BREXX KEY/VALUE Database

Document Version 5.3
Authors: Peter Jacob (pej), Mike Großmann (mig)

Introduction

Accessing the underlying VSAM KSDS is made simple by using the Key/Value Database. You do not require extensive VSAM understanding nor do you need to be familiar with the database's intricate structure. The Key/Value Database is standardised.

The system consists of two VSAM files: a source and a reference database. The reference DB facilitates source record linking of the K/V database. This makes it possible to implement simple databases like hierarchical or graph databases.

Overview of the VSAM Key structure

The K/V functions handle everything; memorizing the VSAM structure is not required, but it should not be overlooked for completeness.

The VSAM structure in an overview, the partition, which will be called the room, is 2 bytes long, the qualifier is 10 bytes long, and the key itself is 32 bytes long, making the total key length 44 bytes. Therefore, having the same key but different qualifiers is possible. Being in multiple rooms allows you to make use of the same key and qualification.

Any changes you make to these default settings will necessitate modifications in the BREXX module DBPROF, which is stored in your RXLIB.

To address records, use the key element, for example:

```
Call DBSET([qualifier.]key, record)
Call DBGET([qualifier.]key)
Say Result
```

The qualifier is always converted to **lowercase**. The key is case-sensitive unless you set in your database profile (usually DBPROF) **ddprof.keyupper to 1**

The qualifier is optional and, if not specified, will be set to **any.** If the qualifier and key are shorter than their definitions, they will be suffixed with "_" to achieve the maximum length feasible.

Installing the Key/Value Database Functions

The Key/Value program is included with BREXX V2R5M3 and does not require any additional installation. However, you must configure and submit the VSAM Clusters in the member $CREKEYV in the BREXX installation dataset. For more details, see Step 6 in the BREXX installation dataset manual.

Key/Value Modules

Library content:

DBOPEN	Key/Value REXX contains all procedures
DBPROF	Key/Value profile with essential information
SAMPLE1	Key/Value sample to test it
$CREKEYV	as part of the BREXX installation library

Customisation (optional)

Aside from the fundamental parameters (names and lengths), the PDS member DBPROF allows you to modify the behaviour. Especially you can instruct the DBLINK function to construct dummy source records if one of the records to be connected does not exist. Another option is to have it translate the key into uppercase. This is the default setting.

```
…
…
  ddprof.EnableDummy=1              /* allow links between records   */
                                    /* which do not exist, they will */
                                    /* create DUMMY source records   */
                                    /* 0: no, 1: yes                 */

   ddprof.keyupper=0                 /* upper case translation of the */
                                     /* key. 0: no, 1: yes            */
```

Loading the sample Key/Value Database

A sample is included in the BREXX installation and can be loaded using the REXX:

REXX.V2R5M3.SAMPLE(DBWORLD).

It includes all countries in the world, a few trade unions, and the country's main cities. It is used in the following examples to demonstrate how the K/V and reference databases work.

```
KV160I    Check-in Standard Room  (AA)
KV120I    Key/Value DB successfully opened
10:25:20.626631 Start loading Countries
10:25:29.178898 Loading Countries completed, 203 loaded
10:25:29.180352 Link them to Trade Unions
10:25:31.231292 Link to Trade Unions completed
10:25:31.603774 Start loading Cities and Link them to their Country
10:25:51.079262 loading Cities completed, 774 loaded
```

Key/Value Database Functions

To use the Key/Value database functions you need to import KEYVALUE (part of your RXLIB).

```
   CALL IMPORT KEYVALUE
```

Let us start with a simple example, it should be self-explanatory:

```
  call import keyvalue
  say "OPEN  "DBOPEN()

/* Add Continents  */
   Call DBSET('Continent.Europe','Continent Europe')
   Call DBSET('Continent.Asia','Continent Asia')
   Call DBSET('Continent.Africa','Continent Africa')
   Call DBSET('Continent.North America','Continent North America')
   Call DBSET('Continent.South America','Continent South America')
   Call DBSET('Continent.Australia','Australia and Oceania')

/* Get Continents  */
   call DBGET('Continent.Europe')
   say dbresult
   call DBGET('Continent.Asia')
   say dbresult
   call DBGET('Continent.Africa')
   say dbresult
   call DBGET('Continent.North America')
   say dbresult
   call DBGET('Continent.South America')
   say dbresult
   call DBGET('Continent.Australia')
   say dbresult
   say "CLOSE "DBCLOSE()
 exit
```

Result

```
KV160I    Check-in Standard Room  (AA)
KV120I    Key/Value DB successfully opened
OPEN 0
Continent Europe
Continent Asia
Continent Africa
Continent North America
Continent South America
Australia and Oceania
CLOSE 0
```

… and now the available functions follow in detail.

