BREXX/370 V2R2M0 VSAM User's Guide

Document Version 1.0
Authors: Peter Jacob (pej), Mike Großmann (mig)

The VSAM User's Guide contains the BREXX functions to access VSAM KSDS files.

The VSAM Interface is based on Steve Scott's VSAM API: https://sourceforge.net/projects/rxvsam/ .

We gratefully thank Steve for allowing us the integration in BREXX and his support to achieve it.

The underlying VSAM API allows full support for KSDS, RRDS and ESDS, but we focused just on the KSDS functionality, so there is no support for RRDS and ESDS. If this limitation will be lifted in the future depends on user requests.

I. Integration of the VSAM Interface in BREXX

We decided to integrate the interface as host commands rather than BREXX functions. It is now similar to the EXECIO host command for sequential datasets. The host command name is VSAMIO. Host commands are typically enclosed in quotes or double-quotes.

Example:

"VSAMIO OPEN VSIN (UPDATE"

1. Limitations/Restrictions

The implementation has only tested with UNIQUE cluster definitions, not with type SUBALLOCATION (which requires in MVS 3.8 a DEFINE SPACE and DEFINE VSAM catalogue definition). The UNIQUE specification does not allow the REUSE CLUSTER definition, which would be necessary for the initial loading of an empty KSDS dataset.

2. Initialising empty VSAM Files

An empty VSAM file cannot be directly processed by a program it must be initialised first. The procedure to achieve this is to use IDCAMS REPRO to write a "null"-record into it. After this exercise, the VSAM file can be updated by BREXX/370 with normal VSAM Write commands from BREXX.

If we find a better solution, we will integrate it in the next releases.

3. Key of Records

The key must be part of the record, nevertheless you must additionally specify the key in the following commands:

- "VSAMIO READ *ddname* (KEY *key … "*
- "VSAMIO LOCATE *ddname* (KEY *key … "*
- "VSAMIO WRITE *ddname* (KEY *key … "*
- "VSAMIO DELETE *ddname* (KEY *key … "*

The key of a record must consist of a sequence of contiguous non-space characters. This means blanks are not allowed being part of a key. This limitation might be lifted in one of the forthcoming releases. You can easily convert spaces in a key with the TRANSLATE function:

```
key=translate(key,'_',' ')
```

4. Return Codes

Each VSAMIO command call returns two return codes:

RC the usual rc return code, containing
- 0 call was successful
- 4 call was not successful and ended with warnings, typically in record-not-found Situations
- 8 call ended with errors

RCX is the extended VSAM Return code, it consists of a 9 character field with the following format **rrr-vvvvv** rrr is the function return code, vvvvv is the VSAM return code

you can look up the details of the extended VSAM return code in IBM's MVS System Messages under message IDC3351I.

5. System Abend A03

The RXVSAM API runs as independent subtask within the address space. By end of the REXX Script, an automatic shutdown of the subtask will be performed. If the REXX script terminates unexpectedly you will maybe see a SYSTEM ABEND A03, which means the main task (BREXX) has been terminated and there is still a subtask active in the background. MVS forces the ABEND of the subtask with A03. There are no further actions required, there is no impact on the system or the VSAM datasets.

6. Random and Sequential Access

The used VSAM IO module distinguishes two access methods:

- Random Access always requires a key to read/write/delete a record
- Sequential Access allows to position to a certain record and read/write/delete records from there sequentially

Both methods can be used concurrently, but it is important to understand they do not mutual interfere. Having read a record with random access does not allow to read from this record sequentially the next records, as this is sequential access. But you can perform a LOCATE command with a key and continue the read from there sequentially.

7. VSAM Dataset reference

Each VSAMIO command uses the DDNAME as a reference to the VSAM dataset. It must be pre-allocated via a JCL DD Statement or a TSO ALLOCATE command.

There are no plans to allow a dataset name (DSN) instead of the DDNAME!

