                             x
               WORKA=RACWORK,MF=(E,ROUTWRK1) Extract some userid info
         LTR   R15,R15               Extract ok?
         BNZ   RETURN                No - we're done
         LR    R6,R1                 Copy the extract area address
*-------------------------------------------------------------------*
*   Save the relevant information from the EXTRACT workarea.         *
*--------------------------------------------------------------------*
         XR    R15,R15               Clear R15
FLD1     DS    ØH
         ICM   R15,B'ØØ11',4(R6)     Get data offset
         LA    R6,Ø(R15,R6)          Point to FLAG4 data area
         MVC   FLAG4SAV(1),4(R6)     Save the flag data
FLD2     DS    ØH
         ICM   R15,B'1111',Ø(R6)     Get field length
         LTR   R15,R15               Any data?
         BZ    FLD3                  No - check resume date
         LA    R6,4(R15,R6)          Point to REVOKEDT data area
         MVC   RVKDTSAV(3),4(R6)     Save the revoke date data
FLD3     DS    ØH
         ICM   R15,B'1111',Ø(R6)     Get field length
         LTR   R15,R15               Any data?
         BZ    FLD4                  No - do next field if any
         LA    R6,4(R15,R6)          Point to RESUMEDT data area
         MVC   RSMDTSAV(3),4(R6)     Save the resume date data
FLD4     DS    ØH
         XR    R8,R8                 Clear R8
         XR    R9,R9                 Clear R9
         IC    R9,Ø(,R1)             Save the subpool value
         ICM   R8,B'Ø111',1(R1)      Save w/a length
         STORAGE RELEASE,LENGTH=(R8),ADDR=(R1),SP=(R9)
*-------------------------------------------------------------------*
         STORAGE OBTAIN,LENGTH=RIXWLEN,LOC=ANY
         XC    Ø(16,R1),Ø(R1)        Clear the storage
         L     R3,RIXWA              Get address of work word
         ST    R1,Ø(,R3)             Save the work address
         LR    R5,R1                 Copy the storage address
         USING RIXWAREA,R5           Set addressability
         MVC   RIXWID(4),=C'RIXW'    Move in the eyecatcher
         L     R6,RIXUID             Get userid area address
         LA    R6,1(,R6)             Skip past length
         MVC   RIXWUID(8),Ø(R6)      Save the userid
         MVC   RIXWFLG1(1),FLAG4SAV  Copy FLAG4
         MVC   RIXWRVDT(3),RVKDTSAV  Copy revoke date
         MVC   RIXWRSDT(3),RSMDTSAV  Copy resume date
*-------------------------------------------------------------------*
         DROP  R5
*-------------------------------------------------------------------*
RETURN   DS    ØH
         L     R3,SAVEAREA+4         Copy old savearea address
         LR    R1,R13                Get temp storage address
         STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
         LR    R13,R3                Copy the old savearea address
         LM    R14,R12,12(R13)       Restore the registers
         XR    R15,R15               RC=Ø
         BR    R14                   Return
*--------------------------------------------------------------------*
FLDLIST1 DC    F'3'
         DC    CL8'FLAG4   '
         DC    CL8'REVOKEDT'
         DC    CL8'RESUMEDT'
*-------------------------------------------------------------------*
RACROUT1 RACROUTE REQUEST=EXTRACT,                                    x
               TYPE=EXTRACT,                                          x
               CLASS='USER',                                          x
               RELEASE=1.9.2,                                         x
               MF=L
ROUTLEN1 EQU   *-RACROUT1
*-------------------------------------------------------------------*
WORKAREA DSECT
SAVEAREA DS    18F
FLAG4SAV DS    XL1
RVKDTSAV DS    XL3
RSMDTSAV DS    XL3
CURRDATE DS    XL3
RETCODE  DS    F
ROUTWRK1 DS    ØD,CL(ROUTLEN1)
RACWORK  DS    ØD,CL(512)
WORKLEN  EQU   *-WORKAREA
*-------------------------------------------------------------------*
RIXWAREA DSECT
RIXWID   DS    CL4
RIXWUID  DS    CL8
RIXWFLGS DS    ØF
RIXWFLG1 DS    XL1
RIXWFLG2 DS    XL1
RIXWFLG3 DS    XL1
RIXWFLG4 DS    XL1
```

```
RIXWRVDT DS    XL3
RIXWRSDT DS    XL3
RIXWLEN  EQU   *-RIXWAREA
*---------------------------------------------------------------------*
         ICHRIXP ,
*---------------------------------------------------------------------*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*---------------------------------------------------------------------*
         END
```

## ICHRIX02 ASSEMBLER

```
*----------------------------------------------------------------------*
*                                                                      *
*   ICHRIXØ2 is the RACF VERIFY(X) post exit that is invoked by RACF   *
*   after userid verification.                                         *
*                                                                      *
*   This ICHRIXØ2 exit is designed to use information captured in the *
*   ICHRIXØ1 (VERIFY(X) pre exit) to determine whether a userid has    *
*   gone into REVOKE status between the VERIFY pre exit and the        *
*   VERIFY post exit.  If that is the case, ICHRIXØ2 sets the x'Ø4'    *
*   bit on in the FLAG4 of the BASE segment record for the             *
*   corresponding userid.  This indicates that the userid has gone     *
*   into REVOKE status because of too many invalid password attempts. *
*                                                                      *
*   ICHRIXØ2 must be reentrant and is entered in supervisor state,     *
*   key Ø so be careful.                                               *
*                                                                      *
*   The following JCL provides a sample job to linkedit the            *
*   ICHRIXØ2 exit:                                                     *
*                                                                      *
*   //IEWL      EXEC  PGM=HEWLHØ96,PARM='XREF,LIST,MAP,RENT'           *
*   //SYSPRINT DD    SYSOUT=*                                          *
*   //SYSUT1   DD    UNIT=SYSDA,SPACE=(CYL,(2,1))                      *
*   //OBJECT   DD    DSN=object.code.pds,DISP=SHR                      *
*   //SYSLMOD  DD    DSN=lpalst.load.library,DISP=SHR                  *
*   //SYSLIN   DD    *                                                 *
*      INCLUDE OBJECT(ICHRIXØ2)                                        *
*      ENTRY   ICHRIXØ2                                                *
*      SETCODE AC(1)                                                   *
*      NAME    ICHRIXØ2(R)                                             *
*                                                                      *
*   Under normal RACF operation, the existence of the ICHRIXØ2 exit   *
*   is determined at system IPL time with module ICHRIXØ2 residing    *
*   somewhere in the LPALSTxx concatenation of datasets.  RACF does   *
*   not honour this exit via dynamic LPA activation.  In the absence  *
*   of a dynamic RACF exit loader, it will be necessary to IPL your   *
*   system to activate this exit.                                      *
*                                                                      *
*   This ICHRIXØ2 exit releases storage that has been previously      *
*   acquired by the companion ICHRIXØ1 exit.                           *
*                                                                      *
*---------------------------------------------------------------------*
ICHRIXØ2 CSECT
ICHRIXØ2 AMODE 31
ICHRIXØ2 RMODE ANY
         STM   R14,R12,12(R13)
         LR    R12,R15                Copy module base address
         USING ICHRIXØ2,R12           Set addressability
         LR    R3,R13                 Copy savearea address
         LR    R2,R1                  Save parm address
         STORAGE OBTAIN,LENGTH=WORKLEN,LOC=ANY
         LR    RØ,R1                  Copy storage address
         LR    R14,R1                 Again
         LR    R13,R1                 And again
         L     R1,=A(WORKLEN)         Get length
         XR    R15,R15                Clear the fill byte
         MVCL  RØ,R14                 Clear the storage
         USING WORKAREA,R13
         ST    R3,SAVEAREA+4          Save savearea address
*----------------------------------------------------------------------*
         USING RIXPL,R2               Set parameter addressability
*---------------------------------------------------------------------*
         L     R6,RIXFLAG             Get flag address
         TM    Ø(R6),RIXENVCH+RIXENVDE Change or delete flag set?
         BNZ   RETURN                 Yes - return
         TM    Ø(R6),RIXPSCKN         PASSCHK=NO?
         BO    RETURN                 Yes - return
*---------------------------------------------------------------------*
         XC    RACWORK(256),RACWORK  Clear the RACROUTE ...
         XC    RACWORK+256(256),RACWORK+256 work area
         MVC   ROUTWRK1(ROUTLEN1),RACROUT1 Copy the RACROUTE model
         L     R6,RIXUID              Get userid area address
         LA    R6,1(,R6)              Skip past length
         RACROUTE REQUEST=EXTRACT,                                     x
               TYPE=EXTRACT,                                           x
               ENTITY=(R6),                                            x
               RELEASE=1.9.2,                                          x
               FIELDS=FLDLIST1,                                        x
               SUBPOOL=1,                                              x
               WORKA=RACWORK,MF=(E,ROUTWRK1) Extract some userid info
         LTR   R15,R15                Extract ok?
         BNZ   RETURN                 No - we're done
         LR    R6,R1                  Copy the extract area address
*-------------------------------------------------------------------*
         XR    R15,R15                Clear R15
FLD1     DS    ØH
         ICM   R15,B'ØØ11',4(R6)      Get data offset
         LA    R6,Ø(R15,R6)           Point to flag4 data area
         MVC   FLAG4SAV(1),4(R6)      Save the flag data
FLD2     DS    ØH
         ICM   R15,B'1111',Ø(R6)      Get field length
         LTR   R15,R15                Any data?
         BZ    FLD3                   No - check resume date
         LA    R6,4(R15,R6)           Point to revokedt data area
         MVC   RVKDTSAV(3),4(R6)      Save the revoke date data
FLD3     DS    ØH
         ICM   R15,B'1111',Ø(R6)      Get field length
         LTR   R15,R15                Any data?
         BZ    FLD4                   No - do next field if any
         LA    R6,4(R15,R6)           Point to resumedt data area
         MVC   RSMDTSAV(3),4(R6)      Save the resume date data
FLD4     DS    ØH
         XR    R8,R8                  Clear R8
         XR    R9,R9                  Clear R9
         IC    R9,Ø(,R1)              Save the subpool value
         ICM   R8,B'Ø111',1(R1)       Save w/a length
         STORAGE RELEASE,LENGTH=(R8),ADDR=(R1),SP=(R9)
*------------------------------------------------------------------*
         L     R3,RIXWA               Get address of work word
         L     R5,Ø(,R3)              Get ICHRIXØ1 work area address
         LTR   R5,R5                  A work area?
         BZ    RETURN                 No - go home
         USING RIXWAREA,R5            Set addressability
         L     R6,RIXUID              Get userid area address
         LA    R6,1(,R6)              Skip past length
         CLC   RIXWUID(8),Ø(R6)       Same userid?
         BNE   RETURN                 No - go home
         CLC   RIXWFLG1(1),FLAG4SAV   Status the same?
         BE    RETURN                 Yes - go home
*------------------------------------------------------------------*
```