DBOPEN() Opens the key/value VSAM dataset.

Opens the standard Key/Value Database. If DBOPEN was successful an RC of zero is returned, else it failed.

```
  say "OPEN  "DBOPEN() ->  OPEN  0
```

DBROOM(room-name) Assigns a room

The DBROOM function divides the Key/Value Database into distinct areas that can be projects, applications, or anything else. This means that you have many Key/Value Databases in a single VSAM dataset.

The definition of a room is optional if not defined it automatically switches into a standard room. Use the **DBROOM** command to define a new room or to reuse an existing one. If the room is not already there, it will be created and recorded in the Key/Value database's control record. There will always be an uppercase translation of the room name.
By designating a room, you can specify the space where the following commands take effect.
You can define the same Key of a record more than once by using various rooms. Since they cannot see one another, rooms cannot communicate.

For example, write records (with the same key) in different rooms and report them.

```
call import 'KeyValue'
call DBOPEN
call DBROOM 'Beverages'
say '*** Room Beverages ***'
call DBSET('Beer','Munich Hofbraeuhaus')
call DBSET('Whisky','Black Label')
call DBGET('Whisky')
   say left(dbkey,8)': ' dbresult
call DBGET('Beer')
   say left(dbkey,8)': ' dbresult
call DBROOM 'Booze'
say '*** Room Booze     ***'
call DBSET('Beer','Guiness')
call DBSET('Whisky','Bushmills')
call DBGET('Whisky')
   say left(dbkey,8)': ' dbresult
call DBGET('Beer')
   say left(dbkey,8)': ' dbresult
call DBROOM 'Beverages'
say '*** Room Beverages ***'
call DBGET('Whisky')
   say left(dbkey,8)': ' dbresult
call DBGET('Beer')
```

```
   say left(dbkey,8)': ' dbresult
call dbclose
```

Result

```
KV160I    Check-in Standard Room  (AA)
KV120I    Key/Value DB successfully opened
*** Room Beverages ***
Whisky  :  Black Label
Beer    :  Munich Hofbraeuhaus
*** Room Booze     ***
Whisky  :  Bushmills
Beer    :  Guiness
*** Room Beverages ***
Whisky  :  Black Label
Beer    :  Munich Hofbraeuhaus
```

DBCLOSE() Closes the key/value DB.

Closes the Key/Value Database.

```
say "CLOSE "DBCLOSE()   ->  CLOSE 0
```

DBSET([qualifier.]key, value) Writes a key/value record to the DB

The record indicated by the qualifier and key arguments will be saved into the Key/Value Database. If the record is already present, it will be overwritten.

A BREXX DCL data structure can be used to compose the record's various components. Go to the BREXX User's Guide for further information. DCL('structure-name,'$DEFINE').

The qualifier parameter defaults to **"any".**

Return code : 0 Put successfully
8 Put failed

dbKey contains the key
dbFKey contains the full key (including the qualifier, key)
dbResult contains the full record (including the full key)
dbRC return code of the function

For example, adding continents to the database:

```
/* Add Continents  */
   SAY DBSET('Cont.Europe','Continent Europe')
   SAY DBSET('Cont.Asia','Continent Asia')
```

```
   SAY DBSET('Cont.Africa','Continent Africa')
   SAY DBSET('Cont.North America','Continent North America')
   SAY DBSET('Cont.South America','Continent South America')
   SAY DBSET('Cont.Australia','Australia and Oceania')
```

DBGET([qualifier].key) Reads a key/value record from the DB

A BREXX DCL data structure can be used to compose the record's various components. Go to the BREXX User's Guide for further information. DCL('structure-name,'$DEFINE').