8. REXX VSAM Debugging

By using BREXXDBG as the BREXX interpreter you can produce additional log entries in the operator's console, as well as in the spool output of a batch job:

JCL:

```
//BRXVSMKY JOB CLASS=A,MSGCLASS=H,REGION=8192K,
//         NOTIFY=&SYSUID
//*
//*RELEASE   SET 'V2R2M0'
//* ... BREXX          Version V2R2M0 Build Date 23. Oct 2019
//* ... INSTALLER DATE 23 Oct 2019 15:30:36
//* ---------------------------------------------------------------
//* READ STUDENT VSAM FILE VIA KEY
//* ---------------------------------------------------------------
//*
//BATCH EXEC RXTSO,BREXX='BREXXDBG',
//         EXEC='$STUDENK',
//         SLIB='BREXX.V2R2M0.SAMPLES'
//SYSPRINT DD  SYSOUT=*,
//             DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133)
//SYSUDUMP DD  SYSOUT=*
//
```

Spool Output:

```
                                                J E S 2   J O B   L O G
07.35.01 JOB 1466  $HASP373 PEJRXKEY STARTED - INIT  1 - CLASS A - SYS
TK4-
07.35.01 JOB 1466  IEF403I PEJRXKEY - STARTED - TIME=07.35.01
07.35.02 JOB 1466  +VSAMIO - STUDENTM ACCESS TRACE, REQUEST = OPEN
07.35.02 JOB 1466  +VSAMIO - KEY=NONE
07.35.02 JOB 1466  +VSAMIO - STUDENTM ACCESS TRACE, REQUEST = READU
07.35.02 JOB 1466  +VSAMIO -
KEY=X"C1D5C4C5D9E2D6D55EC2C5D56D6D6D6D6D6D6D6D6D6D6
07.35.02 JOB 1466  +VSAMIO - STUDENTM ACCESS TRACE, REQUEST = READU
07.35.02 JOB 1466  +VSAMIO -
KEY=X"C1D5C4C5D9E2D6D55EC7C1C2D9C9C5D36D6D6D6D6D6D6
07.35.02 JOB 1466  +VSAMIO - STUDENTM ACCESS TRACE, REQUEST = READU
07.35.02 JOB 1466  +VSAMIO -
KEY=X"C2C1D3C4E6C9D55EC1D9D3C5D5C56D6D6D6D6D6D6D6D6
07.35.02 JOB 1466  +VSAMIO - STUDENTM ACCESS TRACE, REQUEST = READU
07.35.02 JOB 1466  +VSAMIO -
KEY=X"E2E3C5D7C8C5D5E2D6D55ED7C1E3D9C9C3C9C16D6D6D6
07.35.02 JOB 1466  +VSAMIO - STUDENTM ACCESS TRACE, REQUEST = CLOSE
07.35.02 JOB 1466  +VSAMIO - KEY=NONE
07.35.02 JOB 1466  IEFACTRT - Stepname  Procstep  Program   Retcode
07.35.02 JOB 1466  PEJRXKEY   BATCH     EXEC      IKJEFT01  RC= 0000
07.35.02 JOB 1466  IEF404I PEJRXKEY - ENDED - TIME=07.35.02
07.35.02 JOB 1466  $HASP395 PEJRXKEY ENDED
```

II. VSAM Commands in BREXX

1. OPEN VSAM Dataset

"VSAMIO OPEN *ddname* **([***READ/UPDATE***]> "**

Example:

```
"VSAMIO OPEN VSIN1 (READ"
"VSAMIO OPEN VSIN2 (UPDATE"
```

VSIN1 is opened in reading mode, VSIN2 in UPDATE mode.

2. READ with KEY

Access-Type: Random

"VSAMIO READ *ddname* **(KEY** *key-to-read* **VAR** *rexx-variable***"**

If you want to update the record you must prepare for it by adding the UPDATE keyword

"VSAMIO READ *ddname* **(KEY** *key-to-read* **UPDATE VAR** *rexx-variable***"**

The UPDATE keyword requires a File OPEN with UPDATE

Example:

```
"VSAMIO READ VSIN1 (KEY "key1" VAR record1"
"VSAMIO READ VSIN2 (KEY "key2" UPDATE VAR record2"
```

Read a record with key1/key2 (contained in a rexx variable) into the rexx variable record1/record2

3. READ NEXT

Access-Type: Sequential

After positioning with LOCATE to a certain record, you can read sequentially the next records. If no LOCATE has been previously performed, the first record is read.

