

RAKF User's Guide

A Security System for MVS 3.8J

Release: 1.2.0 with additional PTF's and auxiliary utilities.

October 2021, updated in December 2024

```
*****
*                                     *
*   © Craig J. Yasuna               *
*   See Appendix for detailed Copyright Information      *
*                                     *
*****
```

```
*****
*                                     *
*   RAKF is based on the ESG Security System           *
*   written by Craig J. Yasuna          (Mar 1991)    *
*   adapted to MVS 3.8J: A. Philip Dickinson (Aug 2005) *
*                           Phil Roberts     (Apr 2011)    *
*                           Jürgen Winkelmann (Apr 2011)  *
*                                     *
*****
```

```
*****
*                                     *
* This document applies to RAKF Version 1 Release 2 Modification 0  *
* which includes PTFs RRKF001, RRKF002, RRKF003, RRKF004, RRKF005,  *
* RRKF006, RRKF007 and RRKF008 are already installed.            *
*                                     *
*****
```

```
*****
*                                     *
* The information in this User's Guide is adapted from the original  *
* ESG Security System's documentation as prepared by Sam Golob in   *
* 1991.                         *
*                                     *
*****
```

Contents

Introduction	3
Installation	4
PTF Installation.....	6
Auxiliary Utilities	7
Installation of the Auxiliary Utilities	8
Customization	9
Users and Profiles Tables.....	10
Users Table.....	10
PROFILES table.....	11
Batch Jobs and Started Task Considerations	16
Security Authorization Facility (SAF) Support in MVS 3.8J	17
General Resource Classes.....	17
Dataset Protection using the RACF indicator.	19
Turning the RACF Indicator On or Off	19
Setting or Clearing the RACF Indicator of a Single Non-VSAM Dataset	19
Setting or Clearing the RACF Indicator of a Single Catalog or VSAM Object....	20
Setting or Clearing the RACF Indicator of All Eligible Non-VSAM Datasets	20
Setting or Clearing the RACF Indicator of All Catalogs and VSAM Objects	21
Auxiliary Utilities	21
Version History	22
ESG Security System (March 1991)	22
RAKF "RAcK oF" Security System (August 2005).....	22
RAKF 1.2.0 (April 2011)	22
RAKF 1.2.0 PTF Summary	23
RAKF 1.2.6 (October 2021).....	23
Appendix A - Generating your own release	25
Appendix B - Add users to UADS.....	25
Appendix C - SORT job.....	25
Appendix D - USERMOD RAK0001.....	27
Appendix E - RACIND Example	30

Introduction

The RAKF Security System is a RACF™-like MVS System Authorization Facility (SAF). RACF™ Version 1.7 facilities are emulated, except for the RACF™ database. Two tables, the users and the profiles table, are kept in storage. The actual security verifications are made by ICHSFR00, using these two tables. The formats of these users and profiles tables are compatible with RACF™ database entry data. Protection is achieved by routing all operating systems or vendor product security calls (including RACDEF, RACINIT, and RACHECK) through the ICHSFR00 RACROUTE interface. ICHSFR00 contains the real verification code. The RAKF Security System is designed to force "one point of handling" for all security calls. ICHSFR00 processes the various kinds of security calls in a standard way that is (mostly) documented by IBM. ICHSFR00 refers to the installation-coded user and profile in-core tables, to make its judgments. These in-core user and resource tables are each reloadable at any time by the execution of their special started tasks, this can be done in one of two ways:

From the master console issue, the command `s RAKFPROF` and `s RAKFUSER` or in the hercules console `/S RAKFPROF` and `/S RAKFUSER`

From JCL:

```
//RAKFUPDT JOB (RAKF), 'UPDATE RAKF', CLASS=A, MSGCLASS=A  
//RAKFUSER EXEC RAKFUSER  
//RAKFPROF EXEC RAKFPROF
```

When MVS is informed that "security" is present on the system, all previously defined passwords, including VSAM passwords, are ignored. Any password protection must be reinstated by the security system. Evidently, the MVS designers wanted any passwords designated by security, not to be interfered with by any other password mechanisms in MVS.

Additionally, it should be noted that the ESG Security System, which RAKF is based on, had been developed in 1991 for MVS versions released 5 to 10 years later than RAKF's target MVS 3.8j. Consequently, RAKF supports security features that are used by MVS 3.8j differently than by later MVS versions or are not used at all and thus will not work as expected.

The way that protection will work on your system is entirely up to your control. Protection depends completely on the way you code the users and profiles tables. RAKF utilizes these tables for its protection decisions, instead of using a RACF™ database. Minimal tables to create a state equivalent to MVS without an active security product have been provided. Starting from these you should gradually move forward to tighten security as needed on your system.

Installation

Installation is only necessary if you run a RYO MVS 3.8J system. RAKF is already installed on the MVS-Turnkey 4- and MVS-Turnkey 5 systems.

Installation is performed through the standard SMP RECEIVE, APPLY and ACCEPT procedure. The process has been tested with SMP level 04.48 which is the level provided by the Tur(n)key 3 MVS system. This presumably is equivalent to having PTFs UR13349, UR15994, UR17644 and UR19590 applied, which can be found on the allptfs.het tape of the Tur(n)key 3 MVS distribution CD. Tests have shown that APPLY processing doesn't work reliably at lower levels: For example at level 04.44 APPLY doesn't assemble and link all modules. So before starting please update your SMP as close to level 04.48 as possible.

1. Download File 850 from <https://cbttape.org>. Locate job RAKF12. It is in member \$RAKF12 of the received PDS.
Adapt the name of the installation job control library (//INSTJCL DD statement) in job RAKF12 to your needs and submit the job.
2. Edit jobs A@PREP, C@APPLY and D@ACCPT in the installation job control library:
 - change the //ASAMPLIB DD and //SAMPLIB DD statements (only in job A@PREP) to read:

```
//ASAMPLIB DD DISP=(,CATLG),DSN=HLQ.ASAMPLIB,VOL=SER=dddddd,  
//           UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=19040),  
//           SPACE=(TRK,(120,40,10))  
//SAMPLIB   DD DISP=(,CATLG),DSN=HLQ.SAMPLIB,VOL=SER=ssssss,  
//           UNIT=SYSDA,DCB=(RECFM=FB,LRECL=80,BLKSIZE=19040),  
//           SPACE=(TRK,(120,40,10))
```

The changes to the job as distributed on the product tape are the space allocation values printed in bold above. After having made this change, you can ignore the special installation instruction in PTF RRKF006 requesting these two libraries to be reallocated due to larger space requirements.
 - change all occurrences of HLQ to the desired high level qualifier for the RAKF libraries, e.g. RAKF, RAKF.V1R2M0, etc. Please note that HLQs where HLQ.SAMPLIB, HLQ.ASAMPLIB, HLQ.APROCLIB, HLQ.APARMLIB HLQ.MACLIB, HLQ.AMACLIB, HLQ.SRCLIB or HLQ.ASRCLIB already exist are not recommended.
 - change all occurrences of ddddd to the desired volume for the RAKF distribution libraries.
 - change all occurrences of sssss to the desired volume for the RAKF target libraries.
 - change all occurrences of tttt to the device type and all occurrences of rrrrrr to the volume of the sysres device where SYS1.LPALIB and

SYS1.LINKLIB of the system reside on which RAKF is to be installed (If you are running TK3 MVS and install on your current system the device type is 3350 and the volume is MVSRES).

