



VSE Automation with REXX VSE

Charles Rice
Carolina Steel Corporation



Sample Application

- ✦ Console Automation Sample (REXXCO)
 - Documented in the REXX/VSE Reference
 - Framework that can be modified to fit your environment

REXXCO

- ✦ Maintains an active REXX console
- ✦ Monitors all messages that are available at this console
- ✦ User will define a message action table
- ✦ REXXCO will run in a partition all the time

Message Action Table

- ✦ Table is catalogued in a VSE Library
- ✦ Each entry in table has three fields
 - Action – First word in the entry and may be used to initiate an action to be taken after a match of the Find_String is found in a console message
 - Find_String – The search string to find in a console message. Starts in the second word of the entry and can extend until column 71
 - Data – Any character string that is additional information for the function that performs action.

Message Action Table

```
/* REXXTABL REXX/VSE Message-Action-Table for Demonstration Purposes */
/* Action FindString(2nd word..pos 71) Data,in continuation lines */
FLUSH1Q52I 1Q52I
FLUSHCPU ARX0998I
REPLY HIT ENTER TO CLOSE THE FILE
REPLY HALT EXIT REACHED, PRESS ENTER TO END
&REXXSTOP THIS LOOP CAN BE STOPPED
REXXCO<< VSAM WORKFILE: X
INNAME=PRD1.BASE.SKRXVSAM Z OUTNAME=PRD1.BASE.REXXSPCE.JOB X
VAR001=REXXSPCE VAR002=PRD1.BASE VAR003=PRD1.BASE.SKRXJCL.Z
```

REXXCO Actions

- 🔦 Position 1 of action = &
 - Remainder of action is the name of an external REXX program that will be called
- 🔦 Position 1 of action = *
 - Creates a new batch job using a given job skeleton and submits the job to the VSE/POWER reader queue
- 🔦 All other actions
 - Calls a REXX program called REXXEVT and will execute a routine in the program with the label equal to action

Run External REXX

```
&REXXSTOP THIS LOOP CAN BE STOPPED

ARG lockid,message,tblentry
PARSE SOURCE . . rexxname .
rc = 0
position = POS('-',WORD(message,1)) /* Did we get something like: */
IF position > 0 THEN /* R1-47 // PAUSE */
DO /* Extract from message... */
  replyid = SUBSTR(WORD(message,1),position+1) /* Reply id */
  pid = SUBSTR(WORD(message,1),1,position-1) /* Partition id */
END
ELSE DO
  replyid= WORD(message,2) /* Reply id */
  pid = WORD(message,1) /* Partition id */
END
CALL ASSGN 'STDOUT','SYSLOG'
SAY rexxname "ISSUES 'MSG' pid",DATA=HI' IN ORDER TO HALT EXECUTION"
CALL SENDCMD 'MSG' pid',DATA=HI' /* Stop the REXX program */
RETURN rc
```

Submit a Job Action

```
REXXCO<< VSAM WORKFILE: X
      INNAME=PRD1. BASE. SKRXVSAM Z OUTNAME=PRD1. BASE. REXXSPCE. JOB X
      VAR001=REXXSPCE VAR002=PRD1. BASE VAR003=PRD1. BASE. SKRXJCL. Z

      SSSS JOB JNM--var001--, CLASS=Y, PRI=8, DISP=D
      SSSS LST CLASS=Q, DISP=D
      // JOB --var001-- Work request ALLOC/FREE VSAM space
      /. X Skeleton for REXX Console Automation Demo Purposes
      /. X Used by REXXCO as input file for the MERGE function
      // LIBDEF *,SEARCH=PRD1. BASE
      // EXEC REXX--var001--, PARM='--rexxlock--,--rexxnum--,--var003--,--reX
      xxmsg--'

      SS/&
      SSSS EOJ
```