The qualifier parameter defaults to **"any".**

Return code : 0 Get successfully
8 Get failed

dbKey contains the key
dbFKey contains the full key (including the qualifier, key)
dbResult contains the record
dbRC return code of the function

for example, reading the Continents

```
/* Get Continents  */
   call DBGET('Cont.Europe')
   say dbresult
   call DBGET('Cont.Asia')
   say dbresult
   call DBGET('Cont.Africa')
   say dbresult
   call DBGET('Cont.North America')
   say dbresult
   call DBGET('Cont.South America')
   say dbresult
   call DBGET('Cont.Australia')
   say dbresult


Continent Europe
Continent Asia
Continent Africa
Continent North America
Continent South America
Australia and Oceania
```

DBDEL([qualifier.]key) Deletes a key/value record from the DB

Deletes a record specified by the qualifier and key parameters.
The qualifier parameter defaults to **"any".**

Return code : 0 Delete successfully
8 Delete failed or key not available

dbKey contains the key
dbFKey contains the full key (including the qualifier, key)
dbResult contains the deleted record (including the full key)
dbRC return code of the function

```
/* Remove Europe and Asia, add Eurasia instead */
   say "Delete "DBDEL('Cont.Europe')
   say dbresult
   say "Delete "DBDEL('Cont.Asia')
   say dbresult
   call DBSET('Cont.Eurasia','Continent of Eurasia')
   say dbresult
   call DBGET('Cont.Eurasia','Continent of Eurasia')
   say dbresult


Delete 0
CONT____Europe_____________________________Continent Europe
Delete 0
CONT____Asia_______________________________Continent Asia
CONT____Eurasia____________________________Continent of Eurasia
Continent of Eurasia
```

DBREMOVE([QUALIFIER/ALL/ONLY/ANY/CONTAINS],string) removes records of a project.

REMOVE records of the actual room.
Keywords:
ALL removes all members of the active room
QUALIFIER removes all records of the qualifier defined in the string parameter
ONLY removes all records whose key starts with the given string. The key is the plain key without the qualifier
ANY removes all records whose key contains the string parameter. The key is the plain key without the qualifier
CONTAINS removes all records whose record content contains string parameter

There is no function to remove all records of the VSAM dataset.

```
call import KeyValue
call dbmsglv 'N'
say "OPEN  "DBOPEN()              /* Open Key/Value Database */
say "ROOM  "DBROOM('WORLD')       /* switch to WORLD         */
call dbremove('QUA',"Continent") /* Remove records w. qualifier Continent
*/
call dbremove('ANY',"Mu")         /* Remove recs containing Mu in the key */
call dbremove('ONLY',"Wa")        /* Remove recs with starting key Wa */
call dbremove('ALL')              /* Remove all recs of the active room */
say "CLOSE "DBCLOSE()
```

Result:

```
OPEN  4
ROOM  0
Remove all records of room WORLD(AB) with Qualifier continent
--------------------------------------------------------------------------
continent.Africa removed
continent.Asia removed
continent.Australia removed
continent.Europe removed
continent.North_America removed
continent.South_America removed
Number of records removed 6

Remove records of room WORLD(AB) containing Mu in key
--------------------------------------------------------------------------
city.Multan removed
city.Mumbai removed
city.Munich removed
city.Murcia removed
city.Muscat removed
Number of records removed 5

Remove records of room WORLD(AB) with starting key Wa
--------------------------------------------------------------------------
city.Warsaw removed
city.Washington removed
city.Washington_DC removed
Number of records removed 3

Remove all records of room WORLD(AB)
--------------------------------------------------------------------------
city.`Ajman removed
city.Aarhus removed
city.Aba removed
city.Abidjan removed
city.Abomey-Calavi removed
city.Abu_Dhabi removed
city.Abuja removed
city.Accra removed
city.Ad_Dammam removed
city.Adana removed
city.Addis_Ababa removed
…
…
```

```
country.Zambia removed
country.Zimbabwe removed
union.BRICS removed
union.Common_Wealth removed
union.European_Union removed
union.North_American_Free_Trade removed
union.Southeast_Asian_Nations removed
Number of records removed 1036

CLOSE 0
```

DBLOCATE([[qualifier.]key/key-prefix]) Positions to a key/value record

Positions the VSAM to the first occurrence of the qualifier and key. If the key parameter does not exist or is the prefix of a key, LOCATE positions it to the key which comes next.