- Verify that in jobs C@APPLY and D@ACCPCT the sequence of the SYSLIB and the target and distribution library DD statements matches your SMPAPP and SMPACC procedures. The jobs have been tested with the SMP procedures as defined in the Tur(n)key 3 MVS system. If the sequence of DD statements in the SMP procedures in your system doesn't match the sequence of the overriding DD statements in C@APPLY and D@ACCPCT RAKF elements may erroneously be placed in SYS1.ASAMPLIB, SYS1.SAMPLIB, SYS1.APROCLIB, SYS1.APARMLIB, SYS1.AMACLIB or SYS1.MACLIB instead of their HLQ.xxxxLIB counterparts.
3. Submit job A@PREP. This job performs the following actions to prepare your system for RAKF 1.2.0 installation:
 - Delete MODs and LMODs that are defined in MVS 3.8j as placeholders for RACF modules from SMP's target and distribution zones.
 - Delete the placeholder modules as well as potentially installed pre RAKF 1.2.0 modules from SYS1.LINKLIB and SYS1.LPALIB.
 - Allocate RAKF 1.2.0 target and distribution libraries.
 4. Receive RAKF by submitting job B@RECV. Be prepared to mount the RAKF distribution tape rakf12.aws after submitting the job.
 5. Apply RAKF by submitting job C@APPLY. This job places all RAKF components in their designated target libraries and assembles and link-edits the SVCs (ICHRIN00) and the SAF router (ICHESFR00) to SYS1.LPALIB and the support modules (ICHSEC00, RAKFUSER, RAKFPROF and RAKFPWUP) to SYS1.LINKLIB. These six load modules constitute the complete code provided by RAKF, making up an extremely small footprint as compared with RACF™.
 6. Accept RAKF by submitting job D@ACCPCT. This step can be skipped if you don't want to accept the RAKF sysmod now. It is, however, strongly recommended to accept it as only then will it be possible to "test drive" updates that might be published later and to return to the previous state in case of problems.
 7. Perform an IPL CLPA of your system. The system will come up with all RAKF elements installed but without activating RAKF. Successful installation can be verified from the RAKF banner being displayed at the beginning of the IPL and by checking the system's SVC table for example using menu option T in IMON and scrolling forward to the RACF SVCs (130 - 133) which should show entries like this:

130 82 00EAEB30 3/4	RACHECK	-RACF IGC00130 RAKF mm/dd/yy
131 83 00EAEB52 3/4 YES	RACINIT	-RACF IGC0013A RAKF mm/dd/yy

```
132 84 00EAEB74 3/4 YES      RACLIST      -RACF ICHRIN00 DUMMY mm/dd/yy  
133 85 00EAEB9A 3/4 YES      RACDEF       -RACF IGC0013C RAKF   mm/dd/yy  
where mm/dd/yy is RAKF's installation date.
```

8. Check for any applicable RAKF PTFs and install these following the steps in chapter “PTF Installation” on page 6. PTFs are available in folder files/RAKF/PTFs of the H390-MVS Groups.io group and in CBT file 850.
9. Chapter “Installation of the Auxiliary Utilities” on page 8 outlines the installation of some utilities that are not part of RAKF. They are distributed with RAKF because they are required to efficiently handle the RACF indicator of datasets, catalogs and VSAM objects, which is crucial to achieve a reliable dataset protection. It is recommended to install these utilities after having installed all PTFs and before continuing to customize RAKF.
10. Consult chapter “CUSTOMIZATION” on page 8 for information on how to customize and activate RAKF.

PTF Installation

To install a RAKF PTF pretty much the same procedure as for RAKF base installation is used:

- Upload the PTF to a sequential file or PDS member (LRECL=80,RECFM=FB) on your MVS system.
- Customize job B@RECV from your HLQ.SAMPLIB (HLQ = highlevel qualifier for RAKF libraries as chosen at RAKF installation time): Have the //SMPPTFIN DD statement point to the dataset or PDS member where you placed the PTF and change the RECEIVE command to read:
RECEIVE S(PTF-ID).

PTF-ID is the ID of the PTF, for example RRKF001. Submit the job. Return code 0 is expected.

- Customize job C@APPLY as described in chapter “Installation” on page 4 (if not already done) and change the APPLY command to read:
APPLY S(PTF-ID) DIS(WRITE) .
Check the comment section at the beginning of the PTF (the “Cover Letter”) for any special installation instructions for APPLYing the PTF and perform these as appropriate. Then submit the APPLY job. Return code 0 is expected.
- Test RAKF operation with the PTF in place to ensure that no function you use got broken.
 - When RAKF with the applied PTF operates as it is supposed to, accept the PTF: Customize job D@ACCEPT as described in chapter “Installation” on page 4 (if not already done) and change the ACCEPT command to read:
ACCEPT S(PTF-ID) DIS(WRITE) .

Check the Cover Letter for any special installation instructions for ACCEPTing the PTF and perform these as appropriate. Then submit the ACCEPT job. Return code 0 is expected.

- When RAKF with the applied PTF exhibits problems, restore the PTF: Customize job D@ACCEPT as described in chapter “Installation” on page 4 (if not already done) and change the ACCEPT command to read:

RESTORE S(PTF-ID) DIS(WRITE).

Submit the job. Return code 0 is expected. After running the RESTORE command your RAKF system is restored to the state before the PTF was applied.

Note: If there are no special installation instructions that cannot be satisfied at the same time, multiple PTFs can be RECEIVED, APPLIED or ACCEPTED at once in a single SMP run. Although some of the PTFs available as of the publication date of this User’s Guide carry special installation instructions, none of them are in conflict with each other, i.e. all PTFS currently available can be handled together by a single RECEIVE, APPLY and ACCEPT run.

Special installation instructions frequently involve receiving a PDS packaged in XMI format. If the RCV370 utility isn’t installed on your system you cannot use the sample jobs provided to receive these datasets. In this case, please use the Hercules dasdload utility instead.

