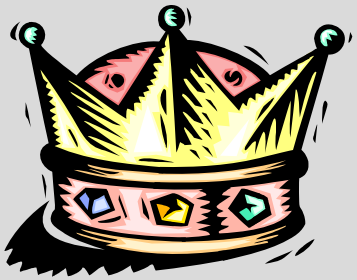
Vanguard Enterprise Security Expo 2009 - Session RAA1 - June 2009



Using REXX with the RACF Database Unload

Robert L. Whittle

Senior RACF Specialist - RSH Consulting, Inc.

R.Whittle@rshconsulting.com - 617-969-9050 - www.rshconsulting.com

TOPICS

RACF Database Unload

REXX

Execution

Input

Process

Output

RACF DATABASE UNLOAD

Generates sequential, text file copy of a RACF database

Program - IRRDBU00

Record series

- **0100 Group**
- **0200 User**
- **0400 Dataset**
- **0500 General Resource**

RACF DATABASE UNLOAD

```
//jobname  JOB (account),'username',CLASS=x,MSGCLASS=x
//STEP0001 EXEC PGM=IRRUT200
//SYSPRINT DD  SYSOUT=*
//SYSRACF  DD  DSN=racf.database.dsname,DISP=SHR
//SYSUT1   DD  DSN=&&RACFBK,
//             DISP=(NEW,PASS,DELETE),UNIT=DISK,
//             SPACE=(CYL,(100)),DCB=(RECFM=F,LRECL=4096)
//SYSUT2   DD  SYSOUT=*
//SYSIN    DD  *
  MAP
  END
/*
//STEP0002 EXEC PGM=IRRDBU00,PARM=NOLOCKINPUT
//SYSPRINT DD  SYSOUT=*
//INDD1    DD  DSN=&&RACFBK,DISP=(OLD,DELETE,DELETE)
//OUTDD    DD  DSN=output.racf.flatfile.dsn,DISP=SHR
//             DISP=(NEW,PASS,DELETE),UNIT=DISK,
//             SPACE=(CYL,(100,25),RLSE),DCB=(RECFM=VB,LRECL=4096)
```

REXX

REXX - REstructured eXtended eXecutor

Simple interpreted procedural language

Developed by IBM 1979 - 1982

Design concepts

- **Readability**
- **Natural data typing**
- **Nothing to declare**
- **Dynamic scoping**
- **Small number of keywords and functions**
- **System independent**

Sample REXX Routine

```
/* REXX - REMOVE ALL USERS FROM A GROUP */
ARG GROUPID
ADDRESS TSO 'EXECIO 1 DISKR DBUNLOAD'
DO UNTIL RC \= 0
  PULL DBUREC
  REC_TYPE = SUBSTR(DBUREC,1,4)
  IF REC_TYPE = '0205' & ,
     STRIP(SUBSTR(DBUREC,15,8)) = GROUPID THEN
    DO
      MEMBER = SUBSTR(DBUREC,6,8)
      QUEUE 'REMOVE' MEMBER 'GROUP('GROUPID')'
      ADDRESS TSO 'EXECIO 1 DISKW RACFCMDS'
    END
  ADDRESS TSO 'EXECIO 1 DISKR DBUNLOAD'
END
EXIT
```

REXX - General Syntax

Comments: Text enclosed in /* */ (/* REXX */ - first line)

Clause: Program line - terminated by line end or ' ; '

- **Null** Blank (readability)
- **Label** Section or Called Routine Names - ends with ' : '
- **Instruction** Assignment, Keyword, or Command

Tokens: Components of an Instruction Clause

- **Literal** Enclosed in quotes - ‘ ’ or “ ”
- **Symbols** Constant, Variable, Keyword - A-Z a-z 0-9 . ! ? _
- **Operators** + - * / % // | & = < > \ ¬
- **Special** , ; : ( ) (, = line continuation)

REXX - General Syntax

Operators

- **Arithmetic (in order of precedence): ** * / % // + -**
- **Comparative:**

	Normal	Strict
Equal	=	==
Not Equal	\= ¬= <> ><	\== ¬==
Greater/Less than	> <	>> <<
Equal or Greater	>= \< ¬<	>>= \<< ¬<<
Equal or Less	<= \> ¬>	<<= \>> ¬>>

- **Logical (in order of precedence): & =and | =or && =exclusive or**
- **Concatenate:**
  - **With intervening blank: literal/variable *blank(s)* literal/variable**
  - **Without intervening blank: || -or- (abuttal) -or- (abuttal)''(abuttal)**
  - **Examples: USERID = 'RSH001' ; NAME = 'BOB'**

    IDNAME = USERID NAME ➔ RSH001 BOB

    IDNAME = USERID || NAME ➔ RSH001BOB

    IDNAME = USERID'-'NAME ➔ RSH001-BOB

REXX - General Syntax

Symbols

- **Constant:** Starts with a numeric value or period (99)
- **Simple:** Single name (USERID)
- **Compound:** Multi-level names, separated by periods (NAMES.ID)
- **Stem:** First level of multi-level name (NAMES.)