The records can be read sequentially by **DBNEXT**.
The qualifier parameter defaults to **"any".**

Return code : 0 Locate successfully
8 Locate failed

dbKey contains the key
dbFKey contains the full key (including the qualifier, key)
dbRC return code of the function

DBNEXT([ALL]) Reads the next key/value record

Following a DBLOCATE, the first, or next, record will be read. It will be returned as the next record if it matches the key given in DBLOCATE; otherwise, ALL must be supplied as a parameter. Any records that come after will be returned. The ending must be controlled by your REXX script. After the last record, the EOF condition will be reported

The qualifier parameter defaults to **"any".**

Return code : 0 Next successfully
4 Last entry of the qualifier reached, or EOF reached if **ALL** is specified
8 Next failed

dbKey contains the key
dbFKey contains the full key (including the qualifier, key)
dbResult contains the record

dbRC return code of the function

```
/* Locate North America and read all continents following */
   say DBLOCATE('Cont.North America')
   do forever
      if DBNEXT()>0 then leave
      say DBRESULT
   end
```

```
Continent North America
Continent South America
```

DBLINK([qualifier.]key,[qualifier.]key, link-type) Link two key/value records

DBLINK produces a record in the Reference Dataset that contains both keys and the link-type as the reference key. Both records must exist. This allows the navigation between the source records.

The qualifier parameters default to **"any".**

Return code : 0 Both records successfully linked
8 link failed, check the existence of source and target record
dbRC return code of the function
dbLHS full-key of the source-record
dbRHS full-key of the target-record

DBDELREF([qualifier.]key,[qualifier.]key, link-type) de-link two key/value records

DBDELREF removes the link between two source records. It is the counterpart to DBLINK. The qualifier parameters default to **"any".**

Return code : 0 link successfully removed
8 link remove failed
dbRC return code of the function
dbLHS full-key of the source-record
dbRHS full-key of the target-record

DBDELREFALL([qualifier.]key de-link all references of a source record

All links that were started from a source record are removed by DBDELREFALL. Links initiated from another source record to this one, are not removed. The default value for the qualifier is "**any**".

Return code : 0 links successfully removed
8 link remove failed

DBPRINT([qualifier.]key) prints VSAM KEY Value Record including created References

Prints VSAM KEY Value Record including created References. If an information model is defined it also splits the record in its attributes.

Example:

```
DBPRINT country.U.S.A
```

```
country.U.S.A
// ----------------------------------------
// Source available
//            Attributes
// ----------------------------------------
 *ACRONYM           USA
 *CAPITAL           Washington DC
 *DESCRIPTION       ?_
 *VISITED           ?_
// ----------------------------------------
//            Links to other Records
// ----------------------------------------
 >CAPITAL-IS        city.WASHINGTON DC
 >CONTAINED-IN      continent.NORTH AMERICA
 >MEMBER-OF         union.NORTH AMERICAN FREE TRADE
```

DBOUTARRAY(array-number) stores all created output of a command in an array

```
s4=screate(250)                  /* create an array */
call setg('dbOutArray',s4)       /* assign it to dboutarray */
call dbprint(country,'DETAILS')  /* run command, output stored in array */
call setg('dbOutArray','')       /* reset assignment, next commands will be
                                    normally displayed */
```

DBSAY(output-line) prints the output line

Prints the line (like SAY does), but DBOUTARRAY can also redirect it to an array. This lets you mix your output with any Key/Value command.

DBRCOUNT([qualifier.]key,'REFERENCES/USAGES') counts references of a key

Counts the references or usages of a specific Key/Value record.

Report and Maintenance Functions

DBLIST([[QUALIFIER/ONLY/ANY/CONTAINS],string]) lists records

List all records using the keyword (1. Parameter) and string combination.
If DBLIST is run without any parameters, it returns all entries for the active room.