Auxiliary Utilities

Chapter “Turning the RACF Indicator on or Off” on page 19, outlines procedures for setting or clearing the RACF indicator of datasets, catalogs and VSAM objects, which is crucial to providing reliable dataset protection. These procedures make use of a set of utilities that are neither part of the RAKF product nor of native MVS 3.8J. This package is available in member AUXUTILS in RAKF.SAMPLIB. In MVS-Turnkey 5, these packages are available in TK5.ASM and TK5.CNTL.

- **VTOC:** A TSO command to list DSCB information for selected datasets. The version used here is an adaptation of CBT File 112, created by Phil Roberts. The changed source, which contains the RACF indicator, and the UNIT field is resident in MVS Turnkey 5 in library TK5.ASM, member VTOC.
- **CDSCB:** A TSO command to modify the DSCB of a dataset. The source of this command is available in library TK5.ASM member CDSCB. This command should be assembled, zapped and linked with job CDSCB\$ in TK5.CNTL. The zap in this job enables the CDSCB command to run in the batch. Remove the zap if running in the batch is not wanted.
- **SVC 244:** This SVC was originally provided with the Tur(n)key 3 MVS system. In MVS-Turnkey 5, the original SVC244 has been replaced by a secured version of SVC244.

Installation of the Auxiliary Utilities

Perform the following steps to install the auxiliary utilities in MVS Turnkey 5. This is an optional step, since these utilities are already installed in MVS-Turnkey 5

1. Run job VTOC in TK5.ASM to install the TSO command processor VTOC.
2. Run job CDSCB\$ in TK5.CNTL to install the TSO command processor CDSCB.
3. Run job SVC244\$ in TK5.CNTL to install the secured version of SVC244.
4. Define the SVC244 profile in the FACILITY class and authorize the users that should have access to SVC 244. The following RAKF profiles table entries are provided as an example on how to do this:

FACILITYSVC244		NONE
FACILITYSVC244	ADMIN	READ
FACILITYSVC244		STCGROUPREAD

This example defines a universal access of “none” (i.e. nobody has access unless authorized explicitly) and explicitly authorizes user ADMIN (or all members of group ADMIN) and all started tasks (the STCGROUP).

If the RAKF profiles table has not yet been defined on your system, then postpone the above step until you have created the profiles table according to step Users and Profile Tables of the customization procedure on page 10.

5. Perform an IPL CLPA at your earliest convenience to activate the secured SVC 244.

Customization

The RAKF security configuration is defined by the RAKF users table and the RAKF profiles table, which are text files maintained by the security administrator to define the system's users, groups, resources and their protection. These tables are members of the partitioned data set SYS1.SECURE.CNTL:

USERS: The RAKF users table

PROFILES: The RAKF profiles table

 Warning: each TSO user still needs a UADS entry to define the TSO attributes and authorizations. These are not covered by the RAKF users table. See appendix B for an example job to add users on MVSCE and Jay Moseley sysgen.

During the process of adapting the profiles and users tables to meet your security requirements special attention to the protection of the `SYS1.SECURE.CNTL` and `SYS1.SECURE.PWUP` should be paid:

`SYS1.SECURE.CNTL`: started tasks and the user(s) and/or group(s) responsible for RAKF administration need UPDATE access to this dataset.

`SYS1.SECURE.PWUP`: started tasks need UPDATE access to this dataset.

Both datasets contain clear text passwords and therefore users should not be allowed any access to them. An easy way to protect these datasets is to define a dataset profile `SYS1.SECURE.*` with universal access NONE and selectively allow the RAKF administrator user(s)/group(s) UPDATE access to this profile. If the standard setup is used started tasks have operations authority and thus don't need to be explicitly allowed.

Users and Profiles Tables

In order for the profiles tables to be valid, it must be in sort order. See Appendix C for MVTSORT jcl which can sort tables. Sort errors will inhibit initialization of the tables and will generate error messages.

*Note that the tables may contain comment lines starting with an asterisk () . **

In the profiles table, DEFAULT, or "UNIVERSAL" access for any facility or dataset must be coded before any specific access is coded. You code a universal access entry in the profiles table by leaving the user group field blank. Then you code other entries for the same facility, specifying different settings for each user group that will have special access (or denial of access) to that facility.

Generally, only those features included in "IBM RACF™ Version 1.7" have been emulated. RAKF uses the decision logic of the ESG Security System without modifications. The author of the ESG security system has referred to two RACF™ manuals during their planning. It is important for all users of RAKF to obtain these manuals also. These are needed in the security administration, which will be ongoing. The two manuals are:

Manual	Reference	Description
SPL RACF	SC28-1343-2	Referred to for information how to write macros and return codes.
RACF Administrator's Guide	SC28-1340	Has an overview of profiles that should be used, and their structure.

You should try to find versions of these manuals being as close as possible to RACF™ Version 1.7 because later manuals describe features not available with RAKF and/or MVS 3.8j.

The profiles table is read from the bottom upwards which is the reason why the source table needs to be in ascending sort order: Reading them in reverse order ensures to find the most significant hit for a resource or user search before any less significant hits, so tables search will always be stopped upon the first hit.

Users Table

Below is an example `USERS` table:

HMVS01	ADMIN	*CUL8TR	Y
HMVS01	RAKFADM	CUL8TR	Y
HMVS02	USER	PASS4U	N
IBMUSER	ADMIN	*SYS1	Y
IBMUSER	RAKFADM	*SYS1	Y

This controls user access. Each line is a user, group, password, and operations as described below:

Column	Description
1 - 8	USERID
10 - 17	User Group (Installation defined)
18	Asterisk '*' multiple user groups exist for this userid.
19 - 26	Password
28	Operations Authority (Y or N). Always allow access ¹
31 - 50	Comment field (used by "IBM RACF").

1 Unless explicitly denied access via a rule

Each line can be broken down as:

```
UUUUUUUU GGGGGGGGSPPPPPPP A CCCCCCCCCCCCCCCCCCCC
Username Group *Password 0 Comment
```

UserID's in the users table that were set up with multiple group entries will get the highest authority for all protected objects in all the groups. As a practical example, multiple groups are used for managers who oversee the work of several programming groups. The multiple group arrangement gives these managers access to everything done by all the groups under them.

