REXXTOOLS/MVS™

General Information Manual

Version 7 Release 4 Modification Level 5

Open Software Technologies, Inc

REXXTOOLS/MVS Version 7 Release 4 Modification Level 5

Open
Software
Technologies, Inc.

P.O. Box 162652
Altamonte Springs, FL
32716-2652

Phone: (407) 788-7173
Fax: (407) 788-8494
E-mail: support@open-softech.com
Web: www.open-softech.com

Table of Contents

Introduction **1**
Who this Book is for 1
Related Publications 1
Other REXX Resources 2
Trademarks 2

Frequently Asked Questions **3**
What is REXXTOOLS/MVS? 3
Why Should I Use REXXTOOLS/MVS? 4
Why Should I Use REXX? 5
What Can I Do With REXXTOOLS/MVS? 7
In What Environments is REXXTOOLS/MVS Used? 8

Facilities **9**
Basic Services 9
DB2/SQL Services 17

Installation **21**
Media 21
Prerequisite Software 21
Authorization Considerations 21
Installation Overview 22

Support **25**

Case Studies **27**
Inter-bank Funds Transfer Application 27
Source Library Management and Control 27
Insurance Claim Error Correction 27
NetView-based Help Desk 28
CA-Endevor "Bridge" Application 28
EDI Transaction Processing 28
RACF-to-DB2 Help Desk Application 29
DB2 Application Error Checking 29
DB2 Application Testing 29

Sample Application **31**
Internet CGI Script Using DB2/SQL 31

What's New in REXXTOOLS/MVS **35**
What’s New in Version 7 Release 4 Mod 5 35
What's New in Version 7 Release 4 Mod 3 35
What's New in Version 7 Release 4 35
What's New in Version 7 Release 3 36
What's New in Version 7 Release 2 36
What’s New in Version 7 Release 1 36
What's New in Version 6 Release 1 36
What's New in Version 5 Release 1 38
What's New in Version 4 Release 1 Mod 2 40
What's New in Version 4 Release 1 42
What's New in Version 2 Release 1 Mod 2 44
What's New in Version 2 Release 1 45
What's New in Version 1 Release 2 46

Introduction

Who this Book is for

This manual provides an overview of REXXTOOLS/MVS. Its target audience consists of:

Programmers who need to evaluate the technical qualifications of REXXTOOLS/MVS. If this is your role, you should read the following sections:

- "Facilities" on page 9 and
- "Installation" on page 21.

Managers who need to evaluate the business case for purchasing REXXTOOLS/MVS. The following sections of this manual are designed to address management issues:

- "Frequently Asked Questions" on page 3 and
- "Case Studies" on page 27.

Related Publications

The REXXTOOLS/MVS publications are:

- *REXXTOOLS/MVS General Information*
- *REXXTOOLS/MVS Installation Guide*
- *REXXTOOLS/MVS Basic Services User's Guide*
- *REXXTOOLS/MVS DB2/SQL Services User's Guide*

REXXTOOLS/MVS documentation is available in a variety of formats, electronic and hardcopy. Consult your distributor or Open Software Technologies for more information.

If you are new to REXX, there are two IBM manuals with which you will want to become familiar. These are:

REXX/MVS User's Guide
: This manual provides an excellent tutorial on the REXX language as it is implemented under MVS. If you have never used REXX, we suggest you start by reading this book.

REXX/MVS Reference
: This manual is a comprehensive reference for the REXX language. This book is the definitive source for information on REXX language statements, built-in functions, and IBM-supplied host commands. It does not contain any tutorial information.

Other REXX Resources

Additional information on REXX can be obtained from the following resources:

The REXX Language
(Prentice-Hall, 1990, ISBN 0-13-780651-5), by Mike Cowlishaw, is the seminal REXX document. As the inventor of the REXX language, Dr. Cowlishaw provides a unique perspective on its purpose and design.

The REXX Handbook
(McGraw Hill, 1991, ISBN 0-07-023682-8), edited by Gabe Goldberg and Philip Smith, contains information from a variety of sources concerning the history of REXX, REXX on non-MVS platforms, and REXX programming techniques.

The REXX Language Association
The Uniform Resource Locator (URL) for this page is

http://www.rexxla.org

The REXX web page has information on the latest REXX developments at IBM, and contains many links to other REXX-related web sites.

Trademarks

Frequently Asked Questions

This section provides answers to some of the most commonly asked questions about REXXTOOLS/MVS and REXX.

What is REXXTOOLS/MVS?

REXXTOOLS/MVS (“REXXTOOLS”) is a collection of interfaces and facilities designed to expand the range of applications that may be implemented in REXX. From a REXX language perspective, REXXTOOLS is composed of:

REXX Functions

A function is special kind of subroutine which, in addition to performing some type of service, returns a value when executed. REXXTOOLS functions are used in the same manner as the built-in functions of REXX. That is, no special steps -- other than installing the product -- must be performed in order to use the functions.

Host Commands

A host command is a REXX expression the result of which is a command to be executed by an external program called a host command processor. Typically, a host command is simply a quoted character string, but complex expressions containing function calls and REXX operators are also supported. REXX resolves all expressions prior to invoking the host command processor. Host commands usually produce REXX variables and other side effects.

Utilities

Utilities are programs used in the development of REXX applications. Some of the utilities are invoked as TSO commands. Others are edit macros, and batch execution programs.

Example Applications

The example applications are REXX programs that demonstrate REXXTOOLS functions and host commands.

Many of the REXXTOOLS functions and host commands provide access to vital system services and data stores. Interfaces are included for accessing:

- Sequential, partitioned, SYSIN, SYSOUT, and VSAM data sets
- DB2 tables
- DFP, DFSMS, and IDCAMS information
- z/OS, including DYNALLOC
- TSO services
- CICS Distributed Program Link (DPL) transactions
- The JES spool

REXXTOOLS manages the tedious details of storage allocation and data conversion, freeing the programmer to concentrate on the requirements of the application. Where necessary, REXXTOOLS maintains a task-level context between function calls and host commands. By doing so, it is able to conserve and reuse the system resources it allocates on the application's behalf. For example, system service modules such as IDCAMS are loaded just once per MVS task. REXXTOOLS saves module addresses so that subsequent invocations proceed more quickly. For the DB2 interfaces, the amount of contextual information saved by REXXTOOLS is extensive, yielding a significant boost in application performance.

In addition to the system service interfaces, REXXTOOLS augments the data manipulation capabilities of the REXX language with functions, host commands, and utilities. In this category, REXXTOOLS has services for:

- specialized data conversion and formatting
- advanced string parsing
- high-performance in-memory sorting of strings and arrays
- sharing variables between external REXX programs
- pattern matching

REXXTOOLS may be used in REXX/MVS (interpreted REXX) or REXX/370 (compiled REXX) programs. Please see "Sample Application" for an example of REXXTOOLS programming.

Developmental Notes

1. REXXTOOLS/MVS is implemented by over 100000 S/390 assembler language statements (In comparison, the core of the REXX/MVS interpreter is less than 10000 assembler statements). An additional 10000 lines of REXX code is contained in utilities and the EXEC and SAMPLIB data sets.

2. As of Version 6, approximately 16 man-years have been expended to develop REXXTOOLS/MVS.

3. All modules (except for data-only CSECTs) are re-entrant. This permits a single in-memory image of REXXTOOLS to serve all users on a system.

4. With few exceptions, all modules use the 31-bit addressing mode and can reside above the 16M line.

Why Should I Use REXXTOOLS/MVS?

With numerous licenses worldwide, REXXTOOLS/MVS is used by many different types of organizations for many different purposes. Despite its diverse uses, one or more of the following factors appear in every REXXTOOLS purchase:

The right tool for the job.

Under Netview, Tivoli® AF/OPERATOR®, and many other system automation products, REXX is the primary scripting language. REXXTOOLS provides the right set of complementary facilities to make application development in these environments a reality.

Cost savings.

Some customers have justified purchasing REXXTOOLS based on cost savings. For example:

- The REXXTOOLS ALLOCATE command can reduce the CPU time required to process logon allocations.

- The DB2/SQL services of REXXTOOLS often perform better than SPUFI and QMF. In some cases, the reduction in CPU utilization pays for the product. For other organizations, the displacement of one or more expensive software products justifies the purchase of REXXTOOLS/MVS.

Application backlogs.

Because REXXTOOLS/MVS is powerful yet programmer-friendly, customers are able to accomplish more with fewer resources. As more corporations have "downsized" and "rightsized," this factor has appeared with increasing frequency.

Similarity to small systems programming.

The REXX/REXXTOOLS combination provides MVS programmers with the types of facilities that Visual Basic, Power Script, and Java Script provide for small systems programmers. A significant number of customers are using REXX and REXXTOOLS to provide a "culture-shock-free" environment for programmers who must work in both PC and mainframe settings.

Flattening the learning curve.

Today, application development organizations are hard-pressed to find qualified programmers. As a consequence, programmers are being recruited from previously untapped sources such as data center operations and business units. For these new programmers, REXX and REXXTOOLS can shorten the "ramp-up" time from months to weeks.

In sum, because of its versatility, power, and ease-of-use, the combination of REXX and REXXTOOLS is improving the operation of companies everywhere.

Note: Please see "Case Studies” on page 27 for examples of customer applications.

Why Should I Use REXX?