Variables (Simple or Compound Symbol)

- Named object - assigned a value -or- defaults to its name (upper case)
- Name - 1st character cannot be a number; up to 250 characters long
- Assignment: symbol = expression (USERID = 'RSH001')

```
EX:   USERID = SUBSTR(DBUREC,6,8)
      DATA.USERID = SUBSTR(DBUREC,125,25)
      RPTLINE = 'USER:'  USERID  DATA.USERID
```

REXX - Keyword Instructions

ADDRESS	**Direct command to environment (e.g. TSO)**
ARG	**Retrieve input data and load into variable(s)**
DO	**Repetitive loop - terminated by END**
EXIT	**Terminate processing**
IF	**THEN-ELSE Conditional execution**
NOP	**No operation, dummy instruction**
PARSE	**Parse string of data into multiple variables**
PULL	**Read data from the external data queue**
QUEUE	**Add data to a queue/stack (FIFO)**
SAY	**Display data to default output stream**
SELECT	**WHEN-WHEN-OTHERWISE Conditional execution, multiple alternatives - terminated by END**
TRACE	**Provide diagnostic information**

REXX - Built-in Functions

DATATYPE	**Test for alpha, numeric, alphanumeric**
DATE	**Obtain current date**
LEFT	**Align to the left and pad**
LENGTH	**Find length of string**
POS	**Find location of a string within a string**
RIGHT	**Align to the right and pad**
SPACE	**Pad words with spaces or strip spaces**
STRIP	**Strip leading and/or trailing characters (e.g., blanks)**
SUBSTR	**Obtain a sub-string (i.e., portion) of a string**
TRANSLATE	**Convert characters**

REXX - TSO Reserved Keywords

DELSTACK	**Deletes most recently created stack**
NEWSTACK	**Create new stack**
DROPBUF	**Drops most recently created buffer**
MAKEBUF	**Make new buffer**
EXECIO	**Execute I/O - File read & write**

Enclose in quotes - e.g. 'DELSTACK'

REXX - TSO Stream I/O

Optional Add-on Function Package for TSO/E

CHARIN	**Read characters from input stream**
CHAROUT	**Write characters to output stream**
LINEIN	**Read a line from input stream**
LINEOUT	**Write a line to the output stream**
LINES	**Return number lines remaining in input stream**
STREAM	**State of stream or action on stream (e.g., OPEN)**

REXX - TSO External Functions

LISTDSI	*	**Obtain dataset information**
MSG	*	**Turn display of messages 'ON' and 'OFF'**
MVSVAR		**Return information about MVS, TSO, & Session**
OUTTRAP	*	**Capture output from a command**
RACVAR	**	**Return RACF information about current user**
STORAGE		**Obtain data from address in storage**
SYSDSN	*	**Determine if dataset exists**
SYSVAR	*	**Return information about MVS, TSO, & Session**
USERID		**Return current ID or Job/Step Name**

*** - can only use when running in a TSO/E address space**

**** - must be installed to use**

EXECUTION

TSO Foreground or Background Batch

```
//jobname  JOB (account),'name',CLASS=x,MSGCLASS=x,REGION=50M
//STEP0010 EXEC PGM=IKJEFT01,DYNAMNBR=5
//DBUNLOAD DD DSN=input.racf.flatfile.dsn,DISP=SHR
//REPORT   DD DSN=report.dsn,DISP=(NEW,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//            DCB=(RECFM=FB,LRECL=350)
//SYSTSPRT DD SYSOUT=*
  ------------- EITHER -------------
//SYSTSIN  DD *
EXEC 'rexx.lib.dsn(execname)' 'arguments'
  ------------- OR -------------
//SYSEXEC  DD DSN=rexx.lib.dsn,DISP=SHR        -or- //SYSPROC
//SYSTSIN  DD *
  %execname arguments
```