PROFILES table

Below is an example PROFILES table:

```
*****
*
* RAKF profile definitions for TK4-
*
*****
*
* rdefine dasdvol * uacc(read)
* permit * cl(dasdvol) id(admin,stcgroup) access(alter)
*
DASDVOL *                                     READ
DASDVOL *                                     ADMIN   ALTER
DASDVOL *                                     STCGROUPALTER
*
* addsd '.*.*'                               uacc(read)
* permit '.*.*' id(admin,stcgroup)          access(alter)
*
DATASET *                                     READ
DATASET *                                     ADMIN   ALTER
DATASET *                                     STCGROUPALTER
*
* addsd 'IPCS.DATA.SET.DIRECTRY'             uacc(update)
* addsd 'IPCS.PROBLEM.DIRECTRY'              uacc(update)
```

```

*
DATASET IPCS.DATA.SET.DIRECTRY                      UPDATE
DATASET IPCS.PROBLEM.DIRECTRY                      UPDATE
*
* addsd 'RPFPROF.CLUSTER'             uacc(update)
*
DATASET RPFPROF.CLUSTER                           UPDATE
*
* addsd 'SYS1.BRODCAST'                  uacc(update)
*
DATASET SYS1.BRODCAST                            UPDATE
*
* addsd 'SYS1.PAGE.*'                     uacc(none)
*
DATASET SYS1.PAGE.*                             NONE
*
* addsd 'SYS1.SECURE.*'                   uacc(none)
* permit 'SYS1.SECURE.*' id(admin)    access(none)
* permit 'SYS1.SECURE.*' id(rakfadm)   access(update)
*
DATASET SYS1.SECURE.*                           NONE
DATASET SYS1.SECURE.*                           ADMIN  NONE
DATASET SYS1.SECURE.*                           RAKFADM UPDATE
*
* addsd 'SYS1.STGINDEX'                    uacc(none)
*
DATASET SYS1.STGINDEX                          NONE
*
* addsd 'SYS1.UCAT.TSO'                   uacc(update)
*
DATASET SYS1.UCAT.TSO                           UPDATE
*
* rdefine facility ddtapfauth uacc(none)
* permit ddtapfauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYDDTAPFAUTH                           NONE
FACILITYDDTAPFAUTH                           ADMIN   READ
FACILITYDDTAPFAUTH                           STCROUPREAD
*
* rdefine facility diag8cmd uacc(none)
* permit diag8cmd cl(facility) id(admin,stcgroup) access(read)
*
FACILITYDIAG8CMD                            NONE
FACILITYDIAG8CMD                            ADMIN   READ
FACILITYDIAG8CMD                            STCROUPREAD
*
* rdefine facility dssauth uacc(none)
* permit dssauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYDSSAUTH                           NONE
FACILITYDSSAUTH                           ADMIN   READ
FACILITYDSSAUTH                           STCROUPREAD
*
```

```

* rdefine facility etpsauth uacc(none)
* permit etpsauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYETPSAUTH                               NONE
FACILITYETPSAUTH                               ADMIN   READ
FACILITYETPSAUTH                               STCROUPREAD
*
* rdefine facility jrpauth uacc(none)
* permit jrpauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYJRPAuth                                NONE
FACILITYJRPAuth                                ADMIN   READ
FACILITYJRPAuth                                STCROUPREAD
*
* rdefine facility nonqauth uacc(none)
* permit nonqauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYNONQAUTH                               NONE
FACILITYNONQAUTH                               ADMIN   READ
FACILITYNONQAUTH                               STCROUPREAD
*
* rdefine facility pgmauth uacc(none)
* permit pgmauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYPGMAUTH                               NONE
FACILITYPGMAUTH                               ADMIN   READ
FACILITYPGMAUTH                               STCROUPREAD
*
* rdefine facility rakfadm uacc(none)
* permit rakfadm cl(facility)           access(none)
* permit rakfadm cl(facility) id(rakfadm,stcgroup) access(read)
*
FACILITYRAKFADM                               NONE
FACILITYRAKFADM                               ADMIN   NONE
FACILITYRAKFADM                               RAKFADM READ
FACILITYRAKFADM                               STCROUPREAD
*
* rdefine facility showauth uacc(none)
* permit showauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYSHOWAUTH                               NONE
FACILITYSHOWAUTH                               ADMIN   READ
FACILITYSHOWAUTH                               STCROUPREAD
*
* rdefine facility stepauth uacc(none)
* permit stepauth cl(facility) id(admin,stcgroup) access(read)
*
FACILITYSTEPAuth                               NONE
FACILITYSTEPAuth                               ADMIN   READ
FACILITYSTEPAuth                               STCROUPREAD
*
* rdefine facility svc244 uacc(none)
* permit svc244 cl(facility) id(admin,stcgroup) access(read)

```

```

*
FACILITYSVC244                               NONE
FACILITYSVC244                               ADMIN   READ
FACILITYSVC244                               STCGROUPREAD
*
* rdefine facility $tstvs uacc(none)
* permit $tstvs cl(facility) id(admin,stcgroup) access(read)
*
FACILITY$TSTVS                                NONE
FACILITY$TSTVS                                ADMIN   READ
FACILITY$TSTVS                                STCGROUPREAD

*
* rdefine tapevol * uacc(read)
* permit * cl(tapevol) id(admin,stcgroup) access(alter)
*
TAPEVOL *                                     READ
TAPEVOL *                                     ADMIN   ALTER
TAPEVOL *                                     STCGROUPALTER
*
* rdefine terminal * uacc(read)
*
TERMINAL*                                      READ

```

This controls access to "classes" if you are familiar with RACF and can be read as follows:

Column	Description
1 - 8	Facility title: DASDVOL, DATASET, FACILITY, etc ¹
9 - 52	Dataset Name, or Generic Name (using asterisk '*')
53 - 60	User Group Id (Installation defined) ²
61 - 66	Permission Level (NONE, READ, UPDATE, ALTER)
67 - 72	Blank

1 See the RACF Administrator's Guide. You need to know the different facility types used by the operating system, CICS, TP products, and vendor products.

2 Blanks in this field denote universal access rules for this resource.

To protect products other than MVS 3.8j, they must have an interface to the security system, typically through the use of specific profiles in the FACILITY class.