Simply put, REXX is the easiest to learn, yet most powerful programming language ever devised. Some of the features that elevate REXX to a level above COBOL, PL/I and C are:

Typeless data.

REXX treats all data -- including numbers -- as character strings. Most conversions are performed, automatically, as required.

No variable declarations.

A variable is created whenever it is referenced. An especially powerful feature of the language is that variable names can be constructed dynamically. Moreover, a variable can be assigned many different types of values (numeric and non-numeric; binary and printable) during the execution of a program.

Flexible data structures.

REXX variables can be used to construct arrays, structures, arrays of structures, linked lists, and trees. REXX "stem" arrays may have an unlimited number of dimensions and may use non-numeric subscripts.

Dynamic scoping.

Variables may be "exposed" to subroutines or hidden depending on the application's requirements. The scope of a variable depends on the order of execution, not on its lexical location.

Dynamic execution.

REXX programs can construct and execute REXX statements using the INTERPRET instruction.

Extensive string handling capability.

Language instructions and built-in functions are provided for composing and decomposing strings and records. REXX's powerful string handling makes it ideal for dynamically composing statements for programming and job control languages.

Extremely high-precision decimal operations.

All types of mathematical operations can be performed on arbitrarily large decimal numbers.

Unsurpassed environmental integration.

Whereas most other mainframe languages require external commands or job control to use MVS facilities, REXX is designed to directly interface with these services. When augmented by REXXTOOLS/MVS, no other language -- except assembler -- is better integrated into MVS.

Extensive and easily understood control constructs.

REXX supports many types of loops and decision control constructs. REXX encourages programmers to create well-structured programs through the use of functions, subroutines, and condition handlers.

Powerful interactive debugging facilities.

The execution of a REXX program can be traced and halted at any point. When a breakpoint is reached, the contents of variables can be inspected and modified. No special debugging environment is required to make use of these facilities. They are incorporated into the REXX/MVS interpreter.

Interpreted for rapid application development.

REXX encourages experimentation and incremental development. Most programs can be run from within the ISPF editor.

Compiler for speed-sensitive applications.

Although the REXX/MVS interpreter is fast and efficient, computationally intensive applications may require an additional performance boost. The IBM REXX/370 compiler satisfies this requirement. Compiled REXX programs exhibit performance characteristics comparable to other compiled languages.

Although the analogy is less than perfect, REXX can be thought of as a kind of Visual Basic for mainframe programmers. That is, it is a language that simultaneously extends programming to the non-programmer and permits the development of sophisticated applications.

What Can I Do With REXXTOOLS/MVS?

REXXTOOLS/MVS can be used to perform many common programming tasks quickly and easily. You can, for example, use REXXTOOLS to:

Create Internet Applications.

REXXTOOLS/MVS services can be used in REXX-based CGI scripts for popular MVS-based web servers, including IBM's HTTP Server. In addition, REXXTOOLS can be used with the RXSOCKET function of IBM TCP/IP to create applications using other Internet protocols.

Create Client/Server Applications.

REXXTOOLS can be used in client/server environments such as APPC/MVS, IBM TCP/IP, and DB2's Distributed and Remote Units of Work.

Create ISPF Applications.

REXX and REXXTOOLS are fully integrated into ISPF. REXXTOOLS lets you build applications to display and share VSAM and DB2-hosted data in hours instead of weeks.

Create System Automation Applications.

REXXTOOLS is used in all of the major MVS automation packages. In these environments, customers are using REXXTOOLS to develop automated trouble ticketing, exception logging, and data warehousing applications.

Create Bridge Applications.

REXXTOOLS often is used to bridge gaps between vendor products and in-house applications. Because it provides access to a wide range of data stores, REXXTOOLS has been used with vendor products as diverse as CA-Endevor and Soft Switch.

Write Batch Reports.

REXX and REXXTOOLS can be used in batch jobs and started tasks, to produce ad hoc and production reporting job streams. The REXXTOOLS currency formatting routine, D2PIC, provides the REXX programmer with the same support for number formatting as is available in COBOL.

Post Process Reports.

Often the source to older report writing programs has been lost or is too difficult to modify. Using REXX and REXXTOOLS, you can quickly construct programs to read report files and make the necessary modifications.

Create Custom Utilities.

Many customers have found that REXXTOOLS can be used to augment and in some cases replace expensive IBM and ISV- supplied utilities. In particular, our customers tell us that report writing using the REXXTOOLS Dynamic DB2/SQL Service is, in most cases, easier and performs better than equivalent IBM utilities.

Improve System Performance.

The highly efficient REXXTOOLS ALLOCATE command can be used to replace TSO ALLOCATEs in your logon execs. Customers who have implemented this change report cost savings that justify the purchase of REXXTOOLS in a matter of months.

In What Environments is REXXTOOLS/MVS Used?

REXXTOOLS/MVS is used on systems running z/OS. The product has been verified to operate in the following REXX/MVS environments:

- z/OS
- Interactive TSO
- Batch TSO
- Batch REXX
- REXX started tasks
- HTTP Server (IBM)
- Web390 (Information Builders)
- APPC/MVS
- NetView
- Tivoli® AF/OPERATOR®
- AutoOPERATOR
- Unicenter CA-OPS/MVS

In general, REXXTOOLS/MVS may be used wherever REXX/MVS or REXX/370 is used.

Facilities

REXXTOOLS/MVS contains service components which may be licensed, installed, and used together or independently. The components are:

- Basic Services
- Dynamic DB2/SQL Services
- Static DB2/SQL Services

The following sections describe the functions, host commands, and other facilities of the REXXTOOLS/MVS components.

Basic Services

REXXTOOLS/MVS Basic Services provides functions, host commands, and utilities for:

- Utilizing the QSAM, BPAM, and VSAM access methods
- Issuing IDCAMS commands and retrieving IDCAMS messages
- Retrieving information from DFP and DFSMS
- Using z/OS system services
- Building full-screen and line-mode applications outside of ISPF
- Enhancing REXX data conversion, string manipulation, array handling, and variables sharing
- Dynamically allocating and freeing data sets outside of TSO
- Encapsulating REXX programs in OS load modules
- Invoking DPL CICS transactions.
- Accessing the JES spool.

QSAM

The REXXTOOLS/MVS QSAM interface gives REXX programmers access to sequential files, SYSIN and SYSOUT data sets, and partitioned data set members using the Queued Sequential Access Method (QSAM). The interface, which is implemented in REXX functions, provides several capabilities not provided by the REXX/MVS EXECIO command:

- You can read, write and update all record formats including spanned. The EXECIO command specifically does not support "U" and spanned format data sets.
- You can read PDS/PDSE directories. The EXECIO command will not read these directories.
- Because it is implemented with functions, the REXXTOOLS QSAM interface can be used with PARSE VALUE and REXX concatenation to decompose and compose records as they are being read and written. EXECIO, because it is implemented as a host command, cannot be combined with REXX instructions.

The REXXTOOLS/MVS QSAM functions are:

OPEN a function for establishing a connection between your program and the data set. The OPEN function accepts arguments specifying the DCB parameters for the data set, as well as the processing option (INPUT, OUTPUT, UPDATE).

CLOSE a function for terminating the connection between your program and the data set.

GET The GET function is used to retrieve records sequentially. GET returns a logical record as its value (or a null string if an error was encountered).

PUT The PUT function is used to sequentially write or re-write records. PUT accepts a record as one of its arguments. If the PUT follows a GET where the file has been opened for update, the record that was retrieved by the GET will be replaced. In all other cases, PUT is interpreted as a request to add a new record.

BPAM

BPAM, which stands for Basic Partitioned Access Method, is used to process Partitioned Data Sets (PDS). Partitioned Data Set Extended (PDSE) data sets may also be accessed with BPAM. Unlike QSAM, BPAM permits multiple PDS members to be processed with just one OPEN/CLOSE sequence. Because of this, and because BPAM has functions for managing PDS/PDSE directories, it is often the preferred access method for processing partitioned data sets.

REXXTOOLS provides a full-function interface to BPAM. Using this interface you can create, update, and delete individual members and maintain PDS directory entries.

The REXXTOOLS/MVS BPAM functions are:

OPEN a function for establishing a connection between your program and the data set. The OPEN function accepts arguments specifying the DCB parameters for the data set, as well as the processing option (INPUT, OUTPUT, UPDATE). With just one OPEN function call, you can process one, some, or all of the members in a PDS or PDSE.

CLOSE a function for terminating the connection between your program and the data set.

FINDM The FINDM function is used to locate a member's position on DASD for reading or updating records. Optionally, the member's directory entry user field may be retrieved. For PDSEs, FINDM establishes a "connection" to a member. The connection mechanism permits PDSE members to be shared among multiple readers and writers.

GET The GET function is used to retrieve records sequentially. GET returns a logical record as its value (or a null string if an error was encountered).

PUT The PUT function is used to sequentially write or re-write records. PUT accepts a record as one of its arguments. If the PUT follows a GET where the file has been opened for update, the record that was retrieved by the GET will be replaced. In all other cases, PUT is interpreted as a request to add a new record.

STOWM
The STOWM function is used to add, replace, or remove a member's directory entry. STOWM also can be used to add, replace, or remove a member alias. Optionally, the member's directory entry user field can be created or updated. For PDSEs, STOWM terminates a connection to a member.