Run Internal REXX

```
FLUSH1Q52I 1Q52I
FLUSHCPU ARX0998I
REPLY HIT ENTER TO CLOSE THE FILE
REPLY HALT EXIT REACHED, PRESS ENTER TO END
```

```
ARG lockid, message, tblentry
rc = 0
PARSE SOURCE . . rexxname . /* Get the REXX program name */
position = POS('-', WORD(message, 1)) /* Did we get something like: */
IF position > 0 THEN /* R1-47 // PAUSE */
DO /* Extract from message... */
replyid = SUBSTR(WORD(message, 1), position+1) /* Reply id */
pid = SUBSTR(WORD(message, 1), 1, position-1) /* Partition id */
END
ELSE DO
replyid= WORD(message, 2) /* Reply id */
pid = WORD(message, 1) /* Partition id */
END
event = WORD(tblentry, 1) /* Extract event from table entry */
```

Run Internal REXX

```
CALL SLEEP 1 /* Sometimes needed for.. */
/* .. synchronisation */
SIGNAL VALUE event /* Branch to label (=Action) */
/*-----*/
/* Actions are handled here */
/*-----*/
/*-----*/
REPLY: /* Just give a Reply */
/*-----*/
CALL ASSGN 'STDOUT', 'SYSLOG'
SAY rexxname': REXX CONSOLE APPLICATION GIVES AUTOMATIC REPLY TO' replyid
CALL ASSGN 'STDOUT', 'SYSLSST'
CALL SENDCMD replyid
RETURN 0
/*-----*/
FLUSH1Q52I:/* FLUSH partition due to 1Q52I */
/*-----*/
IF pos('1Q52I', message) > 0 THEN /* Ok, we got valid message */
DO
```

Run Internal REXX

```
str = EXTRACT(message,'OUTPUT LIMIT EXCEEDED FOR',',')
jname = WORD(str,1)
jnum = WORD(str,2)
jpid = WORD(str,3)          /* Get the partition id */
CALL SENDCMD 'PFLUSH' jpid  /* Flush the partition */
RETURN 0                    /* SUCCESS */
END
/* ERROR: We don't get the */
RETURN 100                 /* expected message */
/*-----*/
FLUSHCPU: /* FLUSH partition due to ARX0998I */
/*-----*/
IF pos('ARX0998I',message) > 0 THEN /* ok, we got valid message */
DO
  jpid = WORD(message,5)          /* Get the partition id */
  jname = WORD(message,7)        /* Get the job name */
  str = EXTRACT(message,'EXCEEDS LIMITS:') /* Get system activity info*/
  no_of_words = WORDS(str)
  elapsed = ''; cpuTime = '';    /* Initialize to null string */
  io      = ''; ioRate = ''; cpuPercent = ''
```

Run Internal REXX

```
DO i=1 TO no_of_words          /* Parse the operands and the */
  iword = WORD(str,i)          /* values for each */
  PARSE VAR iword operand '=' value /* system activity info item */
  SELECT
    WHEN( operand = 'ELAPSED' ) THEN elapsed = value
    WHEN( operand = 'CPUTIME' ) THEN cpuTime = value
    WHEN( operand = '%CPU' ) THEN cpuPercent = value
    WHEN( operand = 'I/O' ) THEN io = value
    WHEN( operand = 'IORATE' ) THEN ioRate = value
    OTHERWISE nop
  END /*select*/
END
IF ( WORDPOS(jname,'REXXCONS REXXTRY POWSTART CICSICCF VTAMSTRT') = 0 ,
  &
  jname ^= 'NO NAME ' , /* Check JOB NAMES and */
  & /* PARTITION IDs */
  WORDPOS(jpid,'F1 F2 F3') = 0 , /* that must not be flushed */
  ) /* */
THEN /* job name is OK... */
```

Run Internal REXX

```
DO
CALL SENDCMD 'PFLUSH' jpid          /* ...PFLUSH the partition */
                                  /* Write LOG entry into */
                                  /* job output */
SAY rexxname 'has PFLUSHed JOB='jname 'PID='jpid 'at DATE='date() ,
'TIME='time() 'due to CPUTIME='cpuTime '%CPU='cpuPercent ,
'I/Os='io 'I/ORATE='ioRate
END
RETURN 0                          /* SUCCESS */
END                                /* END ARX0998I ok */
RETURN 100                        /* ERROR: We don't get the
                                  expected message */
EXTRACT: /* Auxiliary Subroutine, extracts data from an input msg */
/*-----*/
ARG msg,sep1,sep2
PARSE VAR msg part1 (sep1) extract_str (sep2) part2
RETURN extract_str
```