```
         TM    FLAG4SAV,X'80'         Revoked?
         BNO   RETURN                 No - go home
         OI    FLAG4SAV,X'04'         Set password violation revoke flag
         MVC   FLAG4LEN(4),=F'1'      Set length
         XC    RACWORK(256),RACWORK   Clear the RACROUTE ...
         XC    RACWORK+256(256),RACWORK+256 work area
         MVC   ROUTWRK1(ROUTLEN1),RACROUT1 Copy the RACROUTE model
         L     R6,RIXUID              Get userid area address
         LA    R6,1(,R6)              Skip past length
         RACROUTE REQUEST=EXTRACT,                                      x
               TYPE=REPLACE,                                            x
               ENTITY=(R6),                                             x
               RELEASE=1.9.2,                                           x
               FIELDS=FLDLIST2,                                         x
               SEGDATA=FLAG4LEN,                                        x
               WORKA=RACWORK,MF=(E,ROUTWRK1) Update FLAG4
         LTR   R15,R15                Replace ok?
         BNZ   UPDTERR                No - indicate update error
*-------------------------------------------------------------------*
*                                                                   *
*   Enable the following commented code lines (the lines prefixed   *
*   with '*==>') if you want to issue a WTO console message when    *
*   the FLAG4 field has been successfully updated to indicate a     *
*   password violation revoke condition.                            *
*                                                                   *
*-------------------------------------------------------------------*
*==>     MVC   WTOWRK(WTOLN),WTOLST   Copy WTO model
*==>     MVC   WTOWRK+4(L'WTOMSG1),WTOMSG1 Copy message model
*==>     MVC   WTOWRK+4+54(8),0(R6)   Copy the userid
*==>     WTO   MF=(E,WTOWRK)          Issue the WTO
         B     RETURN                 Return
UPDTERR  DS    0H
*------------------------------------------------------------------*
*                                                                   *
*   If the FLAG4 REPLACE operation fails, issue a WTO console       *
*   message that indicates the userid for which the failure occurred *
*   and the SAF and RACF return codes generated by the RACROUTE     *
*   request.                                                        *
*                                                                   *
*-------------------------------------------------------------------*
         MVC   WTOWRK(WTOLN),WTOLST   Copy WTO model
         MVC   WTOWRK+4(L'WTOMSG2),WTOMSG2 Copy message model
         MVC   WTOWRK+4+35(8),0(R6)   Copy the userid
         ST    R15,DBL2               Save the SAF return code
         UNPK  DBL1(9),DBL2(5)        Unpack
         NC    DBL1(8),=8X'0F'        Turn off high order nibbles
         TR    DBL1(8),=C'0123456789ABCDEF' Make things readable
         MVC   WTOWRK+4+53(4),DBL1+4  Copy SAF return code
         MVC   DBL2(4),ROUTWRK1       Save the RACF return code
         UNPK  DBL1(9),DBL2(5)       Unpack
         NC    DBL1(8),=8X'ØF'       Turn off high order nibbles
         TR    DBL1(8),=C'Ø123456789ABCDEF' Make things readable
         MVC   WTOWRK+4+68(4),DBL1+4 Copy RACF return code
         WTO   MF=(E,WTOWRK)         Issue the WTO
         B     RETURN                Return
         DROP  R5
*-----------------------------------------------------------------------*
RETURN   DS    ØH
         L     R3,RIXWA              Get address of work word
         L     R5,Ø(,R3)             Get ICHRIXØ1 work area address
         LTR   R5,R5                 A work area?
         BZ    NORIXWA               No - don't release storage
         STORAGE RELEASE,LENGTH=RIXWLEN,ADDR=(R5)
NORIXWA  DS    ØH
         L     R3,SAVEAREA+4         Copy old savearea address
         LR    R1,R13                Get temp storage address
         STORAGE RELEASE,LENGTH=WORKLEN,ADDR=(R1)
         LR    R13,R3                Copy the old savearea address
         LM    R14,R12,12(R13)       Restore the registers
         XR    R15,R15               RC=Ø
         BR    R14                   Return
*----------------------------------------------------------------------*
WTOMSG1  DC    C'ICHRIXØ2 - Password violation revocation captured for x
               xxxxxxxx and updated in BASE segment FLAG4 data.'
*-----------------------------------------------------------------------*
WTOMSG2  DC    C'ICHRIXØ2 - FLAG4 update failed for xxxxxxxx.  SAF RC(xx
               xxx)  RACF RC(xxxx).'
*----------------------------------------------------------------------*
WTOLST   WTO   '                                                        x
                                                                        x
                        ',ROUTCDE=(1),DESC=(6),MF=L
WTOLN    EQU   *-WTOLST
*----------------------------------------------------------------------*
FLDLIST1 DC    F'3'
         DC    CL8'FLAG4   '
         DC    CL8'REVOKEDT'
         DC    CL8'RESUMEDT'
*----------------------------------------------------------------------*
FLDLIST2 DC    F'1'
         DC    CL8'FLAG4   '
*----------------------------------------------------------------------*
RACROUT1 RACROUTE REQUEST=EXTRACT,                                      x
               TYPE=EXTRACT,                                            x
               CLASS='USER',                                            x
               RELEASE=1.9.2,                                           x
               MF=L
ROUTLEN1 EQU   *-RACROUT1
*----------------------------------------------------------------------*
```

```
WORKAREA DSECT
SAVEAREA DS    18F
FLAG4LEN DS    F
FLAG4SAV DS    XL1
RVKDTSAV DS    XL3
RSMDTSAV DS    XL3
WTOWRK   DS    ØD,CL(WTOLN)
ROUTWRK1 DS    ØD,CL(ROUTLEN1)
RACWORK  DS    ØD,CL(512)
DBL1     DS    2D
DBL2     DS    2D
WORKLEN  EQU   *-WORKAREA
*-------------------------------------------------------------------*
RIXWAREA DSECT
RIXWID   DS    CL4
RIXWUID  DS    CL8
RIXWFLGS DS    ØF
RIXWFLG1 DS    XL1
RIXWFLG2 DS    XL1
RIXWFLG3 DS    XL1
RIXWFLG4 DS    XL1
RIXWRVDT DS    XL3
RIXWRSDT DS    XL3
RIXWLEN  EQU   *-RIXWAREA
*-------------------------------------------------------------------*
         ICHRIXP ,
*--------------------------------------------------------------------*
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R1Ø      EQU   1Ø
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
         END
```

## RVKLST ASSEMBLER