EXECUTION

IRXJCL - Exec Processing Routine

```
//jobname  JOB (account),'name',CLASS=x,MSGCLASS=x,REGION=0M
//STEP0010 EXEC PGM=IRXJCL,PARM='execname arguments'
//DBUNLOAD DD DSN=input.racf.flatfile.dsn,DISP=SHR
//REPORT   DD DSN=report.dsn,DISP=(NEW,CATLG,DELETE),
//            UNIT=SYSDA,SPACE=(TRK,(10,5),RLSE),
//            DCB=(RECFM=FB,LRECL=350)
//SYSEXEC  DD DSN=rexx.lib.dsn,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD DUMMY
```

More efficient - less TSO overhead

<u>Cannot</u> perform most TSO External Functions

INPUT - Argument

Pass parameters or options to EXEC

Argument - execname ‘argument [argument] …’

- **ARG variable [variable …]**
- **PARSE [UPPER] ARG variable [variable …]**

Ex: EXEC IDINFO ‘rsh001’

ARG USERID

USERID ➔ RSH001

INPUT - File

EXECIO #lines DISKR ddname [(STEM stem. FINIS]

Read & process records individually

```
ADDRESS TSO 'EXECIO 1 DISKR DBUNLOAD'
IF RC \= 0 THEN
  DO
    SAY 'ERROR ON OPEN DBUNLOAD'
    EXIT
  END
DO UNTIL RC \= 0
  PULL DBUREC
  -- process records --
  ADDRESS TSO 'EXECIO 1 DISKR DBUNLOAD'
END
```

INPUT - File

Read all records, then process (stem_variable.0 = # records)

```
ADDRESS TSO 'EXECIO * DISKR DBUNLOAD (STEM DBUIN. FINIS’
IF RC \= 0 THEN
  DO
    SAY 'ERROR ON OPEN DBUNLOAD'
    EXIT
  END
DO X = 1 to DBUIN.0
  DBUREC = DBUIN.X
  -- process records --
END
```

PROCESS - Obtaining DBU Record Fields

SUBSTR(string,position#,length)

```
RECNUM = SUBSTR(DBUREC,1,4)
IF RECNUM = '0100' THEN
  GROUPID = SUBSTR(DBUREC,6,8)
```

PARSE VAR string position# variable position# variable

```
PARSE VAR DBUREC ,
  1    RECNUM ,
  5    .
IF RECNUM = '0100' THEN
  PARSE VAR DBUREC ,
    6    GROUPID ,
    14   . ,
    58   GRP_INSTDATA ,
    313  .                          . = ignore
```

PROCESS - Conditional Processing

IF - THEN - ELSE

```
IF USERID = 'IBMUSER' THEN
  NOP
ELSE
  RPTLINE = USERID USERNAME


IF SUBSTR(GROUPID,1,3) = 'SYS' THEN
  IF SUBSTR(GRPMBR,1,4) = 'TECH' THEN
    RPTLINE = GROUPID GRPMBR
-------- or ----------
IF SUBSTR(GROUPID,1,3) = 'SYS' & ,
   SUBSTR(GRPMBR,1,4) = 'TECH' THEN
  RPTLINE = GROUPID GRPMBR
```

PROCESS - Conditional Processing

SELECT - WHEN - OTHERWISE

```
SELECT
  WHEN GROUPID = 'SYS1' THEN
    NOP
  WHEN SUBSTR(GROUPID,1,3) = 'DFL' THEN
    RPTLINE = GROUPID USERID USERNAME
  WHEN SUBSTR(GROUPID,1,3) = '@03' THEN
    RPTLINE = GROUPID USERID USERNAME GROUPAUTH
  OTHERWISE
    RPTLINE = GROUPID USERID
END
```

PROCESS - Find Particular Values

```
/* Find 6 character IDs starting with '@EXT' & 2 numbers */
IF LENGTH(USERID) = 6 & ,
   SUBSTR(USERID,1,4) = '@EXT' & ,
   DATATYPE(SUBSTR(USERID,5,2)) = 'NUM' THEN
  -- process record --


/* Find all IDs starting with 'I' except IBMUSER */
IF SUBSTR(USERID,1,1) = 'I' & ,
   USERID \= 'IBMUSER' THEN
  -- process record --


/* Find all profiles containing '**' */
IF POS('**',PROFILE) > 0 THEN
  -- process record --
```

PROCESS - Parsing Text

PARSE VALUE string WITH variable ‘literal’ variable

```
IF POS(‘EMP#’,INSTDATA) > 0 THEN
  PARSE VALUE INSTDATA WITH . ‘EMP#’ EMPNO .

PARSE VALUE LOGONDATE WITH YEAR ‘-’ MONTH ‘-’ DAY
```