Each line can be broken down as:

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Facility Dataset/Generic name          Group    Perms
```

The RAKFUSER and RAKFPROF utilities are used to create the in-core users and profiles tables, respectively. Any user having access to these utilities can completely take over the system by activating "private" users and profiles tables, which is why these utilities need to be protected carefully. To prevent unauthorized use RAKFUSER and RAKFPROF request READ access to the RAKFADM profile in the

FACILITY class. It is strongly recommended to define the RAKFADM profile and create appropriate permissions to protect it. The following RAKF profiles table entries are provided as an example on how to do this:

FACILITYRAKFADM	NONE
FACILITYRAKFADM	RAKFADM READ
FACILITYRAKFADM	STCGROUPREAD

This example defines a universal access of “none” (i.e. nobody has access unless authorized explicitly) and explicitly authorizes user RAKFADM (or all members of group RAKFADM) and all started tasks (the STCGROUP).

 Note that it is RAKF’s standard behavior to grant ALTER access requested for an undefined resource.

Thus, everyone will be able to use the RAKFUSER and RAKFPROF utilities if there is no RAKFADM profile defined in the FACILITY class.

Special attention should be given to the above-mentioned fact that RAKF allows ALTER access to all undefined resources. A user can, for example, delete any dataset on any volume even if access NONE is defined for the dataset as long as there is no DASDVOL profile defined: ALTER access to a DASDVOL allows scratching of any file on it regardless of the file’s protection and exactly that’s what a user gets if no DASDVOL profile is defined.

Batch Jobs and Started Task Considerations

Batch jobs and STCs are assigned the following by ICHSFR00:

Type:	UserID	Group
Batch jobs	PROD	PRDGROUP
Started tasks	STC	STCGROUP

Any job that has no UserID connected to it is assigned a UserID of PROD and a user group of PRDGROUP by ICHSFR00. That situation is true for jobs submitted by RJE or by a local (card or internal) reader. The authorities (i.e. Operations or not) of the PROD and STC UserIDs are hardcoded in ICHSFR00. Upon initial RAKF installation, the PROD user has no Operations authority while the STC user has Operations authority defined. The specific authorities of the PRDGROUP and STCGROUP groups are controlled by the profiles table.

Under MVS 3.8J no UserID propagation takes place. Without further measure all jobs entering the system have no UserID and thus get UserID PROD and group PRDGROUP assigned by RAKF, be it jobs submitted by already authenticated users or jobs, or by started tasks. That means that all jobs that shall run under a specific UserID need to have the USER and PASSWORD parameters coded on their JOB card. Coding these parameters constitutes a security hole if jobs are saved in publicly readable JCL libraries. But even with read protected JCL libraries having to code these parameters is for sure not desired.

To avoid having to code USER and PASSWORD parameters on JOB cards it is highly recommended to install an IKFEFF10 user exit that automatically provides these parameters for jobs submitted using the TSO submit command processor. The SUBMIT command processor is used in RPF submit and REVIEW submit. Both RPF and REVIEW are installed in MVS Turnkey 5. That way most jobs submitted in day-to-day system usage don't need to code these parameters on their JOB cards. The usermod that supplies this feature is ZP60034, available at

<https://www.prycroft6.com.au/vs2mods/#zp60034>, or in the MVS-TK5 library USERMOD.ZP60034.

Security Authorization Facility (SAF) Support in MVS 3.8J

RAKF supports security features that are used by MVS 3.8j differently than by later MVS versions or are not used at all and thus will not work as expected. The "historical" reason for this is that RAKF's predecessor ESG Security System is dated 1991 which is 5 to 10 years later than most components of "current" MVS 3.8j systems. Some of the issues resulting from this discrepancy are discussed here.

General Resource Classes

There are resource classes supported by RAKF but not used by MVS 3.8J. So, if you are going to define a profile in any resource class in the RAKF profiles table that you never used before making sure to test it by trying an access that should be denied by that profile. Only if it then really gets denied, can you be sure that the corresponding hook to call the SAF is already implemented in the level of MVS you are running.

The most relevant of these classes is the PROGRAM class which is meant to be used to protect programs from being executed by unauthorized users: RAKF will accept profiles in the PROGRAM class flawlessly but none of the programs defined there will be protected because MVS 3.8j simply doesn't ask the SAF for permission before executing a program.

Therefore, with MVS 3.8j one has to make sure that "critical" programs cannot be executed by unauthorized users through other means. The following solutions might be feasible on a case-by-case basis:

Programs that need to call a specific SVC for successful execution:

Introduce a profile in the FACILITY class and have that SVC verify the caller's authorization against this profile. This method can only be used if the SVC in question is specific enough to the calling program, so that no other system functions get broken when access to that SVC is restricted. An example for this method is SVC 244: This SVC is originally provided with the Tur(n)key MVS System, where it is used to enable certain utilities and TSO commands to access supervisor state easily. As it is completely unsecured its use as provided in TK3 can only be recommended for single user systems. Systems with security present typically aren't single user systems but still have requirements to allow certain utilities supervisor access when used by authorized users (sysprogs, etc.), but not when used by "normal" users. For this reason, a modification to SVC 244 was made to have it check the calling user's authority to access profile SVC244 in the FACILITY class before allowing authorization. This way SVC 244 is turned from a big security hole to an efficient means of protecting utilities that need to run authorized. The secured version of SVC 244 is part of the auxiliary utility package distributed with RAKF. If you are running a Tur(n)key MVS system having the unsecured version of

this SVC installed, it is strongly recommended to replace it by installing the usermod in Appendix D.

Programs with source code available:

Introduce a profile in the FACILITY class and have the program verify the caller's authorization against this profile. This method of course provides reliable protection only for programs that will not work, for example due to APF requirements, if called from a private library!

An example of this method can be found within RAKF itself. The utilities RAKFUSER and RAKFPROF used to update the in-core USERS and PROFILES tables are critical in the sense that anyone having access to them can take over the security administration of the whole system and thus can conduct arbitrary fraudulent activities. To enable control over who is authorized to use these utilities they request READ access to FACILITY RAKFADM and don't execute if this access isn't granted. That way the use of these utilities can effectively be restricted by defining profile RAKFADM in the FACILITY class and giving only authorized users or groups READ access to this profile.

Programs without source code available:

Create a separate loadlib protected by a DATASET profile and place the program there. This, of course, is kind of a last resort.

Dataset Protection using the RACF indicator.

The implementation of calls to the security system in MVS 3.8j to protect datasets greatly relies on the concept of indication: Only datasets having their "RACF indicator" turned on are protected. The RACF indicator is a bit in the type 1 DSCB of a non-VSAM dataset or in the catalog entry of a VSAM object. Once activated RAKF ensures that all newly created datasets, catalogs and VSAM objects have their RACF indicator turned on. But this is not the case for already existing ones.

Due to artifacts (not to say bugs ;-)) in the logic flow of DADSM processing it is possible to gain access to a RAKF protected dataset which has its RACF indicator turned on by manipulating an arbitrary unprotected dataset (i.e. one with its RACF indicator not turned on) in a certain way before trying to access the protected one. Consequently, reliable dataset protection can only be achieved by explicitly turning on the RACF indicator for all datasets, catalogs and VSAM objects after RAKF activation, not only for those needing protection. On the other hand, it should be noted that once the whole system is protected (i.e. the RACF indicator is turned on for all datasets, catalogs and VSAM objects) it is no longer feasible to run it without RAKF or any other security system being active as most access will then be denied. Consequently, the RACF indicators of all datasets, catalogs and VSAM objects explicitly need to be turned off prior to removing RAKF from the system.

The section "Turning the RACF Indicator on or off" below outlines how to manipulate the RACF indicator of datasets, catalogs or VSAM objects, individually or for the whole system. Using these procedures clearly is the recommended way to enable dataset protection.

Turning the RACF Indicator On or Off

Setting or Clearing the RACF Indicator of a Single Non-VSAM Dataset

Logon to TSO and enter the following command: `CDSCB 'datasetname' VOL(vvvvvv)
UNIT(SYSDA) SHR racind`

Where:

`datasetname` is the fully qualified name of the dataset the RACF indicator of which is to be set or cleared.

`vvvvvv` is the volume identifier of the DASD on which the dataset resides.

`racind` is the desired setting of the RACF indicator:

RACF if the RACF indicator is to be set.

NORACF if the RACF indicator is to be cleared.

The RACF indicator of the following datasets must never be set:

The `PASSWORD` dataset, i.e., the dataset with the single qualifier name `PASSWORD`

Temporary datasets, i.e. datasets with the following naming format:

`SYSnxxxx.Tnnnnnn.RAnnn.*`, where each n represents a decimal digit.

Setting or Clearing the RACF Indicator of a Single Catalog or VSAM Object

Each component of a catalog or VSAM object has its own RACF indicator bit in its catalog entry. The following requirements must be observed when setting or clearing the RACF indicators of catalogs or VSAM objects:

Catalog: Only the RACF indicator of the catalog's cluster entry is to be set or cleared. The RACF indicators of the catalog's data and index components must never be set.

All other VSAM objects: The RACF indicator of the object's catalog entry, as well as the RACF indicators of the object's component catalog entries must be set or cleared synchronously.

To set or clear the RACF indicator of any catalog entry (the catalog cluster itself, any VSAM objects and their components) the RACIND utility is used. It is as follows:

```
//RACIND EXEC PGM=RACIND
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
control statement
.
.
.
control statement
/*
The control statements consist of a command verb beginning in column 1 and an entry name to act upon beginning in column 11. The command verbs are defined as follows:
```

CATALOG directs RACIND to operate in the catalog specified as the entry name until the next CATALOG statement is encountered.

RACON sets the RACF indicator of the specified entry.

RACOFF clears the RACF indicator of the specified entry.

A line starting with an asterisk ("*") is treated as a comment. Any number of control statements is allowed, but no RACON or RACOFF statement is accepted until a valid CATALOG statement has been executed. See Appendix E demonstrating the use of the RACIND utility.

Setting or Clearing the RACF Indicator of All Eligible Non-VSAM Datasets

To view the current RACF indicator status you can run the program VTOC (included in the AUX folder) with the following syntax:

```
VTOC ALL LIM(DSO NE VS) P(NEW (DSN V RA)) S(RA,D,DSN,A) LIN(66) H('1RACF STATUS BEFORE CHANGE').
```

To set the RACF indicator of all eligible non-VSAM datasets in the system you can submit the job `VTOCSRAC.jcl` located in the `TOOLS` folder or job `VTOCSRAC` in `RAKFR.SAMPLIB`. To clear the RACF indicator you can edit this job and change `RXRUN RACF` to `RXRUN NORACF` and resubmit the job.

 This job requires BREXX V2R5 to run.

`VTOCSRAC` operates on all mounted DASD volumes. Excluding certain volumes from processing vary them offline before submitting `VTOCSRAC`. The job executes a REXX script to find the eligible datasets out of a dataset list and to generate a `CDSCB` command with the `RACF` or `NORACF` parameter for each dataset found. Datasets are considered eligible for RACF indicator modification if they are not: The `PASSWORD` dataset, i.e., the dataset with the single qualifier name `PASSWORD` Temporary datasets, i.e. datasets with the following naming format: `SYSnxxxx.Tnnnnnn.RAnnn.*`, where each `n` represents a decimal digit.

Setting or Clearing the RACF Indicator of All Catalogs and VSAM Objects

To view the current RACF indicator status of all catalogs and VSAM objects in the system submit job `VSAMLRAC.jcl` located in the `TOOLS` folder or job `VSAMLRAC` in `RAKF.SAMPLIB`. This job requires BREXX to run.

To set the RACF indicator of all eligible VSAM datasets in the system you can submit the job `VSAMSRAC.jcl` located in the `TOOLS` folder or job `VSAMSRAC` in `RAKF.SAMPLIB`. To clear the RACF indicator you can edit this job and change `RXRUN ON` to `RXRUN OFF` and resubmit the job.

`VSAMSRAC` operates on the master catalog and on all user catalogs defined in the master catalog. To exclude a user catalog from processing vary the volume on which it resides offline before submitting `VSAMSRAC`. The job executes an AWK script to find the catalogs, VSAM objects and their components out of IDCAMS `LISTCAT` output and to generate `RACIND` control statements for all eligible entries. All VSAM entries, not being components (DATA or INDEX) of a catalog are considered eligible.

Auxiliary Utilities

Section “Turning the RACF Indicator On or Off” outlines procedures for setting or clearing the RACF indicator of datasets, catalogs and VSAM objects which is crucial to provide a reliable dataset protection. These procedures make use of a set of utilities that are neither part of the `RAKF` product nor of native MVS 3.8j. They have been collected or derived from different publicly available sources and are provided with the `RAKF` in the `AUX` folder. This includes:

`VTOC`
`CDSCB`

Version History

ESG Security System (March 1991)

The ESG Security System was published by Craig J. Yasuna as an alternative to IBM's RACF™ and similar products. It uses the ICHRTX00 security router exit to communicate its security decisions to MVS. ICHRTX00 is a user exit of the SAF router ICHSFR00.

RAKF "RAcK oF" Security System (August 2005)

RAKF was published by A. Philip Dickinson as an adaptation of the ESG Security System to MVS 3.8J. ESG Security doesn't natively support MVS 3.8J for several reasons. The major ones are:

At MVS 3.8j times the SAF router ICHSFR00 was part of the RACF™ product and thus not available on MVS systems without RACF™ being installed. So, the ICHRTX00 exit as a convenient place for third party security products to hook into the SAF didn't exist.

incompatible ACEE handling.

The ESG Security System uses the BAS instruction which is not available in S/370. ESG Security's 24/31 bit AMODE handling wasn't compatible with the usage of high order address bytes for flags in MVS 3.8j.

incompatible parameter lists of RACROUTE, RACDEF, RACHECK and RACINIT.

handling of in core profiles incompatible with MVS's VSAM catalog management.

Phil D. converted ESG Security's ICHRTX00 router exit into an ICHSFR00 SAF router to overcome the first major incompatibility. He also solved the next two points and in parts the 24/31 bit AMODE issues. The rest remained undetected when he published his work and caused several problems when using RAKF on MVS 3.8J: Arbitrary invalid authorization decisions: Access denied if it should have been granted and access granted if it should have been denied. The latter is in a way that with some trial and error any user is able to acquire ALTER access to any dataset in the system.

RAKF 1.2.0 (April 2011)

When the above-mentioned problems became visible and identified as being RAKF caused in several MVS 3.8J environments it turned out that Phil D's RAKF source wasn't accessible anymore (lost in a package crate from moving). Phil Roberts stepped in and reconstructed the source by disassembling Phil D's binary RAKF distribution and comparing it with the original source of the ESG Security System. Based on that reconstructed source the rest of the incompatibilities listed above were identified and corrected by Jürgen Winkelmann.

To avoid another loss of the source to occur an additional effort has been made to clean up the source to a publishable state and to provide an SMP4 compatible source distribution enabling RAKF installation using the standard SMP 4 standard RECEIVE, APPLY, ACCEPT procedure.

RAKF 1.2.0 PTF Summary

The following PTFs are available:

RRKF001: Enable comment lines (lines starting with *) to be entered in the source users and profiles tables and minor bug fixes in RAKFUSER utility.

RRKF002: Enhance RACINIT NEWPASS functionality to support permanent password changes to be initiated by an application. This enables standard password change functionality as for example entering currentpw/newpw on the “ENTER CURRENT PASSWORD for uuuuuuu” at TSO logon time.

RRKF003: Security enhancement in users and profiles tables processing.

RRKF004: Consolidation of documentation members from RAKF’s SAMPLIB to the RAKF User’s Guide.

RRKF005: Introduce the RACIND utility to control VSAM RACF indicators.

RRKF006: Sample jobs and utilities to turn the RACF indicator on or off for all non-VSAM datasets, catalogs and VSAM objects. Sample jobs to create the profiles and user’s tables and the password update queue. Documentation changes and minor bug fixes. These PTFs are available in folder files/RAKF/PTFs of the H390-MVS groups.io group and in CBT file 850, future PTFs will be made available at the same locations.

RRKF007: Eliminate the alphabetical SORT requirement of the members USERS and PROFILES.

RRKF008: Replaces VTOCSRAC, VSAMLRAC and VSAMSRAC members in RAKF.SAMPLIB to replace the mawk versions to the BREXX versions.

RAKF 1.2.6 (October 2021)

To make source management more open and to make RAKF more approachable all PTFs were combined to have a unified source repository. The following changes were also made:

Replace vtoc, cdscb binaries with source libraries. In MVS Turnkey 5, these sources are resident in TK5.ASM.

Created shell script to generate SMP4 entries for RAKF.

Removed MAWK (no longer needed).

Replaced IGC0024D binary with SVC 244 source and created usermod RAK0001. In MVS Turnkey 5, the source of SVC 244 is resident in TK5.ASM and the installation JCL is resident in TK5.CNTL, member name SVC244\$.

Replaced VSAMLRAC, VSAMSRAC, and VTOCSRAC with new versions that use BREXX scripts instead of MAWK. In MVS Turnkey 5, sysmod RRKF008 is applied to install these members in the RAKF.SAMPLIB.

Removed the requirement to use tape/XMI files.

Removed xmi/ptf JCL and other SAMPLIB items.

Moved SAMPLIB folder to TOOLS.

Updated RACIND.h1asm to check if RAKF is installed and if not to skip RACHECK, this is helpful for installing before RAKF is enabled.

MVS Turnkey 5 runs with RAKF 1.2.0 with some extra PTF's

Appendix A - Generating your own release

Run the script generate_rakf.py which will generate the JCL jobstream needed to install RAKF.

Appendix B - Add users to UADS

This job is for use with MVS Turnkey 5 and is resident as member ADDUSRON in SYS2.JCLLIB.