Keyword:

QUALIFIER	lists all records of the qualifier defined in the string parameter
ONLY	lists all records whose key starts with the string parameter. The key is the plain key without the qualifier
ANY	lists all records whose key contains the string parameter. The key is the plain key without the qualifier
CONTAINS	lists all records whose record content contains string parameter

The DBLIST Format:

```
List records of room WORLD(AB) with starting key Wa
------------------------------------------------------------------------
 city.Wad Medani                                 Source 265124
 city.Warsaw                                     Source 1428379
 city.Washington                                 Source 3693775
 city.Washington DC                              Dummy  DUMMY
List contains 5 records
```

The first column contains the qualifier and the key
The second column contains either Source or Dummy.
- Source means a source record was inserted via DBSET (or equivalent command)
- Dummy means no source record was inserted yet, but the name has been reserved by a DBREFERENCE command

The third column contains the source record

Some examples

```
CALL DBLIST()

Result:
List all records of room WORLD(AB)
------------------------------------------------------------------
 any.AUS                                         Dummy  DUMMY
 any.Canberra                                    Dummy  DUMMY
 any.Mainland of the Australian continent        Dummy  DUMMY
 any.USA                                         Dummy  DUMMY
 any.Washington DC                               Dummy  DUMMY
 any.YES                                         Dummy  DUMMY
 city.`Ajman                                     Source 376263
 city.Aarhus                                     Source 219041
 city.Aba                                        Source 1174779
```

```
 city.Abidjan                                    Source 3823793
 city.Abomey-Calavi                              Source 503669
 city.Abu Dhabi                                  Source 1138691
 city.Abuja                                      Source 2894719
 city.Accra                                      Source 1833578
 city.Ad Dammam                                  Source 693590
 city.Adana                                      Source 1355973
 city.Addis Ababa                                Source 2334972
 city.Adelaide                                   Source 994888
 city.Aden                                       Source 389562
 city.Aguadilla                                  Source 199889
…
…
```

```
call dblist('QUALIFIER',"Continent")

Result:
List all records of room WORLD(AB) with Qualifier continent
-------------------------------------------------------------------
 continent.Africa                                Dummy  DUMMY
 continent.Asia                                  Dummy  DUMMY
 continent.Australia                             Dummy  DUMMY
 continent.Europe                                Dummy  DUMMY
 continent.North America                         Dummy  DUMMY
 continent.South America                         Dummy  DUMMY
List contains 6 records
```

```
call dblist('ONLY',"Wa")

Result:
List records of room WORLD(AB) with starting key Wa
-----------------------------------------------------------------------
 city.Wad Medani                                 Source 265124
 city.Warsaw                                     Source 1428379
 city.Washington                                 Source 3693775
 city.Washington DC                              Dummy  DUMMY
List contains 5 records
```

```
call dblist('ANY',"Mu")

Result:
List records of room WORLD(AB) containing Mu in key
-------------------------------------------------------------------
 city.Multan                                     Source 1437257
 city.Mumbai                                     Source 19175018
 city.Munich                                     Source 2000981
 city.Murcia                                     Source 516575
 city.Muscat                                     Source 1091400
List contains 5 records
```

```
call dblist('CONTAIN',"265")

Result:
List records of room WORLD(AB) containing 265
------------------------------------------------------------------------
 city.Ahvaz                                    Source 968265
 city.Bamako                                   Source 1542654
 city.Busan                                    Source 2651469
 city.Oskemen                                  Source 265766
 city.Peshawar                                 Source 1512657
 city.Tanch'on                                 Source 265573
 city.Wad Medani                               Source 265124
List contains 7 records
```

DBKEEP([[QUALIFIER/ONLY/ANY/CONTAINS],string]) lists records

Similar to DBLIST in functionality, except it saves the outcome in a SARRAY that the FMTLIST function can utilize to display or post-process.
DBKEEP returns the created SARRAY number

Example

```
s2=dbkeep('Qualifier','Country')
buffer.0='ARRAY 's1
Call fmtlist
```