Other features of the BPAM interface include:

- Support for all record formats, except spanned.
- Direct support for creation, retrieval, and update of ISPF statistics.
- Support for application-defined directory entries.

- Support for high-performance, TTR-based look-up of members.

VSAM

REXXTOOLS/MVS provides a collection of REXX functions for working with VSAM data sets. The functions are closely modeled after the VSAM macro interface.

REXXTOOLS/MVS can be used to access the following types of VSAM data sets:

KSDS Key-sequenced data sets. The records of this type of data set are maintained in key-sequence order. The records may be of variable length, but the key field is always in the same location and is always the same length.

KSDS records may be accessed both sequentially and directly (randomly). Records may be inserted, both in between existing records and onto the end. Records may also be deleted (erased) from the data set.

The REXXTOOLS VSAM interface can be used with alternate indexes.

ESDS Entry-sequenced data sets. The records of the ESDS are maintained in the order they are entered. As new records are added, they are always placed at the end of the data set. Moreover, once a record has been added to an ESDS, it can never be deleted. Like the KSDS, the ESDS permits variable length records. However, the ESDS will not permit a record to grow or shrink once it has been added. A replacement record must fit exactly over the old record's space.

An ESDS may be accessed both sequentially and randomly. For random access, a number called the Relative Byte Address (RBA) is used to specify the record to retrieve. The RBA of a record can be obtained at the time it is inserted, or can be determined by sequentially reading the data set.

RRDS Relative record data set. The RRDS holds data that is accessed by the number of the record. The RRDS supports only fixed length records. Records can be inserted into the middle, or added onto the end of an RRDS. Records can also be deleted.

The RRDS permits both sequential and random access. Random access is by Relative Record Number (RRN), which is the sequence number of a record.

VRDS Variable-length relative record data set. The VRDS, like the RRDS, holds data that is accessed by the number of the record. In addition, the VRDS also supports variable length records. Records can be inserted into the middle, or added onto the end of a VRDS. Records may also be deleted.

The VRDS permits both sequential and random access. Random access is by Relative Record Number (RRN).

LDS Linear data set. This type of VSAM data set is used for the MVS Data-In-Virtual (DIV) service. It is also used by DB2 to contain table-related data. Linear data sets do not contain records in the same sense that KSDS, ESDS, and RRDS data sets do. Because of this, the record-oriented services of VSAM cannot read from or write to linear data sets. REXXTOOLS can, however, access linear data sets by control interval.

The REXXTOOLS/MVS functions for accessing VSAM data sets are:

OPEN a function for establishing a connection between your program and the data set. The OPEN function accepts an argument specifying the ACB options for the data set.

CLOSE a function for terminating the connection between your program and the data set. A temporary close is supported. This type of close forces buffers to be written and the catalog to be updated without terminating the connection between the program and the data set.

GET The GET function is used to retrieve records or control intervals.

PUT The PUT function is used to write records or control intervals. If the PUT follows a GET with the UPD (update) option, the record that was retrieved by the GET will be replaced. In all other cases, PUT is interpreted as a request to insert new records into the data set.

POINT The POINT function is used to position VSAM on a record in a VSAM file. For example, you can position VSAM on the last record in a file when you plan to read the file with the BWD (backward) option.

ERASE
The ERASE function is used to remove records from a KSDS or RRDS. You must GET a record before you can ERASE it. VSAM does not permit ERASE to be used on ESDS data sets.

VERIFYV
a function for correcting catalog information. In most cases this function is not required, since VSAM will perform implicit verifies.

IDCAMS

Using the REXXTOOLS IDCAMS interfaces you can:

- Define, modify, and remove catalogs, page spaces, VSAM and non-VSAM data sets, alternate indexes, generation data groups, etc.
- List or print catalog and data set information.
- Convert and back-up data

The primary IDCAMS interface is the IDCAMS host command environment. Using ADDRESS IDCAMS you can issue any IDCAMS command. For the list of IDCAMS commands, consult the Access Method Services publication for your system's level of DFP or DFSMS.

REXXTOOLS also provides the following IDCAMS functions:

DSNDEL
a function for deleting data sets.

LISTC a function for listing catalog information. The information is returned in a more "programmer friendly" format than is provided by the LISTC command.

The REXXTOOLS/MVS IDCAMS interface dynamically links to your system's IDCAMS module. When you install a new release of DFP or DFSMS, any additional capabilities provided by the new release will be available through the REXXTOOLS IDCAMS interface, without any other action on your part.

Neither the IDCAMS host command environment nor the IDCAMS functions require TSO services. These interfaces can be used in any address space.

Data Set Information

Most high-level language programs process a fixed number of data sets and are generally "unaware" of the data set environment in which they run. REXXTOOLS provides functions for extracting environmental information that may be used to generalize data set processing.

The REXXTOOLS data set information functions are:

DDNINFO
returns in-memory allocation and DCB information for a ddname.

DSNINFO
returns catalog, VTOC, space allocation and utilization, SMS class, and ddname information for single-volume DASD data sets.

LISTA returns allocation information for one or more ddnames. A ddname pattern containing wildcard characters may be specified.

LISTM returns PDS/PDSE directory information for one or more members. Wildcard pattern matching is supported.

MVS Supervisor Services

REXXTOOLS/MVS includes functions for accessing z/OS control program services. Using these services, you can write programs that:

- Explicitly manage virtual storage.
- Obtain system-wide control of resources before using them.
- Query the system authorization facility about the user's authority to use a resource.
- Write messages to the system log and to the system operator's console.

Notes:

1. All REXXTOOLS/MVS facilities operate in problem state. Privileged operations (those requiring APF authorization, supervisor state, or key zero) are not supported.

2. Any REXXTOOLS function that you do not want your programmers to use can be removed when the product is installed.

The MVS Supervisor Services functions are:

DEQ releases a previously reserved (with ENQ) resource.

DOM removes a message from a system console.

ENQ reserves a resource, such as a data set.

FREEMAIN
releases previously allocated (with GETMAIN) storage.

GETMAIN
allocates an area of main storage.

POST signals the completion of an event.

RACROUTE
verifies that a user is authorized to use a resource.

WAIT waits on the completion of an event. A special form is provided for timed waits.

WTL writes a message to the system log.

WTO writes a message to a system console.

WTOR writes a query message to a system console and waits for the operators reply (a time-out function is supported).

TSO Services

Using the TSO Services functions you can create line-mode and full-screen applications outside of ISPF. The following functions implement the REXXTOOLS TSO Services:

SBA a function for converting line/column coordinates to terminal device addresses.

TGET a function for reading from the terminal device. Line-mode and full-screen operations are supported.

TPUT a function for writing to the terminal device. Line-mode and full screen-operations are supported.

General REXX Extensions

The programs of this section are designed to improve the general usability of the REXX language and to increase the productivity of REXX programmers. These programs may be categorized as follows:

Data conversion.
Functions are provided for converting to and from specialized numeric formats:

D2P/P2D Bi-directional packed decimal conversions.

D2F/F2D Bi-directional floating point conversions.

D2PIC/PIC2D COBOL-style numeric editing and de-editing.

Stem handling.
Functions are provided for sorting (STEMSORT) and displaying (STEMDISP) the contents of certain types of stem variables (those that contain numeric subscripts).

String handling.
A function is provided for breaking a string into its constituent tokens where the location and frequency of delimiters is not predictable in advance (PARSETOK). Another function is used to sort the blank delimited words in a string into either ascending or descending collating sequence (WORDSORT).

Pattern Matching.

The MATCH function is used to compare strings to a pattern. The pattern may contain wild card characters.

MVS/Quick-Ref access.

A function is provided for accessing the MVS/Quick-Ref database (QWIKREF). The results of a successful search are placed in a user-specified stem array.

Note: Requires Chicago-Soft's MVS/Quick-Ref Release 3.0 or later.

REXX Environment Control.

A function, RXFUNC, is provided for maintaining entries in REXX function package directories. Another function, RXSUBCOM, allows you to maintain entries in the host command environment table.

Storage Manipulation.

The STORAGEX function gives the REXX programmer direct access to processor main storage, using a relative addressing syntax similar to the one used by TSO TEST.

Global Variables.

A host command environment (ADDRESS REXXTOOL) and several host commands (VPUT, VGET, VERASE) are provided to permit the sharing of data between REXX programs.

Dynamic Allocation

REXXTOOLS/MVS provides implementations of the ALLOCATE and FREE commands. Like their TSO counterparts, the REXXTOOLS/MVS ALLOCATE and FREE commands are invoked by coding host command expressions in REXX programs. Unlike the TSO commands, you can use the REXXTOOLS ALLOCATE and FREE in non-TSO environments. The REXXTOOLS commands also use less CPU time than the TSO commands.

Using the REXXTOOLS/MVS ALLOCATE command you can:

Create new data sets.

The data sets can be assigned attributes (e.g., DCB) directly on the command, or indirectly by referring to model data sets or SMS classes.

Associate a ddname with a data set.

The ddname can be referenced by the EXECIO command and by the REXXTOOLS/MVS VSAM functions. REXXTOOLS ALLOCATE accepts relative generation numbers (TSO ALLOCATE does not).

Concatenate several data sets.

The data sets can be grouped together under one ddname.

Allocate SYSOUT data sets.

The output can be routed to writers, printers, and other systems.