```
//ADDUSRON JOB (TSO),  
//          'Add TSO users',  
//          CLASS=A,  
//          MSGCLASS=X,  
//          COND=(0,NE)  
//*****  
/*  
/* Name: SYS2.JCLLIB(ADDUSRON)  
/*  
/* Desc: Add new TSO users  
/*      without deleting and allocation TSO dataset and  
/*      without defining an ALIAS.  
/*  
/* Note: The newly created userids do not have a password set  
/*  
//*****  
/*  
//ADDUSERP EXEC ADDUSRON,HLQ=newuser,UTYPE=SYSP OR USER
```

In this job replace 'newuser' with the UserID to be created.

Appendix C - SORT job

Below is an MVTSORT job which will sort the USERS and PROFILES members, which is not needed anymore when PTF RRKF007 is applied.

```
//RAKFUPST JOB (TSO),  
//          'UP',  
//          CLASS=S,  
//          MSGCLASS=A,  
//          REGION=8192K,  
//          MSGLEVEL=(1,1),  
/*  
/* RAKF requires these files be in sorted order  
/* use this JCL to sort the tables  
/*  
//SORT    EXEC PGM=SORT,REGION=512K,PARM='MSG=AP'
```

```

//SYSOUT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTOUT DD DSN=SYS1.SECURE.CNTL(USERS),
//          DISP=SHR,DCB=(BLKSIZE=80,RECFM=F)
//SORTWK01 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK02 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK03 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK04 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK05 DD UNIT=3390,SPACE=(CYL,(5,5))
//SYSIN DD *
      SORT FIELDS=(1,80,CH,A)
      RECORD TYPE=F,LENGTH=(80)
      END
/*
//SORTIN DD *
##### PUT USERS HERE
/*
//SORT EXEC PGM=SORT,REGION=512K,PARM='MSG=AP'
//STEPLIB DD DSN=SYSC.LINKLIB,DISP=SHR
//SYSOUT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SORTLIB DD DSNAME=SYS1.SORTLIB,DISP=SHR
//SORTOUT DD DSN=SYS1.SECURE.CNTL(PROFILES),
//          DISP=SHR,DCB=(BLKSIZE=80,RECFM=F)
//SORTWK01 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK02 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK03 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK04 DD UNIT=3390,SPACE=(CYL,(5,5))
//SORTWK05 DD UNIT=3390,SPACE=(CYL,(5,5))
//SYSIN DD *
      SORT FIELDS=(1,80,CH,A)
      RECORD TYPE=F,LENGTH=(80)
      END
/*
//SORTIN DD *
##### PUT PROFILES HERE
/*

```

Appendix D - USERMOD RAK0001

This usermod installs an updated SVC244 that uses RAKF for authorization. In MVS Turnkey 5 this usermod is replaced by member SVC244\$ in TK5.CNTL.