```
RVKLST   CSECT
RVKLST   AMODE 31
RVKLST   RMODE 24                        DCBS NEED 24-BIT ADDRESSES
***********************************************************************
*                                                                     *
*   THE RVKLST PROGRAM CAN WORK WITH ANY RACF DATABASE, BUT IS        *
*   ESPECIALLY DESIGNED TO WORK WITH RACF EXITS AS FOLLOWS:           *
*   - AN IRREVXØ1 RACF COMMAND EXIT THAT SETS THE X'Ø8' BIT IN THE    *
*     FLAG4 USER BASE SEGMENT FIELD WHEN A USERID BECOMES REVOKED     *
*     VIA A RACF ALTUSER COMMAND                                      *
*   - AN ICHRIXØ1/ICHRIXØ2 VERIFY(X) PRE/POST EXIT PAIR THAT CAUSE    *
*     THE X'Ø4' BIT TO BE SET IN THE FLAG4 USER BASE SEGMENT FIELD    *
*     WHEN A USERID BECOMES REVOKED BECAUSE OF TOO MANY INVALID       *
*     PASSWORD ATTEMPTS                                               *
*                                                                     *
*   THIS FLAG IS EXAMINED BY THE RVKLST PROGRAM AND IT CAN BE USED TO *
*   DIFFERENTIATE BETWEEN A USERID THAT HAS BEEN REVOKED DUE TO TOO   *
*   MANY INVALID LOGON ATTEMPTS AND A USERID THAT HAS BEEN REVOKED    *
*   BY A RACF ADMINISTRATOR.                                          *
*                                                                     *
*   THE RVKLST PROGRAM SHOULD RESIDE IN AN APF AUTHORIZED LIBRARY     *
*   AND SHOULD BE LINKEDITED AC(1).                                   *
*                                                                     *
*   THIS CAN BE A USEFUL TOOL FOR A RACF ADMINSTRATOR WHO MAY BE      *
*   TRYING TO DETERMINE WHETHER THE REVOKE STATUS OF A CERTAIN USERID *
*   SHOULD BE CHANGED TO RESUME.  A USERID THAT IS IN 'REVOKED'       *
*   STATUS MAY BE A GOOD CANDIDATE TO BE RESUMED, BUT ONE THAT IS     *
*   'Date REVOKED' OR 'Administrator REVOKED' MAY REQUIRE MORE        *
*   SCRUTINY.                                                         *
*                                                                     *
***********************************************************************
         PRINT GEN
         STM   R14,R12,12(R13)           SAVE THE REGISTERS
         LR    R12,R15                   COPY MODULE BASE ADDRESS
         LA    R11,4Ø95(,R12)            SET SECOND BASE ...
         LA    R11,1(,R11)                  REGISTER ADDRESS
         USING RVKLST,R12,R11            SET ADDRESSABILITY
         LR    R1Ø,R13                   SAVE OLD SAVEAREA ADDRESS
         LR    R2,R1                     SAVE INCOMING PARM ADDRESS
         STORAGE OBTAIN,LENGTH=WALEN     GET SOME WORKING STORAGE
         LR    R13,R1                    COPY THE ADDRESS
         LR    RØ,R13                    AGAIN
         L     R1,=A(WALEN)              SET THE LENGTH
         LR    R14,R13                   SET SOURCE ADDRESS TO TARGET
         XR    R15,R15                   SET FILL BYTE
         MVCL  RØ,R14                    CLEAR THE STORAGE
         ST    R1Ø,4(,R13)               SAVE OLD SAVEAREA ADDRESS
         USING WORKAREA,R13              WORKING STORAGE ADDRESSABILITY
***********************************************************************
         OPEN  (SYSPRINT,OUTPUT),MODE=31 OPEN OUTPUT DATASET
         PUT   SYSPRINT,HDR1             WRITE FIRST HEADER
         PUT   SYSPRINT,HDR2             WRITE SECOND HEADER
***********************************************************************
         L     R8,Ø(,R2)                 GET PARAMETER ADDRESS
         CLC   Ø(2,R8),=H'Ø'             ANY PARM DATA?
         BE    USRIDLST                  NO - IT'S A FULL LIST
         CLC   Ø(2,R8),=H'8'             TOO MUCH DATA FOR A USERID?
         BH    RETURN4                   YES - SET RC=4
         B     ONEUSRID                  PROCESS ONE USERID
***********************************************************************
USRIDLST EQU   *
         XC    XUID(4),XUID              CLEAR XUID LENGTH AREA
         MVC   XUID(2),=H'8'             SET DATA LENGTH
         MVC   XUID+4(8),=8C' '          SET STARTING UID
UIDLOOP  EQU   *
         XC    RACWORK(256),RACWORK      CLEAR RACROUTE ...
         XC    RACWORK+256(256),RACWORK+256 WORKAREA STORAGE
         MVC   ROUTWRK1(ROUTLEN1),RACROUT1 COPY RACROUTE PARM MODEL
         RACROUTE REQUEST=EXTRACT,                                     X
               TYPE=EXTRACTN,                                          X
               ENTITYX=XUID,                                           X
               FIELDS=FLDLIST1,                                        X
               RELEASE=1.9.2,                                          X
               SUBPOOL=1,                                              X
               WORKA=RACWORK,                                          X
               MF=(E,ROUTWRK1)
         LTR   R15,R15                   EXTRACT OK?
         BZ    LISTOK                    YES - PROCESS DATA
         ST    R15,RETCODE               SAVE THE RETURN CODE
         MVC   RACF_RC(8),ROUTWRK1       COPY RACF RTN/RSN CODES
         B     LISTDONE                  WE'RE DONE
LISTOK   EQU   *
***********************************************************************
*                                                                     *
*   A USERID WAS EXTRACTED.  MOVE THE DATA INTO AN OUTPUT BUFFER AND  *
*   WRITE THE RECORD.                                                 *
*                                                                     *
*   USE THE DOOUTPUT ROUTINE FOR THIS PURPOSE.  R1 SHOULD CONTAIN     *
*   THE EXTRACT BUFFER ADDRESS.  THE BUFFER IS RELEASED BY DOOUTPUT.  *
*                                                                     *
***********************************************************************
         MVC   SAVEUID(8),XUID+4         COPY THE USERID
         BAL   R14,DOOUTPUT              CREATE NECESSARY OUTPUT
         B     UIDLOOP                   PROCESS NEXT USERID
LISTDONE EQU   *
         CLOSE (SYSPRINT),MODE=31        CLOSE OUTPUT DATASET
         B     RETURNØ                   WE'RE DONE
***********************************************************************
ONEUSRID EQU   *
         MVC   SAVEUID(8),=8C' '         CLEAR THE TARGET USERID FIELD
         XR    R15,R15                   CLEAR R15
         ICM   R15,B'ØØ11',Ø(R8)         GET THE USERID LENGTH
         BCTR  R15,Ø                     REDUCE BY ONE FOR EX
         EX    R15,USRIDMVC              COPY THE USERID
         XC    RACWORK(256),RACWORK      CLEAR RACROUTE ...
         XC    RACWORK+256(256),RACWORK+256 WORKAREA STORAGE
         MVC   ROUTWRK2(ROUTLEN2),RACROUT2 COPY RACROUTE PARM MODEL
         RACROUTE REQUEST=EXTRACT,                                     X
               TYPE=EXTRACT,                                           X
               ENTITY=SAVEUID,                                         X
               FIELDS=FLDLIST1,                                        X
               RELEASE=1.9.2,                                          X
               SUBPOOL=1,                                              X
               WORKA=RACWORK,                                          X
               MF=(E,ROUTWRK2)
         LTR   R15,R15                   EXTRACT OK?
         BZ    DOUSRID                   YES - PROCESS DATA
         MVC   OUTREC(133),=133C' '      CLEAR THE OUTPUT RECORD
         MVC   OUTREC(8),SAVEUID         COPY THE USERID
         MVC   OUTREC+1Ø(11),=C'Not defined' USERID NOT IN DATABASE
         PUT   SYSPRINT,OUTREC           WRITE THE OUTPUT RECORD
         B     LISTDONE                  GO FINISH UP
DOUSRID  EQU   *
***********************************************************************
*                                                                     *
*   THE USERID DATA WAS EXTRACTED.  MOVE THE DATA INTO AN OUTPUT      *
*   BUFFER AND WRITE THE RECORD.                                      *
*                                                                     *
*   USE THE DOOUTPUT ROUTINE FOR THIS PURPOSE.  R1 SHOULD CONTAIN     *
*   THE EXTRACT BUFFER ADDRESS.  THE BUFFER IS RELEASED BY DOOUTPUT.  *
*                                                                     *
***********************************************************************
         BAL   R14,DOOUTPUT              CREATE NECESSARY OUTPUT
         B     LISTDONE                  GO FINISH UP
***********************************************************************
RETURNØ  EQU   *
         L     R1Ø,4(,R13)               SAVE INCOMING SAVEAREA ADDR
         LR    R1,R13                    COPY TEMP STORAGE ADDR
         STORAGE RELEASE,LENGTH=WALEN,ADDR=(R1)
         LR    R13,R1Ø                   COPY INCOMING SAVEAREA ADDR
         LM    R14,R12,12(R13)           RESTORE REGISTERS
         XR    R15,R15                   SET RETURN CODE
         BR    R14                       RETURN
***********************************************************************
RETURN4  EQU   *
         L     R1Ø,4(,R13)               SAVE INCOMING SAVEAREA ADDR
         LR    R1,R13                    COPY TEMP STORAGE ADDR
         STORAGE RELEASE,LENGTH=WALEN,ADDR=(R1)
         LR    R13,R1Ø                   COPY INCOMING SAVEAREA ADDR
         LM    R14,R12,12(R13)           RESTORE REGISTERS
         LA    R15,4                     SET RETURN CODE
         BR    R14                       RETURN
***********************************************************************
DOOUTPUT EQU   *
         STM   RØ,R15,SVAREAØ2           SAVE REGISTERS
         XC    SEGDATA(SEGDATAL),SEGDATA CLEAR SEGMENT DATA SAVE AREA
         XR    R6,R6                     CLEAR R6
         ICM   R6,B'ØØ11',4(R1)          GET DATA OFFSET
         AR    R6,R1                     CALCULATE DATA ADDRESS
SVFLD1   EQU   *
         ICM   R15,B'1111',Ø(R6)         GET DATA LENGTH
         MVC   SAVEFLG4(1),4(R6)         COPY FLAG4
SVFLD2   EQU   *
         LA    R6,4(R15,R6)              POINT TO REVOKEDT
         ICM   R15,B'1111',Ø(R6)         GET DATA LENGTH
         LTR   R15,R15                   ANY DATA?
         BZ    SVFLD3                    NO - CHECK NEXT FIELD
         MVC   SAVERVDT(3),4(R6)         SAVE REVOKEDT
SVFLD3   EQU   *
         LA    R6,4(R15,R6)              POINT TO RESUMEDT
         ICM   R15,B'1111',Ø(R6)         GET DATA LENGTH
         LTR   R15,R15                   ANY DATA?
         BZ    SVFLD4                    NO - CHECK NEXT FIELD
         MVC   SAVERSDT(3),4(R6)         SAVE RESUMEDT
SVFLD4   EQU   *
***********************************************************************
         XR    R7,R7                     CLEAR R7
         XR    R8,R8                     CLEAR R8
         ICM   R7,B'Ø111',1(R1)          GET STORAGE LENGTH
         ICM   R8,B'ØØØ1',Ø(R1)          GET SUBPOOL
         STORAGE RELEASE,LENGTH=(R7),ADDR=(R1),SP=(R8)
***********************************************************************
RVKCHK   EQU   *
         MVC   OUTREC(133),=133C' '      CLEAR THE OUTPUT RECORD
         MVC   OUTREC(8),SAVEUID         COPY THE USERID
         TIME  DEC                       GET CURRENT DATE/TIME
         STCM  R1,B'Ø111',CURRDATE       SAVE CURRENT DATE
         TM    SAVEFLG4,X'Ø8'            ADMIN REVOKE?
         BZ    CHKLGRVK                  NO - CHK INVALID PWD REVOKE
         CLC   CURRDATE(3),SAVERSDT      CURRDATE >= RESUMEDT?
         BL    SETRVK1                   NO - REVOKED
         CLC   SAVERSDT(3),=3X'ØØ'       A 'REAL' RESUMEDT?
         BNE   NORVK                     YES - NOT REVOKED
SETRVK1  EQU   *
         MVC   OUTREC+1Ø(21),=C'Administrator REVOKED' SET RVK STATUS
         B     RVKCHKDN                  WE'RE DONE THIS USERID
CHKLGRVK EQU   *
         TM    SAVEFLG4,X'8Ø'            REVOKED?
         BZ    DTRVKCHK                 NO - CHECK IF DATE REVOKED
         TM    SAVEFLG4,X'Ø4'           PWD VIOLATION REVOKE?
         BO    SETRVK3                  YES - SET PWD VIO MESSAGE
         CLC   CURRDATE(3),SAVERSDT     CURRDATE >= RESUMEDT?
         BL    SETRVK2                  NO - REVOKED
         CLC   SAVERSDT(3),=3X'ØØ'      A 'REAL' RESUMEDT?
         BNE   NORVK                    YES - NOT REVOKED
SETRVK2  EQU   *
         MVC   OUTREC+1Ø(7),=C'REVOKED'  SET REVOKE STATUS
         B     RVKCHKDN                 WE'RE DONE THIS USERID
SETRVK3  EQU   *
         MVC   OUTREC+1Ø(26),=C'Password violation REVOKED' SET RVK ST
         B     RVKCHKDN                 WE'RE DONE THIS USERID
DTRVKCHK EQU   *
         CLC   SAVERSDT(3),SAVERVDT     RESUMEDT < REVOKEDT?
         BL    RVKOUTWN                 YES - CURRDATE OUT OF RANGE
RVKINWN  EQU   *
         CLC   CURRDATE(3),SAVERSDT     CURRDATE >= RESUMEDT?
         BNL   NORVK                    YES - NOT REVOKED
         CLC   CURRDATE(3),SAVERVDT     CURRDATE < REVOKEDT?
         BL    NORVK                    YES - NOT REVOKED
DATERVK  EQU   *
         MVC   OUTREC+1Ø(12),=C'Date REVOKED' SET REVOKE STATUS
         B     RVKCHKDN                 WE'RE DONE THIS USERID
RVKOUTWN EQU   *
         CLC   CURRDATE(3),SAVERSDT     CURRDATE < RESUMEDT?
         BL    DATERVK                  YES - REVOKED
         CLC   CURRDATE(3),SAVERVDT     CURRDATE >= REVOKEDT?
         BNL   DATERVK                  YES - REVOKED
         B     NORVK                    OUT OF WINDOW, NO REVOKE
NORVK    EQU   *
         MVC   OUTREC+1Ø(11),=C'Not REVOKED' SET REVOKE STATUS
         B     RVKCHKDN                 WE'RE DONE THIS USERID
RVKCHKDN EQU   *
         PUT   SYSPRINT,OUTREC          WRITE THE OUTPUT RECORD
***********************************************************************
         LM    RØ,R15,SVAREAØ2          LOAD REGISTERS
         BR    R14                      RETURN
***********************************************************************
USRIDMVC MVC   SAVEUID(*-*),2(R8)       COPY THE USERID
***********************************************************************
SYSPRINT DCB   MACRF=(PM),LRECL=133,DSORG=PS,DDNAME=SYSPRINT
***********************************************************************
HDR1     DC    CL133'Userid    Revoke Status'
HDR2     DC    CL133'--------  -------------------------'
***********************************************************************
RACROUT1 RACROUTE REQUEST=EXTRACT,                                     X
               TYPE=EXTRACTN,                                          X
               CLASS='USER',                                           X
               RELEASE=1.9.2,                                          X
               MF=L
ROUTLEN1 EQU   *-RACROUT1
***********************************************************************
RACROUT2 RACROUTE REQUEST=EXTRACT,                                     X
               TYPE=EXTRACT,                                           X
               CLASS='USER',                                           X
               RELEASE=1.9.2,                                          X
               MF=L
ROUTLEN2 EQU   *-RACROUT2
***********************************************************************
FLDLIST1 DC    F'3'
         DC    C'FLAG4   '              REVOKE
         DC    C'REVOKEDT'              REVOKE DATE
         DC    C'RESUMEDT'              RESUME DATE
***********************************************************************
         LTORG ,
***********************************************************************
WORKAREA DSECT
SAVEAREA DS    18F
SVAREAØ2 DS    18F
RETCODE  DS    F
RACF_RC  DS    F
RACF_RSN DS    F
EXTSAVE  DS    F
ROUTWRK1 DS    ØD,CL(ROUTLEN1)
ROUTWRK2 DS    ØD,CL(ROUTLEN2)
XUID     DS    ØD
         DS    F
         DS    CL8
***********************************************************************
RACWORK  DS    ØD,CL512
DBL1     DS    2D
DBL2     DS    2D
SAVEUID  DS    CL8
SEGDATA  DS    ØD
SAVEFLG4 DS    XL1
SAVERVDT DS    XL3
SAVERSDT DS    XL3
SEGDATAL EQU   *-SEGDATA
OUTREC   DS    CL133
CURRDATE DS    XL3
WALEN    EQU   *-WORKAREA
RØ       EQU   Ø
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
```