PARSE VAR string variable variable [variable …]

```
PARSE VAR USERNAME NAMEF NAMEMI NAMEL
```

PROCESS - Using Compound Variables

Storing data for reference

```
IF RECNUM = ‘0102’ THEN
  DO
    GROUPID = STRIP(SUBSTR(DBUREC,6,8))
    IF GROUPID = ‘SYS1’ THEN
      DO
        MEMBERID = STRIP(SUBSTR(DBUREC,15,8))
        SYS1.MEMBERID = ‘X’
      END
  END
---
IF RECNUM = '0200' THEN
  DO
    USERID = STRIP(SUBSTR(DBUREC,6,8))
    IF SYS1.USERID = ‘X’ THEN
      -- process record --
```

PROCESS - Using Compound Variables

Storing data for retrieval

```
IF RECNUM = ‘0100’ THEN
  DO
    GROUPID = STRIP(SUBSTR(DBUREC,6,8))
    INSTDATA = STRIP(SUBSTR(DBUREC,58,255))
    GROUPDATA.GROUPID = INSTDATA
  END
---
IF RECNUM = ‘0205’
  DO
    USERID = STRIP(SUBSTR(DBUREC,6,8))
    CONGROUP = STRIP(SUBSTR(DBUREC,15,8))
    GROUP_OPER = SUBSTR(DBUREC,89,3)
    IF GROUP_OPER = 'YES' THEN
      RPTLINE = USERID CONGROUP GROUPDATA.CONGROUP
  END
```

PROCESS - Preparing Data for Output

LEFT(string,length,pad) **- default pad is blank**

RIGHT(string,length,pad)

```
HDRLINE = ,
  'USERID-' ,
  LEFT('USER-NAME',20,'-') ,
  '#GR'
RPTLINE = ,
  LEFT(USERID,8) ,
  LEFT(USERNAME,20) ,
  RIGHT(NUMGRPS,3,0)

HDRLINE ➔ USERID- USER-NAME---------- #GR
RPTLINE ➔ RSH001  BOB HANSEL           010
```

PROCESS - Capturing Command Output

Variable = OUTTRAP('stem.')

```
LURESULTS = OUTTRAP('LUCMD.')
ADDRESS TSO 'LU' USERID
X = OUTTRAP("OFF")
FIRSTLINE = LUCMD.1
```

```
LUCMD.0 ➔ 50
LUCMD.1 ➔ USER=JSMITH1  NAME=JOHN SMITH            OWNER=SECGRP1 …
LUCMD.2 ➔  DEFAULT-GROUP=USRGRPA   PASSDATE=01.351  PASS-INTERVAL= 30
LUCMD.3 ➔  ATTRIBUTES=OPERATIONS
…
```

OUTPUT - Default Output Stream

SAY string

```
SAY ‘Number of records processed’ X

SAY ‘Error - User’ USERID ‘not found’
```

OUTPUT - File

EXECIO #lines DISKW ddname [(STEM stem. FINIS]

Write records individually while processing

```
DO
  -- process records --
  'NEWSTACK'
  QUEUE output-string
  ADDRESS TSO 'EXECIO 1 DISKW REPORT'
  'DELSTACK'
END
```

OUTPUT - File

Write all output lines after completing all processing

```
RPTLINE. = ‘’
R = 0
DO
  -- process records --
  R = R + 1
  RPTLINE.R = output-string
END
ADDRESS TSO 'EXECIO * DISKW REPORT (STEM RPTLINE. FINIS’
```

Writing stops when EXECIO encounters a null line or uninitialized variable

<u>Caution</u>: If writing RACF commands to a dataset and run out of space, remaining commands will be *<u>executed</u>*

REFERENCES

RACF Security Administrator's Guide

RACF Systems Programmer's Guide

RACF Macros and Interfaces

TSO/E REXX User's Guide

TSO/E REXX Reference

REXX Language Assn. - www.rexxla.org

IBM REXX Home Page - www2.hursley.ibm.com/rexx

TSO-REXX Internet Discussion List

The REXX Language: A Practical Approach to Programming (2nd Ed), M. F. Cowlishaw, Prentice-Hall, 1990 (ISBN 0-13-780651-5)

REXX Programmer's Reference, Howard Fosdick, Wiley Publishing, 2005 (ISBN 0-7645-7996-7)