Using the REXXTOOLS/MVS FREE command you can:

Disassociate a data set with your program.

The data set then becomes available for other users in the system.

Override the disposition and destination of data sets.
The new values override the values that were specified at data set allocation.

In addition, the REXXTOOLS/MVS ALLOCATE and FREE commands return extensive diagnostic information to your REXX program, including:

- The text of allocation messages.
- System generated ddnames.

You may use this information to drive conditional logic in your programs.

Note: REXXTOOLS includes a program that may be called from high-level languages (COBOL, PL/I, etc.) to perform dynamic allocation functions.

Interpretive Compiler

The REXXTOOLS/MVS interpretive REXX compiler is used to create load modules from REXX source code. The compiler-generated object modules are in standard OS format. They can be link-edited and invoked in the same manner as modules produced by other language compilers. Compiled REXX programs have significant advantages over source REXX programs:

Source code can be hidden from end users.
The source is encapsulated in a load module and can be encrypted to prevent browsing.

Load modules are loaded into memory faster.
Compiled REXX programs permit load optimizations, including the placement of highly used modules in REXX function packages, or the system link pack area (LPA).

Greater flexibility.
Execute compiled REXX programs using standard load module invocation methods.

Program listings and symbol cross-references are provided.
The additional documentation provided by a source listing eases application support.

The REXXTOOLS/MVS interpretive compiler supports the complete REXX language (including the INTERPRET instruction), as well as all of the published interfaces described in the IBM publication *REXX/MVS Reference*.

Notes:

1. Use of the compiler requires a REXXTOOLS/MVS license. A license is not required to execute compiled REXX programs, unless the compiled programs utilize other REXXTOOLS/MVS services.

2. The REXXTOOLS compiler does not enhance execution speed. If you require faster execution use the REXX/370 compiler from IBM.

External CICS Interface

The REXXTOOLS/MVS EXCI interface gives REXX programmers access to CICS transactions written to the Distributed Program Link (DPL) specification. The REXXTOOLS function **REXCI** (see the file “REXCI.pdf” located in \OST\RXT\V070405\Files) is implemented over the IBM-

supplied EXCI "call" interface. The CICS EXCI interface is described in the IBM manual *CICS External Interfaces Guide*.

External Writer Facility

The REXXTOOLS/MVS REXX external writer facility is a program and a collection of REXX functions that permit the development of external writers in REXX. An external writer is a program, external to the Job Entry Subsystem (JES), that processes JES spool data sets (also called SYSOUT data sets). External writers can examine, copy, and/or delete data sets from the JES spool. The actual processing performed is left to the discretion of the application programmer.

The writer facility is implemented over MVS's SYSOUT Application Programming Interface (SAPI). Because of this, it can be used with either JES2 or JES3.

DB2/SQL Services

REXXTOOLS/MVS gives you two ways to access DB2 from REXX:

The Dynamic DB2/SQL Service.
: This facility lets you code and run programs without preprocessing or binding. Using Dynamic SQL, you can quickly develop applications. All data conversions are handled by the interface and you can skip most of the "set-up" statements, such as table and variable declarations.

The Static DB2/SQL Service.
: This facility provides a preprocessor for creating Data Base Request Modules (DBRMs). The DBRMs can be bound to packages and plans. Because the generated plans are application-specific, data and programs can be protected by limiting execution authority. The Static SQL Service also enhances application performance since expensive run-time operations like statement PREPAREs are unnecessary. The interface performs all data conversions automatically.

In addition, a REXX function, **DB2CMD** (see “DB2CMD Function” located in the “Reference” chapter of the DB2/SQL Services User’s Guide), permits authorized users to issue DB2 commands.

Supported SQL

The following table describes the SQL statements supported by the DB2/SQL Services. For more information on what these commands do, and their exact syntax, refer to the *DB2 SQL Reference* for your system's level of DB2.

Command	Dynamic	Static	Description
ALTER	yes	yes	Structurally modifies certain DB2 objects.
BEGIN DECLARE SECTION	yes	yes	Indicates the beginning of variable declarations.
CLOSE	yes	yes	Closes a cursor.
COMMENT	yes	yes	Updates the descriptions of various DB2 objects.
COMMIT	yes	yes	Terminates a unit of work and makes permanent any changes which may have taken place. Also closes "no hold" cursors.
CONNECT	yes	yes	Connects an application process to a server.
CREATE	yes	yes	Creates various DB2 objects.
DECLARE CURSOR	yes	yes	Associates a SELECT statement with a cursor.
DECLARE TABLE	yes	yes	Used to document the structure of a table.
DELETE	yes	yes	Used to delete rows from a table or a view. Can be used in "searched" or "positioned" modes.
DESCRIBE TABLE	yes	no	Places column information into an SQLDA.
DROP	yes	yes	"Un-creates" or removes DB2 objects.
END DECLARE SECTION	yes	yes	Marks the end of variable declarations.
EXPLAIN	yes	yes	Obtains path selection information about statements that use searches.
FETCH	yes	yes	Retrieves a row from an open cursor's result set.
GRANT	yes	yes	Authorizes access to DB2 objects.
INSERT	yes	yes	Adds new rows to a table or a view.
LABEL	yes	yes	Updates the labels associated with various DB2 objects.
OPEN	yes	yes	Opens a cursor for processing.
RELEASE	yes	yes	Releases connections.
REVOKE	yes	yes	Removes authorizations.
ROLLBACK	yes	yes	Terminates a unit of work and cancels any changes that may have been made. Closes both "hold" and "no hold" cursors.
SELECT INTO	yes	yes	Retrieves rows of information into host variables. The REXXTOOLS implementation retrieves multiple rows.
SET CONNECTION	yes	yes	Establishes an application server.
SET CURRENT DEGREE	yes	yes	Controls use of parallel I/O.
SET CURRENT PACKAGESET	yes	yes	Changes the value of the CURRENT PACKAGESET special register.
SET CURRENT SQLID	yes	yes	Changes the SQL authorization ID.
Other SETs	Yes	No	See the *DB2 SQL Reference*. SET CURRENT PACKAGE PATH is not supported.
SET :var = special register	yes	yes	Assigns the value of a special register into a host variable.
UPDATE	yes	yes	Changes the rows in a table or view. Can be used in "searched" or "positioned" modes.

Special Statements

The DB2/SQL Services include several SQL extensions. These are:

Multi-row SELECT A statement for retrieving multiple rows from a table or view.

OPTIONS A statement for setting REXXTOOLS/MVS processing options.

DECLARE VARIABLE A statement for specifying the type and size of host variables.

FREE VARIABLE A statement for removing variable definitions.

FREE CURSOR A statement for removing cursor definitions.

Host Variables

Program variables may be used to pass information to and receive information from DB2. Indicator variables are supported.

Binary integer, packed decimal, and floating point conversions are automatic and bi-directional. However, the interface does require you to indicate, for numeric input variables only, the types of conversions you need.

Call Attachment Facility

Before a program can pass SQL statements to DB2, a connection known as a "thread" must be established between the task under which the program is running and the DB2 address space. In addition, a plan -- a DB2 object that indicates what kind of processing your program will do -- must be opened. DB2 provides several facilities for creating threads and opening plans. The facility employed by the DB2/SQL Services is the Call Attachment Facility (CAF).

The CAF is a general purpose facility that can be used to establish threads from nearly any MVS address space (IMS and CICS are special exceptions). The DB2/SQL Services provide two ways to use the CAF:

- The CAF is called implicitly, whenever an SQL statement is executed and the SQL host command environment or the static SQL run-time routines do not detect an established thread.
- You can invoke the CAF explicitly using the REXXTOOLS DSNALI function.

Dynamic SQL

The Dynamic DB2/SQL Service provides a command-based interface to DB2. The SQL statements accepted by the interface are classified as follows:

Dynamic SQL Statements.
These statements are PREPAREd and EXECUTEd while your program is running. The Dynamic SQL Service does the work of PREPAREing and EXECUTEing these statements for you.

Pseudo-static SQL Statements.

These statements must be statically prepared and identified to DB2 prior to the execution of your program. The Dynamic SQL Service keeps a pool of generic static SQL statements to be used by the interface whenever they are needed. The pooled statements are:

CLOSE	CONNECT	DECLARE CURSOR
DESCRIBE TABLE	FETCH	OPEN
SET CURRENT PACKAGESET	SET CURRENT SQLID	SET :hostvar = special register

The Multi-row SELECT Statement.

The Dynamic SQL Service provides special support for processing sets of rows using the SELECT command (similar to SPUFI SELECTs).

Interface Control Statements.

These statements are used to change the processing characteristics of the interface.

Static SQL

The Static SQL Service lets your programs take advantage of true DB2 static SQL. Static SQL programs have 2 key advantages over dynamic SQL programs:

- The Static SQL precompiler creates a separate DBRM for every program it processes. These customized DBRMs can be bound into plans or packages which can be protected individually using the GRANT/REVOKE statements of DB2.

- Because DB2's optimizer is able to determine access paths at BIND time, a static SQL program typically outperforms dynamic SQL programs. This is especially important for short-duration but frequently invoked transactions.

Because static SQL programs require preprocessing, they are not as easy to develop and debug as programs developed under the Dynamic SQL Service. Ideally, you should use the Dynamic SQL Service to develop your programs, and the Static SQL Service to ready them for production.