```
R8        EQU    8
R9        EQU    9
R1Ø       EQU    1Ø
R11       EQU    11
R12       EQU    12
R13       EQU    13
R14       EQU    14
R15       EQU    15
          IRRPRXTW
          ICHPRCVT
          CVT    DSECT=YES
          END
```

*Rudy Douglas (Canada)* © Xephon 2004

# RACF protection for IND$FILE

## INTRODUCTION

RACF was designed with five levels of access authorization to datasets in mind – EXECUTE, CONTROL, UPDATE, ALTER, and READ. This dates from the days when interactive access to mainframe datasets was achieved almost exclusively using dumb 3270 terminals. Since the good old 3278s were replaced almost everywhere with intelligent PCs, there should be an extra level that could be summarized as USE. The authority to read a file is obviously not the same as the right to download the file out of its own environment (or the other way around). Imagine, for instance, a customer file with client data. Some employees undoubtedly need the right to consult this information, but this does not amount to the same as the right to take data with them when they leave the company. There exist several ways to extract data from an OS/390 system. To name just a few:

- Remote Job Entry (RJE).
- Network Job Entry (NJE).
- Netview File Transfer Program (NFTP).
- Mail systems (SMTP).
- TCP/IP File Transfer Program (FTP).
- Hypertext Transfer Protocol (HTTP).
- 3270 Emulation program macros.

One of the most easy to use, however, is the nice Graphical User Interface (GUI) that comes with almost every PC-based 3270 emulation program, such as IBM's eNetwork Personal Communications or Attachmate's Extra Personal Client. For TSO/E sessions (most of them also support a CICS FTP), this form of file transfer relies on the IND$FILE program on the host. The IND$FILE command processor comes standard with OS/390 and z/OS and is located in the linklist dataset SYS1.CMDLIB. Without any special knowledge, everybody can download a file to a PC medium – such as a floppy, USB drive, CD, or DVD – provided they have RACF READ access to the mainframe dataset.

The purpose of this article is to describe a way to protect the use of the IND$FILE command processor by RACF profiles.

## MECHANISM

When a 3270 emulation program downloads or uploads a file in a TSO/E environment, the client program will execute a command processor called IND$FILE with a number of arguments (called subcommands). The subcommands consist of the direction of the transfer (GET or PUT), the name of the file (with or without quotes), the carriage return line feed (CRLF) conversion, the ASCII/EBCDIC translation, etc. The SMP/E user modification below wraps the original IND$FILE with a small program that calls RACF to see whether the user has access to the FACILITY class profile:

```
IND$FILE.<direction>.<datasetname>
```

where <direction> is 'GET' or 'PUT' and <dataset name> can be a fully or partly specified file name.

Independently of the user's access rights in the DATASET class, the file transfer is then accepted or refused. If accepted, it could well be that RACF still refuses the download or upload if the DATASET class access right is NONE for the user. On the other hand, the user could have the highest possible DATASET class access level (ALTER), but could be refused by the FACILITY class profile to download or upload the file.

The design is not flawless. Imagine that a user may read datasets A and B, but is authorized to download only A. It might be possible for this user to rename B to A and the download would go through. On the other hand, with a careful implementation, this kind of situation can be avoided. The most important thing is that at least a binary implementation (yes you may, no you may not) can now be put in place. How many people in the organization really need to transfer files from or to a PC from TSO/E? Probably not that many. By activating the mechanism, RACF is also capable of auditing the transfers so that there is at least a trace in case of problems.

## IMPLEMENTATION

Putting the program in place is quite straightforward if the definitions are left unchanged. Alternatively you may prefer to use other profile names or even an entirely new RACF class. The implementation of the usermod 'as is' requires the following actions.

### RACF

#### *Class*

The FACILITY class was provided by IBM as a container for user implemented profiles. Using a different class is possible but activating one requires changes in the RACF class (ICHRRCDE) and router (ICHRFR01) tables. If the class is defined, it must also be activated by the command:

```
SETROPTS CLASSACT(<classname>)
```

If the class is raclisted, like FACILITY for instance, don't forget to refresh it by issuing:

```
SETROPTS REFRESH RACLIST(<classname>)
```

#### *Profiles*

Assuming that you want the default to be that nobody can download or upload files, and that you use the FACILITY class:

```
RDEFINE FACILITY IND$FILE.*.* UACC(NONE)
```

The members of the SYSPROG group have the right to download SYS1 files:

```
RDEFINE FACILITY IND$FILE.GET.SYS1.* UACC(NONE)
PERMIT IND$FILE.GET. SYS1.* CLASS(FACILITY) ID(SYSPROG) ACC(READ)
```

The APPL1 group can only log-on to the system from diskless workstations (defined to RACF as TNODISK). They may upload and download any dataset with APPL1 as the First-Level Qualifier (FLQ) because they produce reports based on mainframe data that are afterwards uploaded and distributed by the mainframe again.

```
RDEFINE FACILITY IND$FILE.*.APPL1.* UACC(NONE)
PERMIT IND$FILE.*.APPL1.* CLASS(FACILITY)     ID(APPL1) ACC(READ) +
WHEN(TERMINAL(TNODISK))
```

## SMP/E

The IND$FILE module is know to SMP/E, so every modification to it must be done using a user modification. Two new macros used in the wrapping program – EYECATCH and AMODE24 – are introduced to the MVS base element Function Management Identification (FMID). The source of the module itself is related to the FMID of the original IND$FILE load module. In order to keep all our user modifications in an easy to reinstall format, we define them into distribution libraries (DLIBs) of our own. Assembler programs go into JEDPLX1.V2R7.AJEDMAC, load modules into AJEDLINK, and macros into AJEDSRC. Before implementing, check the FMIDs and the linkage-edit parameters of IND$FILE as seen by SMP/E.

## Example

The RACF display:

```
RACF - GENERAL RESOURCE SERVICES -  DISPLAY
OPTION ===>

ENTER THE FOLLOWING PROFILE INFORMATION:

   CLASS     ===> facility

   PROFILE   ===> ind$file.get.jedplx1.**
```

```
                              <==end of data
```

Returns:

```
BROWSE - RACF COMMAND OUTPUT------------------ LINE 00000000 COL 001 080
 COMMAND ===>                                           SCROLL ===> CSR
****************************** Top of Data ****************************
CLASS      NAME
-----      ----
FACILITY   IND$FILE.GET.JEDPLX1.** (G)

LEVEL  OWNER      UNIVERSAL ACCESS  YOUR ACCESS  WARNING
-----  --------   ----------------  -----------  -------
 00    IBZZEX4         NONE              NONE     NO
```

The file transfer triggered by the 3270 emulation program generated:

```
ISPF Command Shell
 Enter TSO or Workstation commands below:
 ===> IND$FILE GET JEDPLX1.L.ASM(IND2) ASCII CRLF
```

which produced the message:

```
SE-IND$FILE: MSG04 NON-ZERO SAF OR RACF RETURN/REASON CODE:
SE-IND$FILE: MSG04A SAF  RETURN CODE: 00000008
SE-IND$FILE: MSG04A RACF RETURN CODE: 00000008
SE-IND$FILE: MSG04A RACF REASON CODE: 00000000
```

The user had ALTER access to the JEDPLX1.** datasets.

## CODE

The job below installs the SMP/E user modification. When the APPLY CHECK is OK, it must be replaced by an APPLY to effectively install the program.

```
//IBZZEX4X JOB (JAN),'JAN DE DECKER',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID,
//             REGION=0M
//*
//* PREPARATION: COPY THE SOURCES TO A TEMPORARY VIO DATASET
//*
//S0       EXEC PGM=IEBUPDTE,PARM='NEW'
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
./ ADD NAME=IND$FILX
         TITLE '*** IND$FILX: RACF IND$FILE WRAPPER                JANX
               DE DECKER ***'
*----------------------------------------------------------------------
```

```
* JED:SP N.V.   SCHUITENKAAI 3 1000 BRUSSEL    jan.de.decker@tiscali.be
*----------------------------------------------------------------------
*
* NAME:          IND$FILX
*
* PARAMETERS:    STANDARD TSO/E COMMAND PROCESSOR CPPL
*
* PURPOSE:       WRAPS IND$FILE
*
* SYSTEM:        OS/390 z/OS
*
* LINK:          AMODE 24
*                RMODE 24
*                AC = 0 IS SUFFICIENT
*                REENTRANT
*                CHECK SMP/E SETTINGS OF IND$FILE
*
* USE:           APPLY AS A USERMOD TO PROVIDE SECURITY FOR IND$FILE
*                PROCESSING
*
* AUTHOR:        JAN                  DATE: 12/2003
*
* SAMPLE:        N/A
*
* NOTES:         FACILITY CLASS IS USED
*                DON'T USE WTO OR TPUT BEFORE PASSING CONTROL TO
*                THE REAL ENTRY POINT SINCE THE TSO/E '***' BLOCKS
*                AT LEAST MY EMULATION
*
* MODIFICATION:
*
*----------------------------------------------------------------------
R0       EQU   0          ALL REFERENCES TO REGISTERS MAPPED BY
R1       EQU   1          ASSEMBLER XREF OPTION
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
RA       EQU   10
RB       EQU   11
```

