5 The TSO/E REXX commands

In addition to the normal TSO commands, there are some commands that execute the TSO/E specifically for REXX. These commands are like all TSO commands, using literals to send them to the TSO. Some of these commands are only executable when called up from a REXX procedure. I would like to present the most important commands here.

Note

In REXX on other platforms, these commands are not available. For example, the EXECIO command is not available in OOREXX. This is really a deficit of OOREXX!

5.1 EXECIO – READ AND WRITE DATA SETS

This is the most widely used TSO/E REXX command. EXECIO uses the QSAM method of z/OS. This means that all advantages and disadvantages of this access method are also applicable for EXECIO.

Remarks:

– To run an EXECIO command, a DD name must exist that is named in the EXECIO command. When running a REXX procedure in a batch job, this can also be done by a DD statement in the JCL.

– The EXECIO command can read data from the data stack as well as write data to the data stack.

– The EXECIO command can read the data into a stem as well as write data from a stem. In this case, the name of the stem must be specified in the parameter field.

– The file specified with the DD statement can also be a member of a PDS or PDSE.

– When writing data records that are longer than the LRECL of the file, the records will be truncated and EXCECIO completes the operation without an error being ex- plicitly reported. **However, in this case, the return code 1 is returned by the command EXECIO.**

– Records can be individually read and written. However, this is rather rare in practice, as this would slow down the process.

Format:

"EXECIO lines iotyp ddname (parms"

lines	Specifies the number of records to be read or written: 0 → no records are written or read. * → all records are written or read. n → number of records that are written or read.
iotyp	**DISKR Read operation.** **DISKW Write operation.**
ddname	This is the name of an already allocated DD

Find answers on the fly, or master something new. Subscribe today. See pricing options.

is written or into which is read.
OPEN The data set shall be opened. OPEN will automatically be performed when records are read or written. Normally, this option is never used.
FINIS The file will be closed when the I/O operation ends.

Program 5.1: EXECIOT – Example for the TSO function EXECIO

```
01 /* DOC: REXX EXECIOT                                                */
02 /* DOC: Example of the application of the command EXECIO          */
03 /* © FRANZ LAHL 2015                                              */
04 /*****************************************************************/
05 "alloc dd(out) dsn('PROX.BOOK.REXX(EXECIOD)') shr reuse"
06 "execio 0 diskw out (open finis"
07 a.1 = copies("This is the line1",10) /* a.1 is has now 170 chars.*/
08 a.2 = "This is the line2"
09 a.3 = "This is the line3"
10 a.4 = "This is the line4"
11 a.5 = "This is the line5"
12 a.6 = "This is the line6"
13 a.7 = "This is the line7"
14 a.8 =
15 a.0 = 8
16 "execio "a.0" diskw out (stem a. finis"
17 a.0 = 0
18 "execio * diskr out (stem a. finis"
19 a.5 = "aaaaaa"
20 a.6 = "bbbbbb"
21 "execio "a.0" diskw out (stem a. finis"
22 "free dd(out)"
```

The following table contains descriptions of the statements of this small test program:

Lines	Explanation
05	Allocation of member PROX.BOOK.REXX(EXECIOD) as a sequential file named DD OUT.
06	Empty the file without deleting it.
07-14	Assigning lines to Stem **A.** The line A.1 is much longer than the LRECL 80 of the data set.
15	Store the number of records in stem **A.**
16	Write stem A. into member EXECIOD and CLOSE the data set.
17	Set A.0 to zero.
18	Read all records of the file. A.0 now contains the number of records, namely 8
19-20	Change records 5 and 6 in stem A.
21	Write stem A. again into member EXECIOD and CLOSE the data set.
22	Free the DD name OUT. The data set PROX.BOOK.REXX(EXECIOD) is also freed.

Here you can now see the contents of the member EXECIOD after the above program is running:

```
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8
******************************** Top of Data ********************************
This is the line1This is the line1This is the line1This is the line1Thi is the L
This is the line2
This is the line3
This is the line4
aaaaaa
bbbbbb
This is the line7
****************************** Bottom of Data ******************************
```

As you can see, the first record was shortened to LRECL=80 when writing the member EXECIOD.

5.2 DELSTACK – DELETE DATA STACK CONTENTS

Function:

This command will delete data stacks.

- Deletes the last opened data stack.
- If no data stack was open then the original data stack will be deleted.

Find answers on the fly, or master something new. Subscribe today. See pricing options.

5.3 DROPBUF – DELETE DATA STACK BUFFERS

Function:

Use this command to delete data stack buffers that were previously created. The command has one operand. This indicates what number of the buffer was cleared. If no operand is specified, the last with MAKEBUF created buffer will be cleared.

5.4 TSO COMMANDS

All TSO commands are usable in REXX programs. In addition to the TSO commands discussed so far, I will now explain some more TSO commands that are very useful for programming REXX applications. Important commands are ALLOC, FREE, LISTDS, SUBMIT, EXEC.

Rules:

– As discussed above, all strings which are found in REXX programs and are not recognized by the REXX interpreter, will passed to the currently active **host command environment** for execution.

– When a REXX program starts, then the host command environment for TSO is active. Therefore, an ADDRESS TSO command is not necessary in this case. As long as you do not switch the host command environment to another subsystem, can you execute TSO commands simply by writing them in a program line surrounded by apostrophes.

Example 1: Simple allocation of an existing data set.

Example 3: The TSO command LISTCAT is used to print a catalog extract.

```
"alloc dd(in) dsn('LANZT.LOGON.CLIST') shr reuse"
../* Data set processing */...
"free dd(in)"
```

Example 2: Create and assign a PDSE as a new ISPPROF data set.

```
"alloc dd(ISPPROF)",
        "dsn('LANZT.LOGON.ISPPROF.SYS1') new catalog",
        "unit(sysda) storclas(ispfpool)",
        "dsntype(library) recfm(f b) lrecl(80)",
        "blksize(3200) space(5 5) cylinders dir(10)"
```

Example 3: The TSO command LISTCAT is used to print a catalog extract.

```
Ret = outtrap(list
"listcat entry('PROX.USER.REXX') all"
ret = outtrap("OFF")
do I = 1 to list.0
   say list.i
end i
```

If this small program is executed, the following list will appear:

```
NONVSAM ------- USER001.TEST2.REXX
     IN-CAT --- UCAT.USER001
     HISTORY
       DATASET-OWNER-----(NULL)      CREATION--------2015.135
       RELEASE----------------2      EXPIRATION------0000.000
       ACCOUNT-INFO-----------------------------------(NULL)
       DSNTYPE----------LIBRARY
     SMSDATA
       STORAGECLASS ----XYZBASE      MANAGEMENTCLASS--XYZMGMT
       DATACLASS --------(NULL)      LBACKUP ---0000.000.0000
     VOLUMES
       VOLSER-----------XYZU11       DEVTYPE------X'3010200F'
     ASSOCIATIONS--------(NULL)
```

I truncated some lines on the right.

Remark:

Generally, all commands of the **Access Method Services** are usable as TSO commands and thus also with REXX.

See manual: **z/OS DFSMS Access Method Services for Catalogs**

Recommended / Playlists / History / Topics / Settings / Get the App / Sign Out

PREV
4 The REXX functions

NEXT
6 Execute REXX programs

Find answers on the fly, or master something new. Subscribe today. See pricing options.