Installation

This section discusses REXXTOOLS/MVS installation.

Very Important: REXXTOOLS/MVS runs in problem state only. All modules are link-edited AC=0. Privileged operations -- those requiring APF authorization, supervisor state, or special protect keys -- are not supported.

Media

REXXTOOLS/MVS is distributed on the following media:

- Electronic mail attachment
- CD

Prerequisite Software

The following levels of systems software products are prerequisite to REXXTOOLS/MVS Version 7.4.5:

1. z/OS® Version 1.13, 2.1, 2.2, or 2.3.
2. FTP (for transferring REXXTOOLS/MVS to an IBM z/OS operating system).
3. If you plan on using REXXTOOLS/MVS under NetView®, verify that you have NetView Version 1.3 (or later) installed.
4. If you are installing Dynamic SQL Services, verify that you have DB2 Version 10, 11, or 12.
5. If you are installing Static SQL Services, verify that you have DB2 Version 2.2 through 12.

Authorization Considerations

In order to install REXXTOOLS/MVS, your TSO user ID must be authorized to:

1. Run the High Level Assembler (ASMA90) to install product authorization codes and to set product parameters.

2. Catalog new, or modify existing system-related data sets. Depending on how you install REXXTOOLS/MVS, you may need the authority to create data sets with system-related high level qualifiers like 'SYS1'.

3. Modify TSO LOGON procs, if you install REXXTOOLS/MVS in TSO STEPLIBs.

4. Modify PARMLIB members, if you install REXXTOOLS/MVS load modules in the link list or in the link pack area.

5. Run SMP/E jobs, if you are using a USERMOD to install the product function package into 'SYS1.LINKLIB'.

6. Define CICS CONNECTION and SESSION resources, if you are using the CICS External Interface (REXCI).

7. Define and permit access to profiles in the JESSPOOL class, if you are using the External Writer Facility (RXTWTR).

8. If you are installing the product under NetView you will need the authority to:

    - Modify the NetView proc in SYS1.PROCLIB.
    - Add members to the DSICLD (REXX program) data sets.
    - Add members to the STEPLIB (or add new data sets to the STEPLIB concatenation of data sets).
    - If you are adding the product load library to the STEPLIB concatenation of data sets, you will need to be able to APF authorize the load library.
    - Stop and restart NetView.

9. If you are installing the product under IBM's HTTP Server you will need the authority to:

    - Modify the IMWEBSRV proc in SYS1.PROCLIB.
    - Add REXX programs to the CGI-BIN directory.
    - Add members to the STEPLIB (or add new data sets to the STEPLIB concatenation of data sets).
    - Stop and restart IMWEBSRV.

10. To install either of the DB2/SQL components, will also need authorization to:

    - Bind Data Base Request Modules (DBRMs) into plans and/or packages.
    - Grant authorizations for plans, tables, and other DB2 resources.

Installation Overview

Installation of REXXTOOLS/MVS takes about 30 minutes, provided you have the proper levels of authorization, and assuming you do not need to create customized REXX parameters modules or function packages.

The basic installation steps for REXXTOOLS are as follows:

1. **Unload the Data Sets**. REXXTOOLS is shipped as an email attachment or on CD. In this step you create the installation data sets from the shipping media.

2. **Allocate the Load Library**. The installation data sets do not include the load library for REXXTOOLS executable modules. In this step you create this library.

3. **Link the Load Library**. REXXTOOLS is shipped in object format. In this step you link-edit the object modules into the load library.

4. **Set Product Authorization Codes**. In this step you install the authorization codes that activate REXXTOOLS on your system.

5. **Set Default Parameters**. In this step you set-up default values for various aspects of REXXTOOLS.

6. **Customize the REXX Environment**. In this optional step, you change how REXXTOOLS interfaces with REXX.

7. **Make the Load Library Visible**. In this step you change system PROCs or parameters so that users can access REXXTOOLS load modules.

8. **Make the EXEC Library Visible**. In this step you change system PROCs or logon execs so that users can access REXXTOOLS REXX programs.

9. **Make the PROCs Visible**. In this step you add JCL procedures to system libraries so that users can access them in their jobs.

10. **Verify the Installation**. In this step you run a job to verify that the installation worked. The job produces a report that is useful in diagnosing installation problems.

11. **Distribute the Documentation**. In this step you make the documentation available to the members of your staff who will be using REXXTOOLS.

If you are installing Dynamic SQL Services you must also bind the product plans.

Support

Open Software Technologies provides technical support for REXXTOOLS/MVS on a 24X7 basis. North American customers can contact us using any of the following methods:

Phone: (407) 788-7173

FAX: (407) 788-8494

Email: support@open-softech.com

Customers who purchase REXXTOOLS through a distributor should contact the distributor for technical support. However, we are always happy to hear from our customers, and encourage all of them to contact us with ideas and suggestions, as well as technical questions.

If you purchase REXXTOOLS/MVS, the first year's support is provided at no additional charge. After the first year you have the option to purchase annual maintenance for the product. Customers with up-to-date maintenance contracts receive new versions and releases of REXXTOOLS/MVS at no additional charge.

Customers who lease REXXTOOLS are entitled to technical support for the term of the lease.

Case Studies

This section describes examples of REXXTOOLS/MVS usage to solve real-world, line-of-business problems.

Inter-bank Funds Transfer Application

Problem: A well-known New York-based bank processes tens of thousands of inter-bank funds transfers daily. Previously, the bank had been using a "screen scraper" product to post transaction information into a CICS application used by financial analysts. The system worked well, but because of the large number of CICS logons it generated, an inordinate amount of CPU time was being used.

Solution: The REXXTOOLS VSAM interface was used in conjunction with AF/OPERATOR ® to post the transactions directly to the CICS KSDS. This resulted in savings estimated to exceed one million dollars each year. An additional benefit is that transactions continue to be spooled to the VSAM file even when the CICS region is down (previously transactions were rejected when this happened). When the region is restarted, analysts can access the most recent transactions immediately.

Source Library Management and Control

Problem: A large airline reservations organization developed and maintains many large proprietary applications. As with any large development effort, automated processes were required to keep track of the many components that compose an application. Because of the many platforms and software products used by the organization's application developers, they were unable to find an off-the-shelf product that would satisfy their requirements.

Solution: An in-house developed source library management and control system was developed using REXX and the Basic Services of REXXTOOLS/MVS. The ISPF-based system uses VSAM KSDSes for the central repository. The REXXTOOLS VSAM functions are used to query and update these data sets.

Insurance Claim Error Correction

Problem: A large US insurer processed thousands of claims each day. Due to errors in data entry, some of the claims had to be manually scanned for errors and corrected. The correction process involved comparing rejected claims with information concerning the company's 14 million clients. At any one time, about 100 clerical workers were involved in the claims correction process. The company decided to convert the paper-intensive claims correction operation to an on-line system. However, due to downsizing efforts, the number of programmers available to construct the application was severely diminished.

Solution: REXX and REXXTOOLS/MVS Basic Services were selected to form the foundation for the ISPF-based system. The REXXTOOLS VSAM interface was used to retrieve customer information from the 14 million row KSDS. The system was completed on-time and accepted by the claims analysts. Just one programmer was needed to create the application.

NetView-based Help Desk

Problem: A well-known national bank built and maintained a large SNA network. To manage the network they chose Netview as the primary platform. DB2/MVS was used as the repository for network components (front-ends, controllers, terminals, etc.). The information in the network repository was used by help desk workers to troubleshoot network problems. Because Netview did not include a REXX-to-DB2 interface, network analysts were required to switch between an in-house developed CICS application and the NCCF console to solve problems. This lengthened the time it took to resolve problems and was the source of transcription errors.

Solution: The REXXTOOLS/MVS Dynamic DB2/SQL Service was used to provide access to the network repository directly from the NCCF console. Fewer than 1000 lines of REXX and SQL code were needed to implement this function.

CA-Endevor "Bridge" Application

Problem: A large US telecommunications company, and a large DP consultancy, teamed-up to create a unified billing system to service the telcom's business and residential customers. The scope of this effort was enormous, involving over 200 application programmers. CA-Endevor from Computer Associates was chosen to perform the function of source management and control. However, special impact analysis and reporting functions were required which could not be obtained from Endevor.

Solution: REXX and the Dynamic DB2/SQL Services component of REXXTOOLS were used to create the special impact analysis reports. The ISPF application allows the user to specify analysis parameters and then produces -- at the user's option -- either an on-line or batch report.

EDI Transaction Processing

Problem: Petroleum producers transmit production, tax and royalty data to the State of New Mexico Information Systems Division via Value Added Networks (VANs) and tapes. The transactions are submitted in standard Electronic Data Interchange (EDI) format. The state uses this information to determine the taxes owed by individuals and companies operating within the state. Several years ago, the state hired an outside firm to produce an application to process these transactions. The contractor wrote portions of the code in REXX, because of its superior string handling (REXX was used to parse the EDI packets), and other portions in COBOL and SQL. Due to the structure of the application, an inordinate amount of CPU time was being used as the application frequently switched between REXX and COBOL.

Solution: The REXXTOOLS interpretive compiler was used to compile frequently called REXX modules. This permitted the COBOL and REXX programs to be link-edited together for faster invocation. As a result, the state estimates it has saved five hundred thousand dollars in CPU charges in two years’ time. During this same time period, the Information Systems Division has switched from using COBOL to REXX/REXXTOOLS for "one-shot" DB2 report programs.