```
RC       EQU   12
RD       EQU   13
RE       EQU   14
RF       EQU   15
         EJECT
IND$FILX CSECT
IND$FILX AMODE 24
IND$FILX RMODE 24
         PRINT GEN
         BAKR  RE,Ø                    STACK REGISTERS AND PSW
         LR    RC,RF                   RC -> OUR CSECT
         USING IND$FILX,RC             ADDRESS IND$FILX WITH RC
         LR    RA,R1                   RA -> PARAMETER LIST
         EYECATCH                      AS IT SAYS
         AMODE24                       CHANGES AMODE,RØ AND R1
*
* WE MUST BE REENTRANT, FETCH SOME STORAGE, CLEAR IT AND DO THE
* SAVEAREA THING
*
         GETMAIN RC,                   STORAGE FOR WORKAREA            X
               LV=L_WORK               AMOUNT IS EQUATED
         LTR   RF,RF                   GETMAIN OK?
         BZ    LØØØØ                   YES -> CONTINUE
         ABEND 1ØØ                     NO --> ABEND
LØØØØ    DS    ØH
         LR    R2,R1                   R2 -> WORK AREA
         LR    R4,R1                   R4 -> WORK AREA
         LR    R6,R1                   R6 -> WORK AREA
         LA    R7,L_WORK               R7 = L(WORK AREA)
         XR    R5,R5                   R5 = Ø
         MVCL  R6,R4                   CLEAR WORK AREA
         USING WORKAREA,R2             R2 ADDRESSES OUR WORKAREA
         MVC   SAVEAREA+4(4),=C'F1SA'  SET LINKAGE STACK INDICATOR
         LA    RD,SAVEAREA             RD -> SAVEAREA
*
         ST    RA,PARMS                KEEP PARAMETER POINTER
         EJECT
*
* START OF PROCESSING
*
* R2 ADDRESSES OUR WORKAREA DSECT
* RA ADDRESSES OUR CPPL
* RC ADDRESSES OUR CSECT
* RD POINTS TO OUR SAVEAREA
*
* R3 WILL ADDRESS THE COMMAND LINE
* R4 WILL CONTAIN THE LENGTH OF THE COMMAND BUFFER INCLUDING THE 4
*    BYTE HEADER
* R8 WILL SHUFFLE THROUGH THE DSN PART OF THE ENTITY FIELD
* R9 WILL COUNT THE BYTES IN THE ENTITY
```

```
*
* DO SOME RUDIMENTARY CHECKS AND FETCH THE SUBCOMMAND (GET/PUT)
*
         MVC   D_PRE,S_PRE             MESSAGE PREFIX
         USING CPPL,RA                 RA ADDRESSES THE PARAMETER LIST
         L     R3,CPPLCBUF             R3 -> COMMAND BUFFER
         LH    R4,Ø(R3)                R4 = L(COMMAND BUFFER)
         CH    R4,=H'17'               MINIMUM LENGTH = 14 LIKE IN
         BH    LØØ1Ø                   LENGTH IS OK
*                                      (4B HEADER) + 'IND$FILE GET X'
* THE COMMAND LENGTH IS TOO SMALL:
* ISSUE MESSAGE MSGØØ AND RETURN TO CALLER
*
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L'MSGØØ),MSGØØ      MESSAGE TO DYNAMIC BUFFER
         LA    RØ,L_S_PRE+L'MSGØØ      RØ = L(MESSAGE)
         LA    R1,TPUT_MSG             R1 -> MESSAGE
         TPUT  (R1),(RØ)               WRITE TO TERMINAL
         B     THE_END                 AND RETURN TO CALLER
LØØ1Ø    DS    ØH
         LA    R3,4(R3)                R3 -> COMMAND
         SH    R4,=H'4'                --(THE 4 HEADER BYTES)
         CLC   =C'IND$FILE ',Ø(R3)     COMMAND OK?
         BE    LØØ2Ø                   YES -> CONTINUE
*
* INTERNAL TSO/E INCONSISTENCY, THE COMMAND IS NOT IND$FILE
* ISSUE MESSAGE MSGØ1
*
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L'MSGØ1),MSGØ1      MESSAGE TO DYNAMIC BUFFER
         LA    RØ,L_S_PRE+L'MSGØ1      RØ = L(MESSAGE)
         LA    R1,TPUT_MSG             R1 -> MESSAGE
         TPUT  (R1),(RØ)               WRITE TO TERMINAL
         B     THE_END                 AND RETURN TO CALLER
         EJECT
LØØ2Ø    DS    ØH
         LA    R3,9(R3)                R3 -> FIRST SUBCOMMAND
         OC    Ø(3,R3),BLANKS          TRANSLATE TO UPPER CASE
         CLC   =C'GET ',Ø(R3)          IS IT A GET REQUEST?
         BE    LØØ3Ø                   NO --> GO CHECK PUT
         CLC   =C'PUT ',Ø(R3)          IS IT A PUT REQUEST?
         BE    LØØ3Ø                   NO --> GO CHECK PUT
*
* THE SUBCOMMAND IS 'GET' NOR 'PUT'
* ISSUE MESSAGE MSGØ2 AND RETURN TO CALLER
*
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L'MSGØ2),MSGØ2      MESSAGE TO DYNAMIC BUFFER
         LA    RØ,L_S_PRE+L'MSGØ2      RØ = L(MESSAGE)
         LA    R1,TPUT_MSG             R1 -> MESSAGE
         TPUT  (R1),(RØ)                 WRITE TO TERMINAL
         B     THE_END                   AND RETURN TO CALLER
         EJECT
*
* CONSTRUCT THE DATASETNAME FROM THE COMMANDLINE AND THE PROFILE PREFIX
*
LØØ3Ø    DS    ØH
         SH    R4,=H'13'                 R4 = #(REMAINING BYTES)
         MVC   PROFILE,=CL9'IND$FILE.'   SET PROFILE PREFIX
         MVC   SUBCOM(3),Ø(R3)           GET THE SUBCOMMAND IN THE ENTITY
         MVI   SUBCOM+3,C'.'             ADD '.'
         LA    R9,13                     R9 = L(RACF ENTITY) SOFAR
         LA    R3,4(R3)                  R3 -> FIRST CHARACTER DSN
         LA    R8,DSN                    R8 -> START DSN PART ENTITY
         CLI   Ø(R3),X'7D'               IS IT A QUOTE?
         BNE   LØØ4Ø                     YES-> GO COPY DSN AS IS
         LA    R3,1(R3)                  R3 -> PAST QUOTE
         BCTR  R4,Ø                      --#(REMAINING BYTES IN COMMAND)
         LTR   R4,R4                     ONLY A QUOTE?
         BNZ   LØØ5Ø                     NO --> CONTINUE WITH DSN
*
* THE DSNAME CONSISTS OF ONLY A QUOTE
* ISSUE MESSAGE MSGØ3 AND RETURN TO CALLER
*
         MVC   MSG,BLANKS                BLANK OUT MESSAGE LINE
         MVC   MSG(L'MSGØ3),MSGØ3        MESSAGE TO DYNAMIC BUFFER
         LA    RØ,L_S_PRE+L'MSGØ3        RØ = L(MESSAGE)
         LA    R1,TPUT_MSG               R1 -> MESSAGE
         TPUT  (R1),(RØ)                 WRITE TO TERMINAL
         B     THE_END                   AND RETURN TO CALLER
         EJECT
LØØ4Ø    DS    ØH
         L     RA,CPPLUPT                RA -> USER PROFILE TABLE
         DROP  RA                        FORGET THE CPPL
         USING UPT,RA                    RA ADDRESSES THE UPT
         XR    R5,R5                     R5 = Ø
         IC    R5,UPTPREFL               R5 = L(PREFIX)
         LTR   R5,R5                     IS THERE A PREFIX?
         BZ    LØØ5Ø                     NO --> CONTINUE
         AR    R8,R5                     ADAPT THE DSN POINTER
         AR    R9,R5                     ADAPT THE LENGTH OF THE ENTITY
         BCTR  R5,Ø                      --R5 FOR EXECUTE
         LA    R6,DSN                    R8 -> START DSN PART ENTITY
         LA    R7,UPTPREFX               R7 -> PREFIX
         DROP  RA                        FORGET THE UPT
         EX    R5,S_MVC                  MOVE TSO PROFILE TO RACF ENTITY
         MVI   Ø(R8),C'.'                ADD '.' AFTER THE PREFIX
         LA    R8,1(R8)                  ++(DSN NEXT BYTE POINTER)
         LA    R9,1(R9)                  ++LENGTH(ENTITY)
LØØ5Ø    DS    ØH
```

```
         XR    R5,R5                   R5 = Ø (COUNTER)
         LR    R6,R3                   R6 -> FIRST BYTE DSNAME
         LR    R7,R4                   R7 = REMAINING # BYTES IN BUFFER
LØØ6Ø    DS    ØH
         CLI   Ø(R6),C'('              START OF MEMBER?
         BE    LØØ7Ø                   YES -> WE ARE AT THE END
         CLI   Ø(R6),X'7D'             QUOTE?
         BE    LØØ7Ø                   YES -> WE ARE AT THE END
         CLI   Ø(R6),C' '              BLANK?
         BE    LØØ7Ø                   YES -> WE ARE AT THE END
         LA    R5,1(R5)                ++COUNTER
         LA    R6,1(R6)                R6 -> NEXT BYTE IN BUFFER
         BCT   R4,LØØ6Ø                LOOP FOR THE # REMAINING BYTES
LØØ7Ø    DS    ØH
         LR    R6,R8                   R6 -> NEXT BYTE IN ENTITY
         LR    R7,R3                   R7 -> FIRST BYTE COMMAND BUFFER
         AR    R9,R5                   R9 = L(ENTITY)
         BCTR  R5,Ø                    --R5
         EX    R5,S_MVC                MOVE REMAINING OF DSNAME
*
* PREPARE FOR RACF CALL, DSN IS ALREADY FILLED IN
*
         OC    PROFILE(L_PROF),BLANKS  PROFILE TO UPPER CASE
         LA    R4,L_ENTITY             R2 = L(ENTITY BUFFER)
         STH   R4,L_BUF                STORE IN RACF ENTITY
         STH   R9,L_RES                STORE L(PROFILE)
         LA    R4,ENTITYX              R4 -> ENTITY
         LA    R5,RACWORK              R5 -> RACF WORK AREA
         MVC   D_RACR(L_S_RACR),S_RACR COPY STATIC CALL TO DYNAMIC AREA
         RACROUTE  REQUEST=AUTH,       REQUEST AUTHORITY               X
               ENTITYX=((R4),NONE),    FOR THIS PROFILE                X
               WORKA=(R5),             DETAILS ARE HERE                X
               RELEASE=2.6,            RACF RELEASE IS CONTROLLED      X
               MF=(E,D_RACR)           MACRO FORMAT
*
         LR    R4,RF                    R4 = SAF RETURN CODE
         LA    R5,D_RACR                R5 -> RACROUTE PARAMETER LIST
         USING SAFP,R5                  R5 ADDRESSES SAF PARAMETER LIST
         LTR   R4,R4                    SAF RETURN CODE OK?
         BNZ   LØØ8Ø                    NO --> MESSAGE WITH SAF AND
*                                              RACF ERROR CODES
         CLC   SAFPRRET,=F'Ø'           RACF RETURN CODE = Ø?
         BNE   LØØ8Ø                    NO --> MESSAGE WITH SAF AND
*                                              RACF ERROR CODES
         CLC   SAFPRREA,=F'Ø'           RACF REASON CODE = Ø?
         BE    LØØ9Ø                    YES -> ALL IS WELL
*
* SAF AND/OR RACF REASON CODES ARE NON-ZERO
* ISSUE MESSAGE MSGØ4 AND RETURN TO CALLER
*
```