DBHOOD([qualifier.]key) prints the neighbourhood (References) of a Record

Example

```
DBHOOD country.U.S.A
```

Result

```
  PART-OF           city.ATLANTA
  PART-OF           city.ATLANTA:U.S.A
  PART-OF           city.BOSTON
  PART-OF           city.BOSTON:U.S.A
  PART-OF           city.CHICAGO
  PART-OF           city.CHICAGO:U.S.A
  PART-OF           city.DALLAS
  PART-OF           city.DALLAS:U.S.A
  PART-OF           city.HOUSTON
  PART-OF           city.HOUSTON:U.S.A
  PART-OF           city.LOS ANGELES
  PART-OF           city.LOS ANGELES:U.S.A
  PART-OF           city.MIAMI
  PART-OF           city.MIAMI:U.S.A
  PART-OF           city.NEW YORK
  PART-OF           city.NEW YORK:U.S.A
```

```
  PART-OF           city.PHILADELPHIA
  PART-OF           city.PHILADELPHIA:U.S.A
  PART-OF           city.WASHINGTON
  PART-OF           city.WASHINGTON:U.S.A
   |  Refer(s) to country.U.S.A
   V
+-------------------------------------------------+
| country.U.S.A                                   |
+-------------------------------------------------+
   |  Reference(s) from country.U.S.A
   V
  CAPITAL-IS        city.WASHINGTON DC
  CONTAINED-IN      continent.NORTH AMERICA
  MEMBER-OF         union.NORTH AMERICAN FREE TRADE
```

DBREFERENCE([qualifier.]key,[max-level],[REFS/DETAILS)
lists referred records

Navigates from a given **qualifier.key** combination to all referred records (forward direction).
Max-level defines how many nested levels are allowed (the default is 99).
With **REFS** only the referred entries will be reported.
With **DETAILS** referred entries and the link type are reported.

The qualifier parameter defaults to "any".

```
Call dbreference('country.USA')

References of Country.USA
-----------------------------------------
Country.USA -- part of -->  North America
```

Which references constitute city.Munich, 99 levels down, report referred entries only:

```
Call dbreference('country.USA',99,'REFS')

city.Munich
  PART-OF           country.GERMANY
  CAPITAL-IS        city.BERLIN
  PART-OF           country.GERMANY
  CONTAINED-IN      continent.EUROPE
  MEMBER-OF         union.EUROPEAN UNION
  SIGHT             event.OCTOBERFEST
  BREWERY           any.HACKERBRAEU
  BREWERY           any.HOFBRAEU
  TYPE              any.DARK
  TYPE              any.EXPORT
  BREWERY           any.LOEWENBRAEU
  BREWERY           any.PAULANER
  SIGHT             location.HOFBRAEUHAUS
  BREWERY           any.HOFBRAEU
Elements found 14
```

Which references constitute city.Munich, 99 levels down, report referred and link type entries only:

```
Call dbreference('country.USA',99,'LINK')

city.Munich
  PART-OF            country.GERMANY
  CAPITAL-IS         city.BERLIN
  PART-OF            country.GERMANY
  CONTAINED-IN       continent.EUROPE
  MEMBER-OF          union.EUROPEAN UNION
  SIGHT              event.OCTOBERFEST
  BREWERY            any.HACKERBRAEU
  BREWERY            any.HOFBRAEU
  TYPE               any.DARK
  TYPE               any.EXPORT
  BREWERY            any.LOEWENBRAEU
  BREWERY            any.PAULANER
  SIGHT              location.HOFBRAEUHAUS
  BREWERY            any.HOFBRAEU
Elements found 14
```

Which references constitute city.Munich, 99 levels down (maximum):

```
Call dbreference('city.Munich',99)


References of city.Munich
-------------------------------------------------------------------------
 1 city.Munich
 1 +- part-of        ->  country.GERMANY
 2 |  country.GERMANY
 2 |  +- capital-is     ->  city.BERLIN
 3 |  |  city.BERLIN
 3 |  |  +- part-of        ->  country.GERMANY
 4 |  |  |  |- capital-is     -># city.BERLIN
 4 |  |  |  |- contained-in   ->  continent.EUROPE
 4 |  |  |  country.GERMANY
 4 |  |  |  +- member-of      ->  union.EUROPEAN UNION
 2 |  |- contained-in   -># continent.EUROPE
 2 |  |- member-of      -># union.EUROPEAN UNION
 1 |- sight          ->  event.OCTOBERFEST
 2 |  event.OCTOBERFEST
 2 |  +- brewery        ->  any.HACKERBRAEU
 2 |  event.OCTOBERFEST
 2 |  +- brewery        ->  any.HOFBRAEU
 3 |  |  any.HOFBRAEU
2 |  +- brewery        ->  any.LOEWENBRAEU
 2 |  event.OCTOBERFEST
 2 |  +- brewery        ->  any.PAULANER
 1 city.Munich
 1 +- sight          ->  location.HOFBRAEUHAUS
 2 |  location.HOFBRAEUHAUS
 2 |  +- brewery        ->  any.HOFBRAEU
 3 |  |  |- type           -># any.DARK
 3 |  |  |- type           -># any.EXPORT
    -># references have been reported previously
Elements found 14
```