RACF-to-DB2 Help Desk Application

Problem: The RACF administration group of the Florida Department of Banking and Finance was charged with creating a help desk application to handle customer user ID problems. The application needed to be able to handle "fuzzy" queries, where perhaps only the customer's name was available (e.g., given the customer's name, or a portion thereof, report the user ID and the name of the group administrator for the customer). Due to budgetary restrictions, tight restrictions were placed on the number of programmers available to create the application (in the end, one did most of the work).

Solution: REXX and REXXTOOLS/MVS were selected as the development platform for the ISPF application. The application uses a RACF utility to unload the live RACF database to a collection of DB2 tables. The REXXTOOLS Dynamic DB2/SQL Service is used to provide access to the DB2 tables. The project was completed on time and within budget. Currently, the RACF group is considering re-hosting the help desk application under IBI's Web390 web server. Preliminary testing indicates that REXXTOOLS works well with the Information Builders product.

DB2 Application Error Checking

Problem: A large international insurer uses DB2 in some of its in-house developed applications. As with any large development effort (some of the applications contain thousands of modules), maintaining control of an application's components is a serious concern. In particular, the company's data base administrators are interested in ensuring that all components of an application -- load modules, DBRMs, and DB2 plans -- are at the same level, because component mismatches can cause expensive production outages. Manually comparing an application's components was deemed infeasible due to the size of the problem and the inherent inaccuracy of "eyeball" comparisons.

Solution: A REXX/REXXTOOLS application was developed to scan an application's load modules for DB2 precompiler time stamps. The time stamps are compared to time stamps in the application's DBRMs and DB2 plans. The REXXTOOLS Dynamic DB2/SQL Service is used for the DB2 catalog queries, and the Basic Services' BPAM functions are used to read application load modules (Note: load modules are U-format data sets and cannot be read with the REXX EXECIO command).

DB2 Application Testing

Problem: A paper company made extensive use of DB2 in its applications development organization. To cope with ever-changing regulatory environments, DB2 application developers were constantly modifying the application programs. Testing the changes was difficult because they had no way of recreating the production database environment on their development systems.

Solution: A REXX/REXXTOOLS application was developed to copy selected portions of production databases to the test environment. The REXXTOOLS DB2/SQL Dynamic SQL service was used to query production system catalogs to create Data Definition Language (DDL) statements to be executed on the test system. Subsequently, production data was extracted and copied to the test tables. In fewer than 1000 lines of REXX/REXXTOOLS code, one programmer was able to accomplish a function for which some system vendors charge tens of thousands of dollars.

Sample Application

The following program demonstrates REXXTOOLS/MVS in a real-world application.

Internet CGI Script Using DB2/SQL

The following program demonstrates the REXXTOOLS dynamic SQL interface in a CGI script under IBM's HTTP Server.

```
/* REXX RXTCGI - REXXTOOLS DB2 INTERFACE UNDER HTTP SERVER
**
**              PROPRIETARY AND CONFIDENTIAL INFORMATION
**   AND INTELLECTUAL PROPERTY OF OPEN SOFTWARE TECHNOLOGIES, INC.
**      (C) COPYRIGHT 1991-2013, OPEN SOFTWARE TECHNOLOGIES, INC.
**                       ALL RIGHTS RESERVED
**
**   NAME       - RXTCGI
**   AUTHOR     - E.D. HODIL
**   PURPOSE    - Demonstrates the REXXTOOLS Dynamic SQL feature
**                under IBM's HTTP Server.
**   NOTES      - NONE
**   REFERENCES - 1. TSO/E VERSION 2 REXX/MVS REFERENCE, SC28-1883.
**                2. DB2 VERSION 3 SQL REFERENCE, SC46-4380.
**   RELATED    - NONE
**   PARMS      - NONE
**   KEYWORDS   - NONE
**   LANGUAGE   - TSO/E REXX
**   HISTORY    - 16 JAN 1997 EDH  - 05.01.01 ORIGINAL
**
*/

/*
** Write the HTTP headers.
*/
   parse source . . . . . . . AddressSpace .
   if pos(AddressSpace,'TSO/E ISPF') = 0 then
     "cgiutils -status 200 -ct text/html"
/*
** Write the HTML prolog.
*/
   say "<html>"
   say "<head>"
   say "<title>REXXTOOLS DB2 Query</title>"
   say "</head>"
   say "<body>"
/*
** Try to dynamically add the SQL host command environment. There's
** no problem if it is already there. We'll just get RC=4.
*/
   call rxsubcom 'add', 'SQL', 'rxtasql2'
   if rc > 4 then do
     say '<h1>RXSUBCOM failed. RXTCGI Terminating.</h1>'
     say '<p>RC='rc 'REASON='reason
     signal exit
```

```
   end
/*
** Get the local defaults for DB2. In particular, we want the local
** DB2 subsystem name.
*/
   if db2info('defaults') <> 0 then do
     say '<h1>DB2INFO call failed. RC='rc'</h1>'
     signal exit
   end
/*
** Use the Call Attachment Facility (CAF) to open one of the
** product plans (in this case we use RXTCS for cursor
** stability). $RXTDB2_SSID is a variable created by DB2INFO.
*/
   call dsnali 'open', $rxtdb2_ssid, 'rxtcs'
   if rc <> 0 then do
     say '<h1>OPEN FAILED. RXTCGI Terminating.</h1>'
     say '<p>RC='rc 'REASON='reason
     signal exit
   end
/*
** Declare variables. We really could skip this step since the only
** variable passed to DB2, LN, contains only character data, which
** is the default. Declarations for output variables, the "begin/end
** declare section" statements, the "exec sql" prefix, and the
** terminating semicolons are coded only for compatibility with
** the static sql feature. The dynamic sql interface does not
** require any of these (but they do make for nice documentation!).
*/
   address SQL
   "exec sql begin declare section;"
   "exec sql declare ln       variable varchar(15);"
   "exec sql declare lname    variable varchar(15);"
   "exec sql declare lni      variable smallint;"
   "exec sql declare sal      variable decimal(9,2);"
   "exec sql end declare section;"
/*
** Declare the cursor that we will use. The input host variable
** will not be evaluated until we perform the OPEN below. Note that
** for "DECLARE" statements (variables and cursors) we do not do
** any RC checking. That is because the interface will produce
** error messages if anything is wrong. It is also because these
** statements are not executable in the static interface.
*/
   address SQL
   "exec sql declare empcsr cursor for",
    "select lastname, salary",
       "from dsn8310.emp",
       "where lastname like :ln;"
/*
** Now open the cursor. This is when DB2 creates the results table
** by evaluating the SELECT command. If you change the value of LN,
** remember that DB2 comparisons are case sensitive!
*/
   ln = 'S%'
   "exec sql open empcsr;"
   if rc <> 0 then do
```

```
      say '<h1>Open cursor failed with RC='rc 'REASON='reason'</h1>'
      say '<p>'SQLCODE='sqlca.sqlcode
      say '<p>'MESSAGE='sqlca.sqlmsg
      signal exit
    end
/*
** Loop through the results table, fetching rows into the scalar
** variables listed on the FETCH command. Note that we only need
** to check the RC variable for our loop control. The FETCH after
** the last row will yield an SQLCODE of 100, and an RC value of 1.
** Note also the use of the D2PIC function to nicely format salary.
*/
    say "<h1>REXXTOOLS/MVS DB2 Query Results</h1>"
    say "<p><p>"
    say "<strong>DB2 SSID=</strong>" $rxtdb2_ssid "<p>"
    say "<strong>DB2 LVL =</strong>" $rxtdb2_lvl "<p>"
    say "<hr>"
    say "<pre><p>"
    "exec sql fetch empcsr into :lname:lni, :sal;"
    do while rc = 0
      say left(lname,15)'('lni')'  d2pic(sal,'$,$$$,$$9.99')
      "exec sql fetch empcsr into :lname:lni, :sal;"
    end
/*
** Close the cursor. If we didn't do it task termination would
** close it for us.
*/
    "exec sql close empcsr;"
/*
** Explicitly free our declared variables (optional, but tidy).
*/
    "exec sql free * variable;"
/*
** Explicitly close the plan.
*/
    call dsnali 'close', 'sync'
/*
** Write the HTML epilog.
*/
exit:
    say "</body>"
    say "</html>"
    exit 0
```

What's New in REXXTOOLS/MVS

What’s New in Version 7 Release 4 Mod 5

Release Date: June 2018

Consolidated product updates.

Dynamic SQL Services

Relaxed restrictions on DB2INFO usage.

The SET statements supported by DB2 V11, with the exception of SET CURRENT PACKAGE PATH, have been added. The MERGE statement is now supported. Support for common table expressions. New option CACHE/NOCACHE. Data-change-table-reference (SELECT from INSERT/UPDATE/DELETE/MERGE).

What's New in Version 7 Release 4 Mod 3

Release Date: July 2014

Basic Services

A new function DDINFO has been added. DDINFO returns for ddnames the same information that DSNINFO returns for datasets.

DSNINFO and DDINFO now identify LARGE and EXTENDED datasets.