```
//RAK00011 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD *
++USERMOD(RAK0001)          /* FIX LOGREC DEVICE TYPE SUPPORT * .
++VER(Z038) FMID(EBB1102)
/*
 * Desc: Type 3/4 SVC for setting/unsetting JSCBAUTH
*
*      If a security system (for example RAKF) is installed
*      read access to ressource SVC244 in class FACILITY is
*      requested and the JSCBAUTH change is made only if the
*      security system grants this access
*/.
++MOD(IGC0024D) DISTLIB(LPALIBA).
/*
//SYSUT2 DD DSN=&&SMPMCS,DISP=(NEW,PASS),UNIT=SYSALLDA,
//           SPACE=(CYL,3),
//           DCB=(DSORG=PS,RECFM=FB,LRECL=80,BLKSIZE=4080)
//SYSIN DD DUMMY
/*
//RAK00012 EXEC PGM=IFOX00,PARM='OBJECT,NODECK,NOTERM,XREF(SHORT),RENT'
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD UNIT=SYSALLDA,SPACE=(CYL,10)
//SYSUT2 DD UNIT=SYSALLDA,SPACE=(CYL,10)
//SYSUT3 DD UNIT=SYSALLDA,SPACE=(CYL,10)
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//           DD DSN=SYS1.SMPMTS,DISP=SHR
//           DD DSN=SYS1.AMODGEN,DISP=SHR
//SYSGO DD DSN=&&SMPMCS,DISP=(MOD,PASS)
//SYSIN DD *
        TITLE ' SVC 244 - Toggle Authorization '
*****
*
* Name: IGC0024D
*
* Type: Assembler source
*
* Desc: Type 3/4 SVC for setting/unsetting JSCBAUTH
*
*      If a security system (for example RAKF) is installed
*      read access to ressource SVC244 in class FACILITY is
*      requested and the JSCBAUTH change is made only if the
*      security system grants this access
*
* Regs at Entry:
*
```

```

*      R0 must be 0                                     *
*      R1 = Request code.   R1 = 1 ===> Authon        *
*                                         else ===> Authoff    *
*      R2      undetermined                           *
*      R3      ---> CVT                            *
*      R4      ---> TCB                            *
*      R5      ---> SVRB                           *
*      R6      ---> Entry point                     *
*      R7      ---> ASCB                           *
*      R8      ---> undetermined                   *
*      R9      ---> undetermined                   *
*      R10     ---> undetermined                  *
*      R11     ---> undetermined                  *
*      R12     ---> undetermined                  *
*
***** IGC0024D CSECT , SVC 244
R0      EQU    0
R1      EQU    1
R2      EQU    2
R3      EQU    3
R4      EQU    4
R5      EQU    5
R6      EQU    6
R7      EQU    7
R8      EQU    8
R9      EQU    9
R10     EQU   10
R11     EQU   11
R12     EQU   12
R13     EQU   13
R14     EQU   14
R15     EQU   15
      USING *,6          , use R6 as base register
      LTR    R0,R0          , R0 = 0 ?
      BNZR   R14           , return if not
      ICM    R8,B'1111',CVTSAF(R3) , SAFV defined ?
      BZ     GOFORIT        , no RAC, permit JSCB auth
      USING SAFV,R8         , addressability of SAFV
      CLC    SAFVIDEN(4),SAFVID , SAFV initialized ?
      BNE    GOFORIT        , no RAC, permit JSCB auth
      DROP   R8             , SAFV no longer needed
      LR     R8,R1           , remember R1
      LR     R9,R15          , remember R15
      RACHECK ENTITY=SVC244,CLASS='FACILITY',ATTR=READ ask RAC
      LR     R1,R8           , restore R1
      LR     R8,R15          , remember return code
      LR     R15,R9          , restore R15
      XR     R0,R0           , restore R0
      LTR    R8,R8           , RAC authorization granted?
      BNZR   R14           , return if not
      GOFORIT L    R11,180(,R4) , R11 = JSCB (from TCBJSCB)
      BCT    R1,AUTHOFF       , R1 NOT = 1 ===> Authoff

```

```

        OI      236(R11),X'01'           , set JSCBAUTH on
        BR      R14                      , and return
AUTHOFF NI      236(11),255-X'01'       , set JSCBAUTH off
        BR      R14                      , and return
SVC244  DC      CL39'SVC244'          , facility name to authorize
SAFVID  DC      CL4'SAFV'            , SAFV eye catcher
CVTSAF  EQU    248 CVTSAF doesn't exist but is a reserved field in 3.8J
        ICHSAFV DSECT=YES             , map SAFV
        END                           , of SVC 244
/*
//RAK00013 EXEC SMPREC,WORK='SYSALLDA'
//SMPPTFIN DD DSN=&&SMPMCS,DISP=(OLD,DELETE)
//SMPCNTL DD *
RECEIVE
        SELECT(RAK0001)

/*
///*
//RAK00014 EXEC SMPAPP,WORK='SYSALLDA'
//SMPCNTL DD *
APPLY
        SELECT(RAK0001)
        CHECK
.

/*
///*
//RAK00015 EXEC SMPAPP,COND=(0,NE),WORK='SYSALLDA'
//SMPCNTL DD *
APPLY
        SELECT(RAK0001)
        DIS(WRITE)
/* COMPRESS(ALL) */
.

/*

```

Appendix E - RACIND Example

```
//RACIND   JOB
//*****
//*
//** Name: RACIND
//*
//** Desc: Run RACIND Utility
//*
//** FUNTION: Act upon control statements read from SYSIN to set or
//**           clear the RACF indicator of VSAM catalog entries. The
//**           following control statements are valid:
//*
//**           -----+-----1-----+-----2-----+-----3-----+-----4-----+-----5-----+
//**           CATALOG    name of catalog to search for entries
//**           RACON      name of entry to indicate
//**           RACOFF     name of entry to unindicate
//**           * Comment
//*
//**           Any number of control statements is allowed. The first
//**           none comment statement must be a CATALOG statement. A
//**           CATALOG statement remains active until a new CATALOG
//**           statement replaces it.
//*
//*****
//RACIND   EXEC PGM=RACIND
//SYSPRINT DD  SYSOUT=*
//SYSIN    DD  *
*****
*
* Example: Switch on the RACF indicator for a VSAM catalog
*           and a cluster contained in that catalog.
*
* Note:    - The data and index components of a VSAM catalog
*           MUST NOT be RACF indicated.
*
*           - All other entry types MUST have either all of their
*             components RACF indicated or all components not
*             indicated.
*
*           For that reason in the example only one RACON statement
*           is coded for the catalog, but three for the cluster.
*
*****
CATALOG  SYS1.UCAT.TST
RACON    SYS1.UCAT.TST
RACON    TSTCAT.CLUSTER
RACON    TSTCAT.CLUSTER.INDEX
RACON    TSTCAT.CLUSTER.DATA
/*
//
```