"VSAMIO READ *ddname* **(NEXT VAR** *rexx-variable***"**

If you want to update the record you must prepare for it by adding the UPDATE keyword

"VSAMIO READ *ddname* **(NEXT UPDATE VAR** *rexx-variable***"**

The UPDATE keyword requires a File OPEN with UPDATE

Example:

```
"VSAMIO LOCATE VSIN (KEY "key
Do until rc>0
  "VSAMIO READ VSIN (NEXT VAR record"
   Say record
End
```

Position to record key (contained in a rexx variable) and read all records from there into rexx variable record

4. LOCATE position to a certain record

Access-Type: Sequential

Position the record pointer in front of the provided key or a key prefix

"VSAMIO LOCATE *ddname* **(KEY** [*key-to-position/key-prefix*]**"**

To subsequentially read the next records a READ NEXT is required. After a successful read, the position is shifted to the next record position

Example refer to READ NEXT:

5. WRITE KEY

Access-Type: Random

To update a record it must be priorly read with a READ KEY, regardless whether the record exists. If the record doesn't exist, it will be inserted.

"VSAMIO WRITE *ddname* **(KEY key-to-write VAR** *rexx-variable***"**

Example:

```
"VSAMIO READ  VSIN (KEY "key"  UPDATE VAR CURRENT"
 say 'READ 'rc' Extended RC 'rcx
"VSAMIO WRITE VSIN (KEY "key"  VAR RECORD"
if rc<>0 then say key' Error during Insert'
   else say inkey' Record inserted'
say 'WRITE 'rc' Extended RC 'rcx
```

Insert a new record, the READ is mandatory to verify if a record is already defined.

6. WRITE NEXT

Access-Type: Sequential

To update a record it must be priorly read with a READ NEXT.

"VSAMIO WRITE *ddname* **(NEXT VAR** *rexx-variable***"**

7. DELETE KEY

Access-Type: Random

To delete an existing record.

"VSAMIO DELETE *ddname* **(KEY key-to-delete "**

Example:

```
"VSAMIO OPEN VSERR (UPDATE"
 say 'OPEN 'rc' Extended RC 'rcx
"VSAMIO DELETE VSERR (KEY 0000000"
```

```
 say 'Delete Dummy Record 'rc' Extended RC 'rcx
```

8. DELETE NEXT

Access-Type: Sequential

To delete an existing record it must be priorly read with a READ NEXT.

"VSAMIO DELETE *ddname* **(NEXT "**

Example:

```
"VSAMIO LOCATE VSIN (KEY "prefix
 say "LOCATE "rc
 say "Extended RC "rcx
 do forever
    "VSAMIO READ VSIN (NEXT UPDATE VAR INREC"
     if rc<>0 then leave
     say "record='"INREC"' RC "rc" Extended RC "rcx
     key=substr(inrec,1,8)
    "VSAMIO DELETE VSIN (NEXT "
     if rc=0 then reci=reci+1
   say 'DELETE RC 'rc' Extended RC 'rcx
  end
```

9. CLOSE

"VSAMIO CLOSE *ddname* **"**

To close all open VSAM datasets you can also use

"VSAMIO CLOSE ALL "

Example:

```
"VSAMIO CLOSE VSERR"
```

III. BREXX VSAM Example

The installation file contains in the dataset **BREXX.V2R2M0.JCL** a working example of a student database using fictitious student entries, containing first name, family name, birth date, field of study, address.

You can submit the REXX scripts in batch out of **BREXX.V2R2M0.JCL**

STUDENTC Creates the VSAM Cluster definition

STUDENTI Inserts the student records into the VSAM dataset

STUDENTK Read the VSAM dataset with KEYs

STUDENTN Read the VSAM dataset sequentially

The REXX scripts are stored in **BREXX.V2R2M0.SAMPLES**

@STUDENI insert student records

@STUDENK read student records by key

@STUDENL Query student records by using formatted screens

@STUDENN read student records sequentially

The following examples illustrates the definition and population of a VSAM dataset using BREXX:

1. Define a VSAM Cluster

Define a new VSAM Cluster and import a "Null"-Record

```
//PEJDEFC JOB CLASS=A,MSGCLASS=H,REGION=8192K,
//         NOTIFY=&SYSUID
//* ------------------------------------------------------------------
//* STEP 1 CREATE NEW CLUSTER DEFINITION
//* ------------------------------------------------------------------
// EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD  SYSOUT=*
//FIRSTREC DD  *
0000000000NULL RECORD
/*
//SYSIN    DD  *
  DELETE 'PEJ.KSDS'
  DEFINE CLUSTER                              -
      ( NAME('PEJ.KSDS') INDEXED         -
           RECSZ(16 200) KEYS(8 0)            -
           TRACKS(60 30)                      -
           VOLUMES(PUB002)                    -
           SHAREOPTIONS(2 3) UNIQUE           -
      )                                       -
      DATA (NAME('PEJ.KSDS.DATA') )    -
      INDEX (NAME('PEJ.KSDS.INDEX') )
  REPRO INFILE(FIRSTREC) ODS('PEJ.KSDS')
/*
```