DBUSAGE([qualifier.]key,[REFS/DETAILS) lists used-by records

Navigates from a given **qualifier.key** combination to all used records (backward direction).
Max-level defines how many nested levels are allowed (the default is 99).
With **REFS** only the used entries will be reported.
With **DETAILS** referred entries and the link type are reported.

```
Call dbusage('country.USA')

Usages of Country.USA
-------------------------------------------------
Country.USA -- Economy <--  NAFTA
```

DBROOMS() lists all rooms

DBROOM displays all previously defined rooms.

```
List all defined rooms
  HILBERT'S LOBBY         SAA 0
  MOSHE'S FOOD TRUCK      SAF 0
  PETER'S BEACH BAR       SAD 0
  WORLD                   SAB 0
```

Key/Value Database Tailoring

Defining an Information Model (optional)

You may specify a record structure using an information model, which allows the record to be divided into distinct attributes. Every qualification has to have its model defined if one has been set up. The information model is active in the current room. Here is an illustration of the structure of the world database sample:

Sample information model in the world database:

The qualifier **country** has the following attributes:

```
call dbkvimadd 'country: Acronym Capital Description Visited'
```

The qualifier **city** has the following attributes:

```
call dbkvimadd 'city: population Description'
```

if all qualifiers (regard them as types) the information model must be built, which can be done by:

```
call dbkvimbuild
```

The information model should be defined at the initialisation process before the database is loaded. The following is the definition of the world database::

```
/* ------------------------------------------------------------
 * Store a simple Key/Value Information Model
 * Example:
 *    country: Acronym Capital Description Visited
 *      |      |       |       |           fourth-attribute
 *      |      |       |       third-attribute
 *      |      |       second-attribute
 *      |      first-attribute
 *      record-type
 * ------------------------------------------------------------
 */
call dbkvimadd 'country: Acronym Capital Description Visited'
call dbkvimadd 'city: population Description'
/* ------------------------------------------------------------
 * Build Information Model and save it in the Control Record
 * ------------------------------------------------------------
 */
call dbkvimbuild
```

Once activated the record can be automatically structured by some commands into its single attributes for example DBPRINT.

BREXX KEY/VALUE Database

Defining additional Key/Value Databases

You can add additional Key/Value Databases as needed. Their definition lengths may vary (e.g. qualifier and key)

Setting up a Key/Value Profile

To personalise the profile REXX script, alter the following REXX variables; any valid dataset or dd name may be selected: define a profile, for example, **DBPROF1**

```
/* ------------------------------------------------------------
 * Private Key/Value Profile
 * ------------------------------------------------------------
 */
  ddprof.DDKEY='PRIVALUE'          /* DD Name of Key/Value DB*/
  ddprof.DDREF='PRIREFS'           /* DD Name of Key/Ref DB  */
  ddprof.DSNKEY='BREXX.PRIVALUE'   /* DSN of Key/Value DB    */
  ddprof.DSNREF='BREXX.PROREFS'    /* DSN of Key/REF DB      */
  ddprof.keylen =12                 /* Plain Key length              */
  ddprof.roomlen=2                  /* Room length                   */
  ddprof.quallen=4                  /* Qualifier/Project/Bucket      */
  ddprof.typelen=4                  /* Type length in Reference DB   */

  ddprof.keyupper=1                 /* upper case translation of the */
                                    /* key. 0: no, 1: yes            */
  ddprof.EnableDummy=1              /* allow links between records   */
                                    /* which do not exist, they will */
                                    /* create DUMMY source records   */
                                    /* 0: no, 1: yes                 */
 return 0
```