```
LØØ8Ø    DS   ØH
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L'MSGØ4),MSGØ4      MESSAGE TO DYNAMIC BUFFER
         LA    RØ,L_S_PRE+L'MSGØ4      RØ = L(MESSAGE)
         LA    R1,TPUT_MSG             R1 -> MESSAGE
         TPUT  (R1),(RØ)               WRITE TO TERMINAL
*
* SAF RETURN CODE IS IN R4
*
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L_MSGØ4A),MSGØ4A    MESSAGE TO DYNAMIC BUFFER
         MVC   MSG+ORIGIN(4),=CL4'SAF' INDICATE ORIGINATOR, AND TYPE
         MVC   MSG+CODETYPE(6),=CL6'RETURN'
*
         ST    R4,SAFRC                KEEP FOR PRINTING
         LA    R8,SAFRC                R8 -> RETURN CODE
         LA    R9,MSG+CODE             R9 -> RECEIVE FIELD
         BAL   RE,PR_HEX               GO PRINT CODE
*
         LA    RØ,L_S_PRE+L_MSGØ4A     RØ = L(MESSAGE)
         LA    R1,TPUT_MSG             R1 -> MESSAGE
         TPUT  (R1),(RØ)               WRITE TO TERMINAL
*
* RACF RETURN CODE IS IN SAFPRRET
*
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L_MSGØ4A),MSGØ4A    MESSAGE TO DYNAMIC BUFFER
         MVC   MSG+ORIGIN(4),=CL4'RACF' INDICATE ORIGINATOR, AND TYPE
         MVC   MSG+CODETYPE(6),=CL6'RETURN'
*
         LA    R8,SAFPRRET             R8 -> RETURN CODE
         LA    R9,MSG+CODE             R9 -> RECEIVE FIELD
         BAL   RE,PR_HEX               GO PRINT CODE
*
         LA    RØ,L_S_PRE+L_MSGØ4A     RØ = L(MESSAGE)
         LA    R1,TPUT_MSG             R1 -> MESSAGE
         TPUT  (R1),(RØ)               WRITE TO TERMINAL
*
* RACF REASON CODE IS IN SAFPRREA
*
         MVC   MSG,BLANKS              BLANK OUT MESSAGE LINE
         MVC   MSG(L_MSGØ4A),MSGØ4A    MESSAGE TO DYNAMIC BUFFER
         MVC   MSG+ORIGIN(4),=CL4'RACF' INDICATE ORIGINATOR, AND TYPE
         MVC   MSG+CODETYPE(6),=CL6'REASON'
*
         LA    R8,SAFPRREA             R8 -> RETURN CODE
         LA    R9,MSG+CODE             R9 -> RECEIVE FIELD
         BAL   RE,PR_HEX               GO PRINT CODE
*
         LA    RØ,L_S_PRE+L_MSGØ4A     RØ = L(MESSAGE)
         LA    R1,TPUT_MSG              R1 -> MESSAGE
         TPUT  (R1),(RØ)                WRITE TO TERMINAL
         DROP  R5                       FORGET THE SAF PARAMETER LIST
         B     THE_END                  AND RETURN TO CALLER
         EJECT
*
* SAF RETURN AND RACF AND REASON CODES ARE ALL ZERO
*
LØØ9Ø    DS    ØH
         L    R1,PARMS
         CALL INDFXFER
         EJECT
*
* END OF PROCESSING
*
THE_END  DS    ØH                       MY ONLY FRIEND, THE END
         FREEMAIN RU,                   FREE THE WORK AREA             X
               A=(R2),                  ADDRESSED BY R2                 X
               LV=L_WORK                AND THIS LENGTH
         XR    RF,RF                    ALWAYS RC = Ø
         PR    .                        RETURN TO CALLER
         EJECT
*
* PR_HEX PROCEDURE
* AT INPUT R8 -> FULLWORD TO PRINT
*          R9 -> CL8 STRING TO RECEIVE
*
PR_HEX   DS    ØH
         STM   R2,RA,TEMPSAVE
         XR    R7,R7                    R7 = Ø
         LA    R4,4                     R4 = 4
LØ1ØØ    DS    ØH
         IC    R7,Ø(R8)                 R7 HAS FULL BYTE
         SRL   R7,4                     SHIFT OUT RIGHT HALF BYTE
         LA    R5,2
LØ11Ø    DS    ØH
         CH    R7,=H'1Ø'                R2 >= X'A'?
         BL    LØ12Ø                    YES -> CONTINUE
         AH    R7,=H'183'               NO --> ADD X'CØ'
         B     LØ13Ø                    GO CHECK NEXT BYTE
LØ12Ø    DS    ØH
         AH    R7,=H'24Ø'               ADD X'FØ'
LØ13Ø    DS    ØH
         STC   R7,Ø(R9)                 STORE IN RECEIVE FIELD
         LA    R9,1(R9)                 POINT 1 BYTE FURTHER
         IC    R7,Ø(R8)                 R7 HAS FULL BYTE
         N     R7,=X'ØØØØØØØF'          FIRST HALFBYTE IS GONE
         BCT   R5,LØ11Ø                 GO TRANSLATE THE SECOND HALF
         LA    R8,1(R8)                 R8 -> NEXT FULLWORD BYTE
         BCT   R4,LØ1ØØ
         LM    R2,R9,TEMPSAVE
         BR    RE
         EJECT
*
* CONSTANTS
*
S_PRE    DC    C'SE-IND$FILE: '         MESSAGE PREFIX
L_S_PRE  EQU   *-S_PRE                  L(MESSAGE PREFIX)
*
BLANKS   DC    131C' '                  BLANKS
         EJECT
*
* TERMINAL MESSAGES
*
MSGØØ    DC    C'MSGØØ COMMAND LENGTH TOO SMALL'
MSGØ1    DC    C'MSGØ1 NOT IND$FILE INVOCATION'
MSGØ2    DC    C'MSGØ2 GET NOR PUT REQUEST'
MSGØ3    DC    C'MSGØ3 DSNAME TOO SHORT'
MSGØ4    DC    C'MSGØ4 NON-ZERO SAF OR RACF RETURN/REASON CODE:'
MSGØ4A   DS    ØCL12Ø
         DC    C'MSGØ4A '
ORIGIN   EQU   *-MSGØ4A
         DC    CL5' '
CODETYPE EQU   *-MSGØ4A
         DC    CL6' '
         DC    C' CODE: '
CODE     EQU   *-MSGØ4A
         DC    CL8' '
L_MSGØ4A EQU   *-MSGØ4A
         EJECT
* STATIC RACF MACROS
S_RACR   RACROUTE  REQUEST=AUTH,        REQUEST AUTHORITY               X
               ENTITYX=(,NONE),         FOR THIS PROFILE                X
               MSGSUPP=YES,             SUPPRESS RACF INTERNAL WTO      X
               CLASS='FACILITY',        IN THIS CLASS                   X
               ATTR=READ,               FOR THIS ACCESS                 X
               RELEASE=2.6,             RACF RELEASE IS CONTROLLED      X
               WORKA=,                  DETAILS ARE HERE                X
               MF=L                     MACRO FORMAT
L_S_RACR EQU   *-S_RACR                 LENGTH STATIC RACROUTE
         EJECT
* EXECUTE TARGETS
S_MVC    MVC   Ø(Ø,R6),Ø(R7)            STATIC EXECUTE
         EJECT
* PARAMETER LISTS
         EJECT
* EQUATES
         EJECT
* LITERAL POOL
```

```
         LTORG
         EJECT
* DSECT'S
         PRINT GEN
WORKAREA DSECT                        OUR DYNAMIC WORKAREA
SAVEAREA DS    18F                    SAVEAREA
PARMS    DS    F                      POINTER TO PARAMETER LIST
SAFRC    DS    F                      SAF RETURN CODE
TEMPSAVE DS    18F
* TPUT MESSAGE AREA
TPUT_MSG DS    ØF
D_PRE    DS    CL(L_S_PRE)            PREFIX
MSG      DS    CL12Ø                  MESSAGE
L_MSG    EQU   *-TPUT_MSG             L(TOTAL MESSAGE)
* THE RACF FACILITY PROFILE IS OF THE FORM:
*  IND$FILE.<SUBCOMMAND>.<DSN>
*  EXAMPLE: IND$FILE.GET.SYS1.CMDLIB
ENTITYX  DS    ØF                     PROFILE
L_BUF    DS    H                      BUFFER LENGTH
L_RES    DS    H                      RESOURCE NAME LENGTH
PROFILE  DS    CL9                    FACILITY PROFILE PREFIX
SUBCOM   DS    CL4                    FACILITY PROFILE SUBCOMMAND
DSN      DS    CL44                   FACILITY PROFILE DSN
L_PROF   EQU   *-PROFILE              L(PROFILE)
L_ENTITY EQU   *-ENTITYX              L(ENTITY BUFFER)
*
D_RACR   DS    XL(L_S_RACR)           DYNAMIC RACF RACROUTE AREA
RACWORK  DS    1Ø24X                  RACF WORK AREA
L_WORK   EQU   *-WORKAREA             LENGTH OF ENTIRE WORKAREA
*
         ICHSAFP .                    SAF RACROUTE PARAMETER LIST
         IKJCPPL .                    COMMAND PROCESSOR PARAMETER LIST
         IKJUPT  .                    USER PROFILE TABLE
*
         END
./ ADD NAME=AMODE24
         MACRO
* THIS MACRO SETS THE AMODE OF YOUR PROGRAM TO 24
*
* THE CONTENT OF REGISTER 1 IS DESTROYED
&LABEL   AMODE24
         LA    R1,JED2&SYSNDX        R1 -> JED2XXXX
         N     R1,JED1&SYSNDX        SET FIRST BIT OFF
         BSM   RØ,R1                 BRANCH AND SET MODE
JED1&SYSNDX DS    ØF                 FULL WORD BOUNDARY FOR AND
         DC    X'7FFFFFFF'           SET FIRST BIT OFF
JED2&SYSNDX DS    ØH
*
         MEND
./ ADD NAME=EYECATCH
```