2. Job Output Define Cluster

```
09.45.42 JOB 1203  $HASP373 PEJRXBTH STARTED - INIT  1 - CLASS A - SYS TK4-
09.45.42 JOB 1203  IEF403I PEJRXBTH - STARTED - TIME=09.45.42
09.45.44 JOB 1203  IEFACTRT - Stepname  Procstep  Program   Retcode
09.45.44 JOB 1203  PEJRXBTH                       IDCAMS    RC= 0000
09.46.48 JOB 1203  PEJRXBTH   BATCH     EXEC      IKJEFT01  RC= 0000
09.46.48 JOB 1203  IEF404I PEJRXBTH - ENDED - TIME=09.46.48
09.46.48 JOB 1203  $HASP395 PEJRXBTH ENDED
    1     //PEJRXBTH JOB CLASS=A,MSGCLASS=H,REGION=8192K,
          //         NOTIFY=PEJ,
          //            USER=PEJ,PASSWORD=               GENERATED BY IKJEFF10
          *** ------------------------------------------------------------
          *** CREATE AND LOAD VSAM FILE
          *** ------------------------------------------------------------
          *** ------------------------------------------------------------
          *** STEP 1 CREATE NEW CLUSTER DEFINITION
          *** ------------------------------------------------------------
    2     // EXEC PGM=IDCAMS,REGION=512K
    3     //SYSPRINT DD  SYSOUT=*
    4     //FIRSTREC DD  *
    5     //SYSIN    DD  *
IEF375I  JOB /PEJRXBTH/ START 19256.0945
IEF376I  JOB /PEJRXBTH/ STOP  19256.0946 CPU    0MIN 41.41SEC SRB    0MIN 03.17S
IDCAMS  SYSTEM SERVICES                                        TIME: 09:45:42

  DELETE 'PEJ.KSDS'                                                  00001500
IDC0550I ENTRY (D) PEJ.KSDS.DATA DELETED
IDC0550I ENTRY (I) PEJ.KSDS.INDEX DELETED
IDC0550I ENTRY (C) PEJ.KSDS DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  DEFINE CLUSTER                          -                          00001600
      ( NAME('PEJ.KSDS') INDEXED        -                          00001700
           RECSZ(16 200) KEYS(8 0)      -                          00001800
           TRACKS(60 30)                -                          00001900
           VOLUMES(PUB002)              -                          00002000
           SHAREOPTIONS(2 3) UNIQUE     -                          00002100
      )                                 -                          00002200
      DATA (NAME('PEJ.KSDS.DATA') )    -                          00002300
      INDEX (NAME('PEJ.KSDS.INDEX') )                             00002400
IDC0508I DATA ALLOCATION STATUS FOR VOLUME PUB002 IS 0
IDC0509I INDEX ALLOCATION STATUS FOR VOLUME PUB002 IS 0
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

  REPRO INFILE(FIRSTREC) ODS('PEJ.KSDS')                             00002502
IDC0005I NUMBER OF RECORDS PROCESSED WAS 1
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
```

3. Sample BREXX Program to update the VSAM Dataset

```
/* REXX */
/* ---------------------------------------------------------------------
 * Insert Records in a VSAM Dataset
 * ---------------------------------------------------------------------
 */
ADDRESS TSO
"ALLOC FILE(VSIN) DSN('PEJ.KSDS') SHR"
say 'ALLOC 'rc
/* ---------------------------------------------------------------------
```

```
 * OPEN VSAM Mode Update
 *  KEY Length = 8
 * ---------------------------------------------------------------------
 */
keylen=8
"VSAMIO OPEN VSIN (UPDATE"
say 'OPEN 'rc' Extended RC 'rcx
"VSAMIO DELETE VSIN (KEY 00000000"
say 'Delete Dummy Record 'rc' Extended RC 'rcx
/* ---------------------------------------------------------------------
 * INSERT Records
 * ---------------------------------------------------------------------
 */
letters="ABCDFGHIJKLMNOPQRSTUVWXYZ"
letlen=length(Letters)
reci=0
do i=1 to letlen
   pref=substr(letters,i,1)substr(letters,i,1)
   do for 250
      reci=reci+1
      key=pref''right(reci,keylen-2,'0')
      record='This is Record 'reci
      call insert key,record
   end
end
/* ---------------------------------------------------------------------
 * CLOSE VSAM Dataset
 * ---------------------------------------------------------------------
 */
"VSAMIO CLOSE VSIN"
say 'CLOSE 'rc' Extended RC 'rcx
ADDRESS TSO
"FREE FILE(VSIN)"
say 'FREE 'rc
say 'Records Inserted 'reci
return 0
/* ---------------------------------------------------------------------
 * Insert new Key
 * ---------------------------------------------------------------------
 */
insert:
parse arg inkey,inrec
  inrec=inkey''inrec
  "VSAMIO READ  VSIN (KEY "inkey"  UPDATE VAR CURREC"
   if rc=0 then do
      say inkey' Record already defined'
      say "'"currec"'"
   end
   say 'READ 'rc' Extended RC 'rcx
  "VSAMIO WRITE VSIN (KEY "inkey"  VAR INREC"
  if rc<>0 then say inkey' Error during Insert'
     else say inkey' Record inserted'
   say 'WRITE 'rc' Extended RC 'rcx
return rc
```