Calculate the Key Length for the Cluster Definition

Keylen: Key/Value key length=ddprof.keylen+ddprof.projlen+ddprof.roomlen
In our example **Keylen=18**

Reflen: Reference key length=2***keylen**+1+ddprof.typelen
In our example **Reflen=41**

Cluster Definition

1. Create a copy of the member **$CREKEYV** of the BREXX installation dataset. Change the cluster definition accordingly.
2. Change the **Cluster sizes** so that it fits your need.
3. Submit the JCL

For our Example:

```
//* ------------------------------------------------------------
//* CREATE A NULL RECORD FOR THE KEY/VALUE VSAM FILES
//* ------------------------------------------------------------
//EXEC     EXEC PGM=BREXX,PARM='RXRUN',REGION=6000K
```

```
//STEPLIB  DD   DSN=SYS2.LINKLIB,DISP=SHR
//STDIN    DD   DUMMY
//STDOUT   DD   SYSOUT=*,DCB=(RECFM=FB,LRECL=140,BLKSIZE=5600)
//STDERR   DD   SYSOUT=*,DCB=(RECFM=FB,LRECL=140,BLKSIZE=5600)
//NULLREC  DD DSN=&&NULLREC,DISP=(,PASS),UNIT=VIO,SPACE=(TRK,(1,1)),
//         DCB=(RECFM=FB,LRECL=255,BLKSIZE=255)
//RXRUN    DD   *
   NULLR.1=COPIES('9',255); NULLR.0=1
  "EXECIO * DISKW NULLREC (STEM NULLR. FINIS"
/*
//* ---------------------------------------------------------------
//* DELETE/DEFINE THE PROMOTE VSAM CHANGE (CCID) LIST AND PRIME IT
//* WITH THE CONTROL RECORD
//* ---------------------------------------------------------------
//KEYVALUE EXEC PGM=IDCAMS
//NULLREC  DD DSN=&&NULLREC,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DELETE BREXX.PRIVALUE CLUSTER
  SET MAXCC = 0
  DEFINE CLUSTER                          -
           (NAME(BREXX.PRIVALUE)          -
            INDEXED                       -
            KEYS(18 0)                    -
            RECORDSIZE(64 8192)           -
            SHAREOPTIONS(2,3)             -
            CYLINDERS(600 50)             -
            VOLUMES(PEJ001)               -
            UNIQUE                        -
            SPEED)                        -
         DATA                             -
            (NAME(BREXX.PRIVALUE.DATA))   -
         INDEX                            -
            (NAME(BREXX.PRIVALUE.INDEX))
  IF LASTCC = 0 THEN -
     REPRO INFILE(NULLREC) ODS(BREXX.PRIVALUE)
/*                                                /*
//KEYREF   EXEC PGM=IDCAMS
//NULLREC  DD DSN=&&NULLREC,DISP=SHR
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   *
  DELETE BREXX.PRIREFS CLUSTER
  SET MAXCC = 0
  DEFINE CLUSTER                          -
           (NAME(BREXX.PRIREFS)           -
            INDEXED                       -
            KEYS(41 0)                    -
            RECORDSIZE(64 256)            -
            SHAREOPTIONS(2,3)             -
            CYLINDERS(50 25)              -
            VOLUMES(PEJ001)               -
            UNIQUE                        -
            SPEED)                        -
         DATA                             -
            (NAME(BREXX.PRIREFS.DATA))    -
         INDEX                            -
```

```
             (NAME(BREXX.PRIREFS.INDEX))
  IF LASTCC = 0 THEN -
     REPRO INFILE(NULLREC) ODS(BREXX.PRIREFS)
/*
//
```

Usage of the new Cluster Definition

Once the definition is constructed, you may use it by specifying the profile name in the DBOPEN call, for example, DBPROF1:

```
CALL DBOPEN('DBPROF1')
```

There can only be one database open at the same time! If another database was opened during the same run, it must be closed using a DBCLOSE.

Table of Contents