Dynamic SQL Services

A new installation parameter, DB2ISEM, has been added that controls how the DB2INFO function sources the DEFAULTS information.

What's New in Version 7 Release 4

Released September 2013

Basic Services

DSNINFO has been enhanced to handle requests for information on datasets residing on EAV volumes.

Dynamic SQL Services

REXX Stored Procedures

REXX stored procedures are now supported. A REXX stored procedure is a REXX program that runs in an address space separate from its caller, and possibly on a different machine. It performs some task or set of related tasks for the calling program and, optionally, returns results.

What's New in Version 7 Release 3

Released September 2011

Dynamic and Static SQL Services

These services have been updated to support New Function Mode (NFM) in DB2, especially DB2 Version 10. Earlier releases of REXXTOOLS/MVS might not work in NFM for future releases of DB2.

What's New in Version 7 Release 2

Released February 2010

Basic Services

A new facility for managing events with REXX programs, RXTEVENT, is provided. RXTEVENT may be run as a started task or a batch job.

Version 2 message descriptors MQMD are now supported and are the default for the REXX MQSERIES Interface.

What’s New in Version 7 Release 1

Released July 2005

Basic Services

A new facility, “REXX MQSERIES Interface,” is introduced. It consists of a single REXX function RXMQ (see the file “RXMQ.pdf” located in \OST\RXT\V070405\Files). The function has many sub-functions for getting data into and out of Websphere MQ.

The Interpretive compiler now supports a NONAMES option for hiding all names within the object module.

The RXSUBCOM function has had all authorization checking removed to allow for peaceful coexistence with IBM's RXSUBCOM.

What's New in Version 6 Release 1

Released June 2002

Basic Services

CICS External Interface

A new function, **REXCI**, is provided for accessing CICS transactions. The CICS transactions must be written to the Distributed Program Link (DPL) specification. REXCI uses IBM's EXCI "call" interface.

External Writer Facility

A new batch program, **RXTWTR**, is provided for developing external writers in REXX. RXTWTR uses IBM's SYSOUT Application Programming Interface (SAPI) to select and process JES spool data sets. Many selection parameters are available as are many data items concerning selected data sets. The writer program has the option of reading the data sets and changing their disposition and other characteristics.

VSAM

The VSAM interface has been improved to support extended addressing data sets. The interface automatically detects when an extended addressing data set is being used. The interface will accept and return RBA values larger than 4G as needed.

QSAM

The QSAM interface has been improved to support data sets with spanned records. Records up to 32756 bytes can be read and written.

REXX Compiler

The interpretive compiler accepts a new keyword, PREFIX. The prefix keyword can be used to specify either the RXTRXPRE (the default) or RXTRXPRC prefix modules. This simplifies link-edits when using RXTRXPRC.

RXTEXEC and RXTJCL

RXTEXEC and RXTJCL are included for compatibility, but their use is deprecated. You no longer can use either of these facilities to skip the basic installation steps. Documentation for RXTEXEC and RXTJCL has been removed.

Dynamic and Static SQL Services

A new REXX function, **DB2CMD**, has been added. This function permits authorized users to execute DB2 commands. Command output is returned to the program.

Architectural Improvements

The following improvements have been made:

1. The REXXTOOLS load library is no longer shipped. 2 jobs, RXTALLOC and RXTLINK, are supplied to create the load library. This makes it possible to block the load library in a way that conforms to local standards.

2. The restriction of 20 authcodes per copy of REXXTOOLS has been removed. It is now possible to create one copy of REXXTOOLS that you can distribute to all of your authorized CPUs. A new RXTAUTH job is supplied.

3. Installing default parameters has now been made easier. A new job, RXTPARMS, is supplied for setting these parameters.

What's New in Version 5 Release 1

Released March 1997

Basic Services

QSAM

Queued Sequential Access Method support has been added. Implemented as REXX functions, this interface permits access to physical sequential data sets and members of partitioned data sets. All record formats, except spanned, are supported.

BPAM

Basic Partitioned Access Method support has been added. Implemented as REXX functions, this interface permits access to partitioned data sets (PDS and PDSE). The interface permits multiple members to be read, written, or updated with a single OPEN/CLOSE sequence. Directory entries can be created, updated, read, and deleted. You can also create, update, and delete member aliases. A directory entry's user field may be created and updated in raw binary format or in ISPF statistics format. All record formats, except spanned, are supported. The interface presents a logical record interface to the user.

Data Set Information

New functions are provided for retrieving data set information. Catalog, VTOC, SMS, allocation (ddname), space allocation and utilization, and PDS directory information are provided by this collection of functions. The new functions are:

DSNINFO retrieves 37 data items for cataloged and uncataloged data sets. Included in this information are volume serial number, unit type, DCB parameters, space allocation and utilization, and SMS classes.

DDNINFO retrieves 16 data items for allocated data sets, including DSNAME, volume serial number, unit type, DCB parameters, status and dispositions.

LISTA lists current allocation information, including DSNAMEs, DSORGs, status, and dispositions. DDNAME patterns may be specified to limit the DDNAMEs listed.

LISTM lists PDS/PDSE directory information. Reports TTR, alias, and user field information (in raw or ISPF stats formats). Member patterns may be used to limit the members listed.

The data set information functions can be used in any address space (TSO is not required).

IDCAMS

A new host command environment is supplied as well as 2 IDCAMS-based functions. The IDCAMS host command environment allows IDCAMS commands to be embedded in REXX programs. IDCAMS messages are returned in special stem variables. The IDCAMS functions simplify common tasks. These are:

LISTC lists catalog information. Based on the LISTC command, this function returns one record of information for each data set in fixed column format.

DSNDEL deletes and scratches data sets. Based on the DELETE command, this function also supports data set patterns for multiple data set deletions.

Neither the IDCAMS host command environment nor the IDCAMS functions require TSO services.

REXX Compiler

The REXX compiler, RXTCOMP, no longer requires TSO services. The RXTCL procedure has been updated to reflect this change. In addition, the compiler will now derive a CSECT name using the member name or next-to-last sequential data set qualifier. Because of this, the NAME parameter is no longer required (but is still accepted).

MVS Services

Improvements have been made to the following functions:

RACROUTE A new argument has been added to suppress "ICH" messages. These messages are issued whenever an authorization check fails.

ENQ More accurate reporting of return and reason codes.

DEQ More accurate reporting of return and reason codes.

General REXX Extensions

A new pattern matching function, **MATCH**, has been added. This function supports matching using wildcard characters.

Static SQL

The following improvements have been made:

1. A static SQL program's data area can now be as large as 12K bytes.
2. Null output variables are now REXX dropped (unassigned) as in Dynamic DB2/SQL Services.
3. High-level assembler support has been added.
4. The static SQL preprocessor (RXTHPC) no longer requires TSO services. Skeletons and exec have been updated to reflect this change.

Architectural Improvements

The following improvements have been made:

1. All storage is now allocated from a single subpool. The subpool number can be customized for special environments.

2. The product ESTAE exit has been improved to better report ABEND reason codes.

3. Temporary authorization codes: A single message is issued -- once for each task using REXXTOOLS -- that displays the number of days remaining until expiration. The messages begin appearing 10 days prior to the expiration date.

Documentation

REXXTOOLS documentation has been reorganized into 4 publications:

1. *REXXTOOLS/MVS General Information*

2. *REXXTOOLS/MVS Installation Guide*

3. *REXXTOOLS/MVS Basic Services User's Guide*

4. *REXXTOOLS/MVS DB2/SQL Services User's Guide*

What's New in Version 4 Release 1 Mod 2

Released August 1995

Basic Services

VSAM

The following improvements have been made:

1. Support for Variable-length Relative record Data Sets (VRDS) is included in this release. Code paths have been shortened and other internal improvements have been made, resulting in a 55% reduction in the number of service units consumed during sequential processing of VSAM data sets.

2. New RPL options have been added to control the creation of $RXT variables. The NOV option specifies that $RXT variables are not to be created. The VAR option (the default) specifies that $RXT variables are to be created.

3. New ACB options DDN and DSN have been added to give the user explicit control over the way subtask sharing of VSAM control blocks and buffers is to be conducted. The default, DDN, is consistent with previous releases of REXXTOOLS.

4. The OPEN function will now tolerate multiple OPENs for the same ddname. The CLOSE function will tolerate multiple CLOSEs for the same ddname. In both cases, a return code value of 0 and a reason code value of 1 are returned.

5. The VSAM samples have been completely rewritten to better demonstrate how records can be parsed and formatted using REXX and REXXTOOLS facilities, and to demonstrate the use of VSAM in multi-user and batch environments.

Dynamic Allocation

The following improvements have been made:

1. A new dynamic allocation interface for compiled languages is provided. The module, RXTDAIR, is used to allocate and free data sets from within COBOL, PL/I, and assembler language programs.

2. The ALLOCATE command now supports BLKSIZE(0). This permits SMS determination of the appropriate block size.

General REXX Extensions

Several new functions have been added:

RXTTERM gives the REXX programmer explicit control over the termination of the REXXTOOLS environment. While the use of this function is not generally recommended, it may be necessary in specialized address spaces where automatic environment clean-up is not available.

D2F converts REXX decimal numbers to binary floating point numbers.

F2D converts binary floating point numbers to REXX decimal numbers.

D2PIC converts REXX decimal numbers to COBOL-style numeric picture strings. Full support for floating currency symbols, and other types of editing is available.