4. JCL Upate VSAM Dataset

The BREXX Program is updating the new VSAM Dataset.

```
//PEJVSUPD JOB CLASS=A,MSGCLASS=H,REGION=8192K,
//         NOTIFY=&SYSUID
//* ------------------------------------------------------------------
//* STEP 2 INSERT RECORDS INTO VSAM FILE
//* ------------------------------------------------------------------
//BATCH EXEC RXTSO,BREXX='BREXXSTD',
//         EXEC='VSMINSRT',
//         SLIB='PEJ.EXEC'
//*        EXEC='REPRO',
//SYSPRINT DD  SYSOUT=*,
//             DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133)
//SYSUDUMP DD  SYSOUT=*
//
```

5. Job Output Update VSAM Dataset

```
IEF373I STEP /EXEC    / START 19256.0945
IEF374I STEP /EXEC    / STOP  19256.0946 CPU    0MIN 40.68SEC SRB    0MIN 03.02S
********************************************************************************
*     1. Jobstep of job: PEJVSUPD    Stepname: EXEC       Program name: IKJEFT01
*         elapsed time  00:01:03,77                       CPU-Identifier:  TK4-
*             CPU time  00:00:43,70              Virtual Storage used:   1040K
*           corr. CPU:  00:00:43,70   CPU time has been corrected by  1 / 1,0  m
*
*     I/O Operation
*     Number of records read via DD * or DD DATA:      0
*     240.......2 240.......0 241.......0 191.......0 241.......0 DMY.......0 DM
*     DMY.......0 DMY.......0
*
*                                          Charge for step (w/o SYSOUT):
********************************************************************************
ALLOC 0
OPEN 0 Extended RC 000-00000
Delete Dummy Record 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
AA000001 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
AA000002 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
AA000003 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
AA000004 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
AA000005 Record inserted
…
…
READ 4 Extended RC 004-00004
ZZ006247 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
ZZ006248 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
ZZ006249 Record inserted
WRITE 0 Extended RC 000-00000
READ 4 Extended RC 004-00004
ZZ006250 Record inserted
WRITE 0 Extended RC 000-00000
CLOSE 0 Extended RC 000-00000
FREE 0
Records Inserted 6250
```

6. Using a Formatted Screen Application to Query the Student File

TSO RX "BREXX.V2R2M0.SAMPLE(@SUTDENTL)"

```
------------------------ Student Information System ----------------------

Last Name / Prefix ===> fin             First Name         ===> _
                                        Birth Date         ===> _

Study              ===> _

City               ===> _
```

Result:

```
CMD ==> _____________________________________________ ROWS 00001/00007 COL 001 B01
..... First Name     Surname       Sex Birth Date Study                   City
..... ---------------------------------------------------------------------------
00001 Esme           Findlay        f  12/06/1994 Mechanical Engineering Stoke-
00002 Edward         Finlayson      m  26/03/1997 Physics                 Northa
00003 Elliot         Finlayson      m  22/12/1993 Electrical Engineering London
00004 Jonathan       Finnegan       m  04/12/1994 Economics               Southe
00005 Abi            Finnie         f  18/06/1999 Economics               Grays
00006 Ivy            Finnie         f  29/12/1994 Physics                 Cookst
00007 Matthew        Finnigan       m  23/07/1993 Philosophy              Barnsl
***** ***************************** End of Data ******************************
```

BREXX/370 V2R2M0 VSAM User's Guide

Table of Contents