```
         MACRO
&LABEL   EYECATCH
         B     M&SYSNDX                SKIP BRANCH AROUND DCS
         DC    C'JAN DE DECKER -- JED:SP N.V.'
         DC    C' MODULE: '
SYSECT   DC    CL8'&SYSECT'            MODULE NAME
         DC    C' ASM DATE: '
         DC    CL8'&SYSDATE'           DATE
         DC    C' ASM TIME: '
         DC    CL8'&SYSTIME'           TIME
*
M&SYSNDX DS    ØH
         MEND
/*
//SYSUT2    DD  DSN=&&SRC,DISP=(NEW,PASS),
//              DCB=(DSORG=PO,LRECL=8Ø,BLKSIZE=616Ø),
//              UNIT=VIO,SPACE=(CYL,(1,1,1))
//*
//* COPY THE SOURCES TO THE DLIBS
//*
//S1        EXEC PGM=IEBCOPY
//SRC       DD  DISP=(OLD,PASS),DSN=&&SRC
//AMACLIB   DD  DISP=OLD,DSN=JEDPLX1.V2R7.AJEDMAC
//ASRCLIB   DD  DISP=OLD,DSN=JEDPLX1.V2R7.AJEDSRC
//SYSPRINT DD  SYSOUT=*
//SYSIN     DD  *
 COPY INDD=SRC,OUTDD=ASRCLIB
 SELECT M=(IND$FILX)
 COPY INDD=SRC,OUTDD=AMACLIB
 SELECT M=(AMODE24,EYECATCH)
/*
//*
//* PREPARATION: CREATE AN OBJECT DECK IN A TEMPORARY DATASET
//*
//S2        EXEC PGM=ASMA9Ø,PARM='OBJECT,NODECK,RENT'
//SYSLIB    DD  DSN=SYS1.MACLIB,DISP=SHR
//          DD  DSN=SYS1.MODGEN,DISP=SHR
//          DD  DSN=JEDSP.T.MACRO,DISP=SHR
//SYSUT1    DD  UNIT=VIO,SPACE=(CYL,(1,1))
//SYSUT2    DD  UNIT=VIO,SPACE=(CYL,(1,1))
//SYSUT3    DD  UNIT=VIO,SPACE=(CYL,(1,1))
//SYSPRINT DD  SYSOUT=*
//SYSPUNCH DD  DUMMY,SYSOUT=*
//SYSLIN    DD  DSN=&&OBJ(IND$FILX),DISP=(NEW,PASS),
//              DCB=(DSORG=PO,LRECL=8Ø,BLKSIZE=616Ø),
//              UNIT=VIO,SPACE=(CYL,(1,1,1))
//SYSIN     DD  DISP=(OLD,PASS),DSN=&&SRC(IND$FILX)
//*
//* PREPARATION: LINK THE OBJECT INTO THE DLIB
//*              EXPECT CONDITION CODE 4 (IEW2454W FOR INDFXFER)
```

```
//*
//S3       EXEC PGM=IEWL,PARM='NCAL'
//SYSLMOD  DD DISP=OLD,DSN=JEDPLX1.V2R7.AJEDLINK
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD  UNIT=VIO,SPACE=(CYL,(1,1))
//SYSLIN   DD DISP=(OLD,PASS),DSN=&&OBJ(IND$FILX)
//         DD *
 ENTRY IND$FILX
 NAME IND$FILX(R)
/*
//*
//* SMP/E STEP TO APPLY THE MACROS AND THE SOURCE
//*
//S4       EXEC PGM=GIMSMP
//SMPCSI   DD DISP=SHR,DSN=SMVS270S.GLOBAL.CSI
//JEDSRC  DD DISP=SHR,DSN=JEDPLX1.V2R7.JEDSRC
//AJEDSRC  DD DISP=SHR,DSN=JEDPLX1.V2R7.AJEDSRC
//AJEDLINK DD DISP=SHR,DSN=JEDPLX1.V2R7.AJEDLINK
//AJEDMAC  DD DISP=SHR,DSN=JEDPLX1.V2R7.AJEDMAC
//SYSLIB   DD DISP=SHR,DSN=SYS1.MACLIB
//MACLIB   DD DISP=SHR,DSN=SYS1.MACLIB
//CMDLIB   DD DISP=SHR,DSN=SYS1.CMDLIB
//SRC      DD DISP=(OLD,DELETE),DSN=&&SRC
//OBJ      DD DISP=(OLD,DELETE),DSN=&&OBJ
//SMPCNTL  DD  *
  SET BDY(MVST100) .
  UCLIN .
  REP LMOD(IND$FILE)
      NCAL
      REUS
      RENT
++LMODIN
      ENTRY    IND$FILX
      NAME     IND$FILE(R)
++ENDLMODIN  .
  ENDUCL .
 SET BDY(MVST100).                   /* -> TARGET ZONE            */
  RESTORE S(GUMOD60                  /* RESTORE AMODE24 MACRO     */
            GUMOD61                  /* EYECATCH MACRO            */
            GUMOD62) .               /* IND$FILX SOURCE           */
 RESETRC .
 SET BDY(GLOBAL).                    /* -> GLOBAL ZONE            */
  REJECT  S(GUMOD60                  /* REJECT AMODE24 MACRO      */
            GUMOD61                  /* EYECATCH MACRO            */
            GUMOD62) .               /* IND$FILX SOURCE           */
 RESETRC .
 SET BDY(GLOBAL).                    /* -> GLOBAL ZONE            */
  RECEIVE S(GUMOD60                  /* RECEIVE AMODE24 MACRO     */
            GUMOD61                  /* EYECATCH MACRO            */
            GUMOD62)                 /* IND$FILX SOURCE           */
```

```
          SYSMODS.
 SET BDY(MVST1ØØ).                     /* -> TARGET ZONE             */
  APPLY   S(GUMOD6Ø                    /* APPLY AMODE24 MACRO        */
            GUMOD61                    /* EYECATCH MACRO             */
            GUMOD62)                   /* IND$FILX SOURCE            */
            CHECK
            .
/*
//SMPPTFIN DD DATA,DLM=##
++USERMOD(GUMOD6Ø).
++VER(ZØ38) FMID(HBB66Ø6).
++MAC(AMODE24)                         /* NAME OF NEW MACRO          */
     DISTLIB(AJEDMAC)                  /* DDNAME OF DLIB             */
     TXLIB(SRC)                        /* SOURCE                     */
     SYSLIB(MACLIB) .                  /* DDNAME OF TARGET           */
++USERMOD(GUMOD61).
++VER(ZØ38) FMID(HBB66Ø6).
++MAC(EYECATCH)                        /* NAME OF NEW MACRO          */
     DISTLIB(AJEDMAC)                  /* DDNAME OF DLIB             */
     TXLIB(SRC)                        /* SOURCE                     */
     SYSLIB(MACLIB).                   /* DDNAME OF TARGET           */
++USERMOD(GUMOD62).
++VER(ZØ38) FMID(HFX1112).
++SRC(IND$FILX)                        /* NAME OF NEW SOURCE         */
     DISTLIB(AJEDSRC)                  /* DDNAME OF DLIB             */
     TXLIB(SRC)                        /* DDNAME OF INPUT SOURCE     */
     DISTMOD(AJEDOBJ) .                /* DLIB FOR OBJECT            */
++JCLIN .
//*
//* JCLIN TO FORCE COPY FROM THE DLIB TO THE TARGET LIBRARY
//* FOR THE ASSEMBLER SOURCE
//* ASSEMBLY AND LINK IS DONE AUTOMATICALLY BY SMP/E
//*
//SMPESØ   EXEC PGM=IEBCOPY
//SYSPRINT DD SYSOUT=*
//JEDSRC   DD DISP=SHR,DSN=JEDPLX1.V2R7.JEDSRC
//AJEDSRC  DD DISP=SHR,DSN=JEDPLX1.V2R7.AJEDSRC
//SYSIN    DD *
 COPY INDD=AJEDSRC,OUTDD=JEDSRC TYPE=SRC
 SELECT M=(IND$FILX)
/*
++MOD(IND$FILX)                        /* NAME OF NEW MODULE         */
     DISTLIB(ASPLINK)                  /* DDNAME OF DLIB             */
     TXLIB(OBJ)                        /* DDNAME OF INPUT OBJECT     */
     LMOD(IND$FILE) .                  /* RELATED LOADMODULE         */
##
//
```

## CONCLUSION

There exist many ways to transfer files between the MVS system and non-mainframe machines. This could result in a potential security exposure since RACF does not differentiate between the right to use a dataset in its own environment and the right to export or import it to another machine. Just think of the difficulties you come across to have an outside tape put into the robots in the average shop; or, the other way around, put some files on a cassette and then try to get the cassette mounted to copy them onto the mainframe. The same should be true for all portable media, including the hard disks of LAN-capable laptops. This article is a first step, it secures only a small part of the system – file transfer with IND$FILE. Nevertheless I hope it is the start of a discussion around the USE-level protection.

*Jan De Decker*
*Senior Systems Engineer JED:SP NV (Belgium)*

# RACF 101 – how RACF performs access checking

*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 the various ways a user can be granted (or denied) access to RACF resources.*

OK, so now you know there are profiles for datasets and general resources, and, within each profile, there is an access list that always tells you who can access that resource, right? Well, not quite.

Sooner or later, you will find out that, although access lists in resource profiles (dataset or general resource) are the primary means used by RACF to provide access to resources, they are not the only source. In fact, the access lists are only the tip of the iceberg. There are several other factors that influence RACF’s decision-making process. RACF will take all these other factors into account before making its decision whether to grant or deny access.

Not knowing these other issues, you may wonder how someone is getting access to a resource, even though they are not in the access list of the profile.

These additional considerations can generally be grouped into three categories – resource-related specifications, user attributes, and other assorted considerations.

## UNIVERSAL ACCESS

First, let’s look at resource-related specifications.

For all RACF resource profiles, you can specify something called Universal ACCess, or UACC for short. The possible values you can specify for UACC are – NONE, READ, UPDATE, CONTROL, or ALTER.

If a resource profile has a UACC specified other than NONE, then everybody gets the access specified in the UACC. For example, if the value is READ, everybody gets READ access, regardless of whether they are in the access list or not.

If the UACC of a profile is UPDATE, and there is also an access list entry that gives READ to a user, then the user gets only READ. If, however, the UACC value is READ and the access list specifies UPDATE for a user, then the user gets UPDATE access. In other words, if some access is explicitly specified for the user, then that is what they will get.

To see the UACC on a profile, enter one of the list commands. For dataset profiles:

```
LISTDSD DA('dataset.name') ALL
```

For general resource profiles:

```
RLIST class.name profile.name ALL
```

Here is the sample output from the LISTDSD command (partial output). It shows that the UACC is NONE:

```
LISTDSD DA('ABCD.**')

INFORMATION FOR DATASET ABCD.** (G)

LEVEL  OWNER    UNIVERSAL ACCESS  WARNING   ERASE
----   -------  ----------------  --------- -----------
 ØØ    USER999  NONE              NO        NO
```

## GENERIC USERID

A profile may specify a generic (wildcard) entry. You will see this as '*' in the access list. This is another method of granting access to everybody, and its interpretation is similar to that of Universal Access described above.

There is a slight difference between granting access to everybody via a generic userid and doing the same thing via UACC. The difference is that UACC applies to 'undefined' RACF userids and 'defined' userids, whereas the generic id method applies only to 'defined' RACF users. It is OK if, as a beginner, you don't understand this subtle difference.

For now, just remember that if a userid is not explicitly mentioned in an access list, but there is a generic id present, then that user will get the access specified in the generic entry.

To see whether there is a generic id in an access list, enter one of the list commands. For dataset profiles:

```
LISTDSD DA('dataset.name') ALL
```

For general resource profiles:

```
RLIST class.name profile.name ALL
```

Here is the sample output from the LISTDSD command (partial output). It shows that there is a generic id present in the access list, and it gives READ to everybody.

