BREXX/370 V2R3M0 Software and Migration Notices

Document Version 1.0 created on 25. November 2020
Authors: Peter Jacob (pej), Mike Großmann (mig)

Content

This document covers the amendments and changes in the new version of BREXX/370 V2R3M0. For changed functionality, it contains the migration instruction to upgrade from the previous release(s). The installation process is separately described in the **BREXX/370 Installation** document. New features or functions are described in details in:

BREXX370_Users_Guide_V2R3M0.pdf or a separately mentioned document.

All documents are contained in the installation zip file

Upgrade from a previous BREXX/370 Version

Before upgrading to BREXX/370 V2R3M0 backup your system. The easiest way is creating a copy of your TK4- directory containing all of your settings and the DASD volumes. In the case of errors or unwanted behaviour, you can quickly recover to the backup version.

I. Important Changes

If you migrate from an earlier version of BREXX/370 then additionally take a look into the Migration Notices of V2R1M0 and V2R2M0, which are contained in the installation zip file

1. Software Changes requiring actions

1.1. Authorised BREXX Version available

With this release, we ship a standard installation of BREXX as well as an authorised version, which allows to system programs as IEBCOPY, NJE38, etc. The decision about what to install must be made before installation.

1.2. BREXXSTD load module removed

We have straightened the load module structure and removed the BREXXSTD load module from the installation library. If you use JCL with an explicit BREXXSTD call, replace it by BREXX. During the installation process, any existing BREXXSTD module will be removed from SYS2.LINKLIB.

1.3. Call PLI Functions

Example compile jobs for callable PLI Functions can be found in **BREXX. V2R3M0.JCL:**

RXPI calculate PI with 500 digits

RXCUT Return every n.th character of a string

2. Known Problems

2.1. Reading Lines from a sequential dataset

Reading lines of sequential datasets always truncate trailing spaces; this may be an unwanted behaviour for fixed-length datasets. To circumvent this problem, you can use the following method:

If the dataset is allocated via a DD statement:

```
X=LISTDSI('INFILE FILE')
fhandle=OPEN(infile, 'RB')
Record=READ(fhandle,SYSLRECL)
```

If the dataset is used directly:

```
dsn='HERC01.TEMP'
X=LISTDSI("'" dsn"'")
fhandle=OPEN("'" dsn"'",'RB')
Record=READ(fhandle,SYSLRECL)
```

LISTDSI returns the necessary DCB information (SYSLRECL). The OPEN must be performed with OPTION 'RB' which means READ, BINARY. Read uses the record length to create the record.

2.2. BREXX FORMAT Function

The BREXX FORMAT function differs from the standard behaviour of REXX FORMAT:

*FORMAT rounds and formats number with **before** integer digits and **after** decimal places. **expp** accepts the values 1 or 2 (WARNING different from the ANSI-REXX specification) where 1 means to use the "G" (General) format of C, and 2 the "E" exponential format of C. Where before+after+1 replaces the place of the total width specifier in C. (**expt** is ignored!)*

After determining the code, we discovered that a complete re-write would be necessary. As the effort does not stand in proportion to the benefit, we decided to leave it as it is.

II. New and amended functionality

1. BREXX functions and Commands

CEIL(decimal-number)
CEIL returns the smallest integer greater or equal than the decimal number.

BREXX/370 V2R3M0 Software and Migration Notices

D2P(number,length,fraction-digit)

Converts a number (integer or float) into a decimal packed field. The created field is in binary format

P2D(number,length,fraction-digit)

Converts a decimal packed field into a number.

ENCRYPT(string,password <,rounds>) and
DECRYPT(string,password <,rounds>)

Encrypts/Decrypts a string

DUMPIT(address,dump-length)

DUMPIT displays the content at a given address of a specified length in hex format. The address must be provided in hex format; therefore, a conversion with the D2X function is required.

DUMPVAR('variable-name')

DUMPVAR displays the content of a variable or stem in Hex format-

FILTER(string,character-table <,drop/keep>)

The filter function removes all characters defined in the character-table

FLOOR(decimal-number)

FLOOR returns the smallest integer less or equal than the decimal number.

LISTIT('variable-prefix')

Returns the content of all variables and stem-variables starting with a specific prefix

RHASH(string,<slots>)

The function returns a numeric hash value of the provided string.

ROUND(decimal-number,fraction-digits)

The function rounds a decimal number to the precision defined by fraction-digits

UPPER(string)
LOWER(string)

UPPER returns the provided string in upper cases. LOWER in lower cases.

ROTATE(string,position<,length>]

The function returns a rotating substring

TIMESTAMP([day,month,year])

TIMESTAMP returns the unix (epoch) time, seconds since 1. January 1970.

2. Dataset Functions

CREATE(dataset-name,allocation-information)

The CREATE function creates and catalogues a new dataset

DIR(partitioned-dataset-name)

The DIR command returns the directory of a partitioned-dataset

EXISTS(dataset-name)
EXISTS(partitioned-dataset(member))

The EXISTS function checks the existence of a dataset or the presence of a member in a partitioned dataset.

REMOVE(dataset-name)

The REMOVE function un-catalogues and removes the specified dataset

REMOVE(partitioned-dataset(member))

The REMOVE function on members of a partitioned dataset removes the specified member.

RENAME(old-dataset-name,new-dataset-name)

The RENAME function renames the specified dataset.

RENAME(partitioned-dataset(old-member),partitioned-name(new-member))

The RENAME function on members renames the specified member into a new one.

ALLOCATE(ddname,dataset-name)
ALLOCATE(ddname,partitioned-dataset(member-name))

The ALLOCATE function links an existing dataset or a member of a partitioned dataset to a dd-name.

FREE(ddname)

The FREE function de-allocates an existing allocation of a dd-name.

OPEN(dataset-name,open-option,allocation-information)

The OPEN function has now a third parameter, which allows creating new datasets with appropriate DCB and system definitions.

3. TCP Functions

TCPINIT()
TCPINIT initialises the TCP functionality.

TCPSERVE(port-number)
Opens a TCP Server on the defined port-number for all its assigned IP-addresses.

TCPOPEN(host-ip,port-number[,time-out-secs])
TCPOPEN opens a client session to a server.

TCPWAIT([time-out-secs])
TCPWAIT is a Server function; it waits for incoming requests from a client.

TCPSEND(clientToken,message[,timeout-secs])
SendLength=TCPSEND(clientToken, message[,time-out-secs]) sends a message to a client.

TCPReceive(clientToken,[time-out-secs])
Receives a message from another client or server.

TCPTERM()
Closes all client sockets and removes the TCP functionality

3. New BREXX functions coded in REXX

GETTOKEN

returns a token which is unique within a running MVS System or in this century

BAS64ENC
Encodes a string or binary string with Base64.

BAS64DEC
Decodes a base64 encoded string into a string or binary string
Returns the hash number of a string

STIME
Time since midnight in hundreds of a second

4. Support of Formatted Screens

There is a separate BREXX Formatted Screen document:

BREXX370_Formatted_Screens_V2R3M0.pdf

for details on functions and usage, please take a look into it.

- New **FMTMENU** allows defining a full menu
- **FMTLIST** Buffers can be stacked
- Extended tailoring on FMTMENU, FMTLIST and FMTCOLUM

5. Integration of VSAM I/O

There is a separate BREXX VSAMIO document:

BREXX370_VSAM_V2R3M0.pdf

for details on functions and usage, please take a look into it.