PIC2D converts numbers containing currency symbols and other editing characters to REXX decimal numbers.

The following functions have been modified:

STEMSORT has been modified to support full-length ascending and descending sorts with blank padding.

D2P and P2D have a new optional argument that allows the user to specify a value to be returned in the event of a syntax error. This argument makes programming for null or invalid data much simpler.

Dynamic SQL

The following improvements have been made:

1. New keywords have been added to the OPTIONS statement to permit the dynamic modification of DB2 subsystem and plan name defaults. These changes, which remain in effect for the life of the current task, remove the need for an explicit DSNALI OPEN call.

2. A new function, RXTDBSQL, has been added. This function provides a stand-alone function call interface to the dynamic SQL feature. RXTDBSQL may be invoked under alternative REXX interpreters, provided that these interpreters use the same calling conventions as REXX/MVS (e.g., OPS/REXX).

3. New DB2 samples have been added to the sample library, which demonstrate the use of REXXTOOLS in ISPF applications.

4. New samples have been added which demonstrate the use of REXXTOOLS in TCP/IP-based client/server applications. TCPLNT is an OS/2-based REXX client program that sends SQL requests to TCPSERV, an MVS/REXX server program. TCPSERV uses the

REXXTOOLS dynamic SQL interface to execute the requests and format replies. TCPSERVJ provides JCL for running TCPSERV.

5. Documentation for the "compatibility" interface has been moved to an appendix.

Static SQL

The following improvements have been made:

1. The RXTI Defaults panel has been changed to permit the specification of the DB2 load library.

2. The Preprocess panel has been changed to permit the explicit specification of the output EXEC and ASM member names.

Architectural Improvements

As a result of customer feedback, two changes have been made to make installation easier than ever:

1. Authorization codes have been greatly simplified yet expanded. Now only one authorization code (temporary or permanent) is required per CPU. Each authorization code indicates what features are licensed for a particular CPU. And, a single copy of REXXTOOLS/MVS can be authorized for use on up to 20 CPUs.

   In addition, to reduce the chance of transcription errors, authorization codes have been shortened from 32 to 24 printable hexadecimal digits.

2. Two new programs have been added that permit the use of REXXTOOLS from TSO, ISPF, and batch without permanent installation of the REXXTOOLS load library:

   RXTEXEC a TSO command for running REXX programs that use REXXTOOLS facilities. RXTEXEC can be used from TSO READY mode, and from within ISPF. It does not require a STEPLIB or ISPLLIB installation of the product.

   RXTJCL a program for running batch REXX programs.

   These programs also permit authorization codes to be dynamically specified (i.e., use of AMASPZAP is no longer mandatory).

What's New in Version 4 Release 1

Released July 1994

Basic Services

VSAM

The following improvements have been made:

1. Support for Batch Local Shared Resource (BLSR) subsystem is included in this release. Using BLSR, programs that randomly access VSAM data sets can gain significant performance improvements.

2. Support for TYPE=T (temporary) CLOSEs of VSAM data sets has been added. A TYPE=T CLOSE flushes buffers and updates catalog information without disconnecting the program from the data set.

Dynamic Allocation

ALLOCATE and FREE commands have been added to the REXXTOOL host command environment. These commands, which are modeled after the TSO/E commands of the same names, allow batch and APPC/MVS execs to dynamically allocate and free data sets without using the computationally expensive TSO Terminal Monitor Program (IKJEFT01).

REXX Compiler

New support for inter-language communication has been added to the interpretive compiler. Using an alternative prefix module, RXTRXPRC, compiled execs can now pass data back to their callers. Programs written in COBOL II, PL/I or assembler can use this facility.

Dynamic SQL

The following improvements have been included in this release:

1. The algorithm for PREPARE avoidance has been slightly modified to gain an additional 10 to 15 percent reduction in time spent in DB2 processing.

2. All return codes have messages associated with them. By default, messages will appear if a non-zero return code is produced. Message printing can be turned off or on using the NOMSGS or MSGS keywords of the OPTIONS command (also new with this release).

3. Complete support for DB2 3.1 parallelism and Remote Unit of Work has been added. The new statements that support these features are RELEASE, SET CONNECTION, and SET CURRENT DEGREE.

4. A new sample exec, SP2RX, has been added. SP2RX converts SPUFI-style SQL statements into a format suitable for inclusion in REXX programs. This utility makes it easier to port SPUFI-tested SQL statements to your REXX programs.

Static SQL

The following improvements have been included in this release:

1. The Static SQL feature now supports multi-row SELECT statements, making it easier to transition programs from dynamic to static SQL. The user may selectively choose, on a program-wide basis, whether SELECTs are to be interpreted as a single-row or multi-row.

2. The RXTI interactive application has been improved to provide greater flexibility when preprocessing REXX programs. Options have been provided that permit the user to customize bind processing and to specify whether the preprocessed exec is to be compiled.

What's New in Version 2 Release 1 Mod 2

Released March 1994

Basic Services

General REXX Extensions

The following improvements have been made:

1. A new host command environment (ADDRESS REXXTOOL) and three new host commands have been added to permit the sharing of data between REXX source programs. The new host commands are:

   VPUT A host command for placing a value in a global variable.

   VGET A host command for retrieving a value from a global variable.

   VERASE A host command for deleting a global variable.

2. The D2P function has been improved to accept precision and scale arguments. Precision and scale may be used instead of byte length to determine the size of the packed number.

Dynamic SQL

A new dynamic SQL implementation has been added. The new interface permits programs to be coded that will run in both the dynamic and static SQL environments. The new dynamic SQL interface supports DB2 Version 2 Release 3 features such as CONNECT, and contains numerous performance enhancements, including PREPARE statement avoidance.

The new dynamic SQL interface does not replace the previously released dynamic SQL interface. The older interface will continue to be supported.

Static SQL

A new static SQL implementation has been provided. The static SQL interface consists of a REXX precompiler (similar to IBM's DSNHPC program) and a run-time environment. The precompiler produces a modified REXX source program -- which can be compiled using the REXXTOOLS interpretive compiler -- and a Data Base Request Module (DBRM) that can be used to produce a static plan. The run-time environment provides REXX variable management and conversion services.

What's New in Version 2 Release 1

Released February 1993

Basic Services

VSAM

Improvements have been made to the VSAM functions which permit REXX programs to open an unlimited number of VSAM files. File (ddname) sharing among external REXX programs has also been greatly simplified. In particular, programs no longer are required to pass the $RXTFITB variable in order to share ddnames.

General REXX Extensions

The following functions have been added:

RXFUNC a function for dynamically maintaining (add, update, delete, query) REXX function packages. RXFUNC also pre-loads the load module associated with a function to enhance performance.

RXSUBCOM a function for dynamically maintaining host command environments. As with RXFUNC, RXSUBCOM pre-loads the load module associated with a function to enhance performance.

STORAGEX a function for retrieving and modifying virtual storage using the indirect addressing notation of the TSO TEST command.

REXX Compiler

The REXXTOOLS interpretive compiler has been improved to handle special-purpose parameter lists. Previously, a parameter list that was not recognized as one of the explicitly supported types was reported as an error. With Version 2 Release 1 of the compiler, unrecognized parameter lists cause a printable representation of the address contained in general purpose register 1 to be passed on to the ARG() function.

Dynamic SQL

A new and separately priced component for accessing DB2 has been added. Using the Dynamic SQL feature, REXX programs can perform queries (SELECT), create rows (INSERT), modify rows (UPDATE), and remove rows (DELETE). The interface supports a multi-row SELECT that permits access to an entire result set without the need to code a "fetch loop," as one would do in PL/I or COBOL. Alternatively, a REXX programmer can choose to explicitly code a fetch loop if the application's logic requires it.

The new Dynamic SQL support includes:

1. A new host command environment "Address SQL".
2. New functions:

   DB2INFO a function for extracting information related to DB2 default values, and the subsystem names of installed DB2 subsystems.

DSNALI a function for invoking the services of the DB2 Call Attachment Facility (CAF).

3. New samples have been added to the sample library which demonstrate the use of dynamic SQL.

Architectural Improvements

The following internal improvements have been made to the product:

1. REXXTOOLS no longer requires you to modify your REXX parameters modules. Specifically, the requirement for the REXX host command environment has been removed as has the requirement for the exec termination exit, RXTEXECT.

2. The product's function package has been renamed from RXTFLOC to the standard IRXFLOC name which is found in all IBM-supplied parameters modules. The source CSECT for the IRXFLOC module is provided so that you can add your own REXX functions.

What's New in Version 1 Release 2

Released February 1992

Basic Services

General REXX Extensions

The following functions have been added:

PARSETOK a function for parsing strings into tokens where the position and frequency of token delimiters is unpredictable.

QWIKREF an interface to Chicago-Soft's MVS/Quick-Ref database. You must be an MVS/Quick-Ref customer to use this function.

WORDSORT a function for sorting the blank-delimited words in a string. The words may be of any number or size, and can be sorted in either ascending or descending collating sequence.

REXX Compiler

A new program, RXTCOMP, has been added for batch compiling REXX programs into OS object modules. The compiler has the following interfaces:

RXC a TSO command for invoking the compiler.

RXTCL A JCL procedure for invoking the compiler and the system linkage-editor from batch jobs.