```
INFORMATION FOR DATASET ACCOUNT.** (G)
. . .
. . .
```

```
ID                   ACCESS
----                 -------------
*                    READ
USERnnn              UPDATE
```

## WARN MODE

If a profile is in WARN mode, then FULL access is granted to everybody!

Of course, if the user is not in the access list, and does not qualify to get the access through other means, then the access attempt is logged, and a warning message is issued to the user. But still, you should be careful about profiles in WARN mode, since they grant ALL access, including ALTER access, to ALL users.

To see whether a profile is in WARN mode, enter one of the list commands. For dataset profiles:

```
LISTDSD DA('dataset.name') ALL
```

For general resource profiles:

```
RLIST class.name profile.name ALL
```

Here is the sample output from the LISTDSD command (partial output). It shows that the profile is in WARN mode (WARNING = YES):

```
INFORMATION FOR DATASET PROJ1.** (G)

LEVEL  OWNER    UNIVERSAL ACCESS  WARNING     ERASE
-----  -------  ----------------- ----------  ------------
 ØØ    MGRØØØ1   NONE             YES         NO
```

## UNPROTECTED RESOURCES

If your installation has not told RACF to protect all your data, then whatever is not protected by RACF is available to everybody. In other words, if there is no profile for a resource, then that resource is available to everybody.

To see whether your installation is protecting all its data, enter the command:

```
SETROPTS LIST
```

And look for the PROTECT ALL option specified at your installation. If it says that PROTECT-ALL FAIL option is active, then you are protecting all your data.

Here is the partial output of the SETROPTS LIST command, showing that this installation is protecting all its resources.

```
PROTECT-ALL IS ACTIVE, CURRENT OPTIONS:
   PROTECT-ALL FAIL OPTION IS IN EFFECT
```

That covers resource-related specifications. Now let us look at user-related attributes that can also determine a user’s ability to access a resource.

## OPERATIONS ATTRIBUTE

If a user has the OPERATIONS attribute, then that person can access any dataset unless they are explicitly prohibited by an entry in the access list! Also, if some of your resource classes are set up to ‘honour’ the OPERATIONS attribute, then all resources in those classes are also available to this user!

Of course, RACF will log this action after granting the access.

To see whether a person has the OPERATIONS attribute, enter the LISTUSER command:

```
LISTUSER USRSUPR
```

Here is partial output of this command, showing that the user has the OPERATIONS attribute.

```
USER=USRSUPR  NAME= SUPER USER      OWNER=GRPOWN1   CREATED=Ø3.Ø87
 DEFAULT-GROUP=ABCD PASSDATE=Ø4.13Ø  PASS-INTERVAL= 3Ø
 ATTRIBUTES=OPERATIONS
```

## PRIVILEGED OR TRUSTED ATTRIBUTE

Started tasks generally have one of these special attributes (PRIVILEGED or TRUSTED). If they do, then they have access to all resources, datasets etc, at your installation. The difference between the two is that the TRUSTED attribute will log the access granted, whereas the PRIVILEGED attribute will not.

To see whether a started task has the TRUSTED or PRIVILEGED attribute, use the RLIST command:

```
RLIST STARTED JES2.* STDATA
```

Here is a partial listing of the output from this command. The started task JES2 has the TRUSTED attribute (TRUSTED = YES):

```
CLASS      NAME
-------    --------
STARTED    JES2.* (G)
. . .
. . .
STDATA INFORMATION
------------------
USER= $JES2
GROUP= STCGRP
TRUSTED= YES
PRIVILEGED= NO
TRACE= NO
```

## SPECIAL ATTRIBUTE

Users having the SPECIAL attribute do not directly have access to resources, but be aware that they are capable of giving themselves access to any resource by changing the access lists.

To see whether a person has the SPECIAL attribute, enter the LISTUSER command:

```
LISTUSER USRØØØ1
```

Here is partial output of this command, showing that the user has the SPECIAL attribute (ATTRIBUTES = SPECIAL):

```
USER=USRØØØ1  NAME= SPECIAL USER      OWNER=GRPOWN1   CREATED=Ø3.Ø87
 DEFAULT-GROUP=ABCD PASSDATE=Ø4.13Ø  PASS-INTERVAL= 3Ø
 ATTRIBUTES=SPECIAL
```

## RESTRICTED ATTRIBUTE

RESTRICTED is one attribute that prevents a user from having access that they would have otherwise had. In other words, a user having this attribute will not get any access allowed by UACC or a generic userid mentioned above. In addition, they will not get the access specified in the global access checking table mentioned below.

The only access they will get is where they are explicitly mentioned by userid in the access list (or if they are connected to a group that is mentioned in the access list).

To see whether a person has the RESTRICTED attribute, enter the LISTUSER command:

```
LISTUSER GUESTØ1
```

Here is partial output from this command, showing that the user has the RESTRICTED attribute (ATTRIBUTES = RESTRICTED):

```
USER=GUESTØ1  NAME= GUEST USER      OWNER=GRPOWN1   CREATED=Ø3.Ø87
 DEFAULT-GROUP=ABCD PASSDATE=Ø4.13Ø  PASS-INTERVAL= 3Ø
 ATTRIBUTES=RESTRICTED
```

## OTHER

Finally, we will look at some other considerations that are not profile-related or user-related, but yet can play a role in determining access. This last category includes the global access checking table and exits.

### Global access checking table

The global access checking table is a method RACF uses to allow access to commonly-used resources.

This is done for performance reasons only. If a matching entry for a profile is found in the global access checking table, then access is allowed, and the actual profile is not even checked. However, if this table does not allow access (that is, there is no matching entry), then the resource profile in question is still checked to see whether access should be allowed.

The global access checking table is most often associated with datasets; however, any other resource can also be specified in this table.

To see entries in the global access checking table at your installation, enter the sample command:

```
RLIST GLOBAL DATASET
```

Here is a partial listing of the output:

```
CLASS      NAME
------     --------
GLOBAL     DATASET

MEMBER CLASS NAME
------ ----- ----
GMBR

RESOURCES IN GROUP
--------------------------------
&RACUID.**/ALTER (G)
. . .
. . .
ISPF.**/READ (G)
```

The first entry, &RACUID.**/ALTER (G), is interesting. It means: allow ALTER access to all datasets to all userids where the userid matches the first qualifier of the dataset.

For example, USER01 will have FULL access to all datasets that start with USER01.

### Exits

Lastly, we come to a topic that a new RACF administrator should not be too concerned about.

Here we will mention only that exits in RACF can override the access checking criteria described above! This does not mean that installations use exits for this purpose. However, be aware that it can be done.

## SUMMARY

In this column we saw that RACF access checking is not straightforward, and rightly so. As your knowledge of RACF increases, you will appreciate that although complications discussed here sometimes make RACF difficult to understand, it is precisely these things that make RACF so powerful and versatile.

*Dinesh Dattani (dinesh123@rogers.com)*
*Independent Consultant (Canada)*

## A RACF course review

In preparation for Vanguard's recent RACF/Security conference, I took two Web-based training (WBT) courses to give me the right background to understand the kind of RACF sessions I wanted to attend. I was impressed with the level of information included. My boss was impressed with the price, or, more accurately, the cost/benefit – since cheap courses are a waste of money if they are not effective.

### THE GOAL

My goal was to find courses I could take in modules, mostly at home, but also from my cubicle at work. I didn't want to VPN from home into Terminal Server at work because Terminal Server's 256-colour display can be hard on the eyes.

Computer-Based Training (CBT) was not considered because of the hassles involved in getting software installed on my desktop at work, and the need for two copies – one at home and one at work.

### THE CHOICE

Based strictly on the course content, I chose DataTrain's *How to Use RACF*. It was six times the cost of the Serebra SMP/E course that I took two months earlier, but still extremely cheap when compared with classroom courses. Though both courses did the job, DataTrain courses are just built better.

The Serebra course strongly recommended that I set my screen resolution to 800x600 and drop the number of colours displayed. DataTrain allows you to size your course window as you see fit. Yes, even full screen on my 1600x1200 20-inch LCD monitor at work.

Many of Serebra's questions, especially the fill-in-the-blanks ones, accept only one of several right answers; a few are just plain wrong. Not that DataTrain is perfect in this regard, but they do stick to multiple-choice questions and tell you how many right answers there are for each question. And DataTrain sticks to mainstream questions: the most important topics covered in the course.

The course text is pretty clean in both, with almost all errors occurring in the examples. Serebra has the higher error rate, though DataTrain varies: the RACF course was better than one on Unix System Services (USS).

Serebra overstates the course duration, while DataTrain seriously understates. Finally, neither provides the handouts you get in classroom courses. It was time-consuming, but I wrote my own, for future reference.

## FINDING THEM

http://www.datatrain.net/distrib.html lists DataTrain distributors worldwide. Each distributor packages and prices in a different way. I paid US$199 per course for a single student for three months.

http://learn.serebra.com takes you directly to Serebra's on-line store. I paid US$30 per course for an unlimited number of students for three months.

Another reference, although over three years old now, is *RACF Update*, issue 24 (May 2001, pp. 33–59, http://

www.xephon.com/index/journals/more/RACF), which contains a round-up of all known relevant security training available at the time.

*Jon E Pearkins*
*Adiant (Canada)*

# RACF news

Eurekify has announced Sage Discovery and Audit (DNA) for RACF. With Sage for RACF, administrators can analyse and audit existing privileges and groups, as well as create a new Role-Based Access Control (RBAC) privileges provisioning structure.

As soon as data is imported into Sage, system administrators can see who has access to what, and who else has similar access. Sage can then identify groups of users and/or collections of groups that share a characteristic set of privileges. Role definitions can then be stored as RACF groups, or into any other Identity Store of choice, including an Identity Management platform, Enterprise Directory, etc.

For further information contact:
Eurekify, Atidim Science Park, Building 3, Fourth Floor (Einav), POB 58118, Tel Aviv 61580, Israel.
Tel: +972 3 644 1180.
URL: http://www.eurekify.com/sage_racf.htm.

* * *

Blockade Systems has released ManageID Enterprise Suite for Microsoft Identity Integration Server 2003, Enterprise Edition. This suite encompasses Management Agents for z/OS and OS/390 environments (RACF, plus ACF2 and Top Secret), which integrate with MIIS.

The product provides real-time event detection, which includes the ability to both detect and apply account creations or account deletions and account attribute or state changes on z/OS security environments.

For further information contact:
Blockade Systems, 2200 Yonge Street, Suite 1300, Toronto, Ontario, Canada, M4S 2C6.
Tel: (416) 482 8400.
URL: http://www.blockade.com/products/miis.html.

* * *

Vanguard Integrity Professionals has announced Version 5.2 of Vanguard Security Solutions (VSS 5.2).

Vanguard Administrator provides a range of security administration, data mining, and reporting tools to simplify and enhance administration of the IBM Security Server (RACF). Vanguard SecurityCenter is the first Windows-based GUI for IBM Security Server and DB2 security administration. Users can interface with RACF. Vanguard Registration Manager provides the reporting and maintenance capabilities necessary to effectively manage the userid/platform matrix that is created by ez/SignOn. This includes the ability for users to recover map profiles, delete map profiles associated with a RACF user ID, and report by platform, username, and RACF user ID definitions.

For further information contact:
Vanguard Integrity Professionals, 3035 East Patrick Lane, Suite 11, Las Vegas, NV 89120-3478, USA.
Tel: (702) 794 0014.
URL: http://www.go2vanguard.com/software_solutions/new_in_5.2/.

* * *