2018 RexxLA International Rexx Language Symposium Proceedings

René Vincent Jansen (ed.)

THE REXX LANGUAGE ASSOCIATION
RexxLA Symposium Proceedings Series
ISSN 1534-8954

Publication Data

©Copyright The Rexx Language Association, 2023

All original material in this publication is published under the Creative Commons - Share Alike 3.0 License as stated at `https://creativecommons.org/licenses/by-nc-sa/3.0/us/legalcode`.

A publication of **RexxLA Press**

The responsible publisher of this edition is identified as *IBizz IT Services and Consultancy*, Amsteldijk 14, 1074 HR Amsterdam, a registered company governed by the laws of the Kingdom of The Netherlands.

The RexxLA Symposium Series is registered under ISSN 1534-8954
The 2018 edition is registered under ISBN 978-94-648-5716-0

ISSN 1534-8954

9 771534 895004 >

ISBN 978-94-648-5716-0

9 789464 857160 >

2023-05-31 First printing

Introduction

History of the International Rexx Language Symposium

In 1990, Cathie Dager of SLAC[1] convened the organizing committee for the first independent Rexx[2] Symposium for Developers and Users. SLAC continued to organize this annual event until the middle of the 1990's when the RexxLA took over that responsibility. Symposia have been held annually since 1990.

About RexxLA

During the 1993 Symposium in La Jolla, California, plans for a Rexx User Group materialized. The Rexx Language Association (RexxLA), as it was called, is an independent, non-profit organization dedicated to promoting the use and understanding of the Rexx programming language. RexxLA manages several open source implementations of Rexx.

The selection procedure

Presentation proposals are solicited yearly using a CFP[3] procedure, after which the RexxLA symposium comittee reviews them and votes which presentations are selected for the symposium. The presentations are peer reviewed before being presented. Presenters are not compensated for their presentations.

Location

The 2018 symposium was held in Aruba, Dutch West Indies from 25 Mar 2018 to 29 Mar 2018.

[1] Stanford Linear Accelerator Center, since 2008 SLAC National Accelerator Laboratory
[2] Cowlishaw, M. F., **The REXX Language** (second edition), ISBN 0-13-780651-5, Prentice-Hall, 1990.
[3] Call For Papers.

Contents

1 NetRexx 3.07 New Features – René Vincent Jansen **1**

2 Integrating NetRexx code in ooRexx 5.0 – Rony G. Flatscher **12**

3 IBM Update: Ask the development team – Virgil Hein **19**

4 Rexx Tutorial for Beginners – Rony G. Flatscher **69**

5 The New BSF4ooRexx 6.00 – Rony G. Flatscher **100**

6 ooRexxDoc 5.0 – Alexander Seik **123**

7 Classic Rexx on MUSIC/SP – René Vincent Jansen **129**

8 Open Object Rexx Tutorial – Rony G. Flatscher **142**

9 Redirecting I/O for Commands to an External Environment – Gil Barmwater **164**

10 Anatomy of a GUI (Graphical User Interface) Application for Rexx Programmers – Rony G. Flatscher **180**

11 NetRexx Graphical User Interfaces with Pivot – Jason Martin **235**

1

NetRexx 3.07 New Features – René Vincent Jansen

Date and Time

26 Mar 2018, 14:00:00 CET

Presenter

René Vincent Jansen

Presenter Details

René has used Rexx since it appeared in TSO Extensions in the second half of the eighties when he was a systems programmer at the Central Bank of The Netherlands. He is an independent consultant since the turn of the century, specializing in models and meta models in order to rationalize data governance and model driven development. He likes to program in any language as long as it's Rexx.

NetRexx 3.07 New Features

René Vincent Jansen
29th International Rexx Language Symposium 2018
Aruba, Dutch West Indies

Agenda

```
pipe (testflight2)

literal * from FlightRoute where flight = 'KLM765' ! sqlselect ! console
```

Pipelines: SQLSelect Stage

Pipe the sql statement into it

From 3.07

```
fun main(args: Array<String>) {
  File("input.txt").forEachLine { handleLine(Rexx(it)) }

  // empty Rexx constructor
  val foo = Rexx()
  var bar:  Rexx
  bar = Rexx("test")
  RexxIO.Say(bar.reverse())
  RexxIO.Say(bar.hashCode())

  Val one: Int = 1
  val two: Int = 2
  RexxIO.Say(one + two)

}

fun handleLine(inp: Rexx) {
  var bar = Rexx(inp)
```

Rexx() Constructor Unshared

Make it usable from Kotlin

From 3.07

```
  */
  method ZzFacilityReport()
    return

  method main(args=String[]) static
    z = ZzFacilityReport()
    RexxIO.setOutputStream(FileOutputStream("fiscfac.csv"))
    say 'Contract Partner | Validity Start of Registration | Validity End of Registration | Date
from | Date To | Reporting Period From | Fiscal Facility'
    RexxIO().File("data/erp_wb_dpsob_bp_acc.txt").forEachline(z.file1())
    RexxIO().File("data/pr1_zzfacility.txt").forEachline(z.file2())


  class ZzFacilityReport.file1 dependent implements LineHandler
  method handle(in)
    parse in '|','|bp'|' .
    parent.bpSet.add(bp.strip)

  class ZzFacilityReport.file2 dependent implements LineHandler
  method handle(in)
    z = ZzFacility()
    z.parse(in)
    if z.getPARTNER = '' then return
    if parent.bpSet.contains(z.getPARTNER.strip) then return
    if z.getPARTNER = 'PARTNER' then return
```

RexxIO Runtime Improvements

Set/Push/PopOutputStream

From 3.07

Annotations

From 3.06

```
options binary
@Author(name="Class Author")
class AnnotateTest

properties private unused
propz
a = ArrayList()
test = TreeMap()


  @SuppressWarnings("unchecked")
  method main(args=String[]) static
    say 'hello annotations'
    t=AnnotateTest()
    t.old()

    @Override
    @Deprecated /* just to illustrate a comment */
  method toString() returns String
    return 'Annotations'

    @Deprecated
  method old() /* a comment with an @ in it */
```

OSProcess()

From 3.06

Runtime support for ADDRESS()

```
/**
 * Method cmp compares two binary files
 * @param sha1 is a ObjectId
 * @param sha2 is a ObjectId
 */
method cmp(sha1=ObjectId,sha2=ObjectId) protect
  retrieveFileFromSHA(sha1,'tmpf1')
  retrieveFileFromSHA(sha2,'tmpf2')

  command    = ArrayList()
  command.add('cmp')
  command.add('tmpf1')
  command.add('tmpf2')
  os = OSProcess()
  a  = os.outtrap(command)
  i  = a.iterator()
  loop while i.hasNext()
    line     = Rexx i.next()
    say line
  end
```

Rexx.soundex()

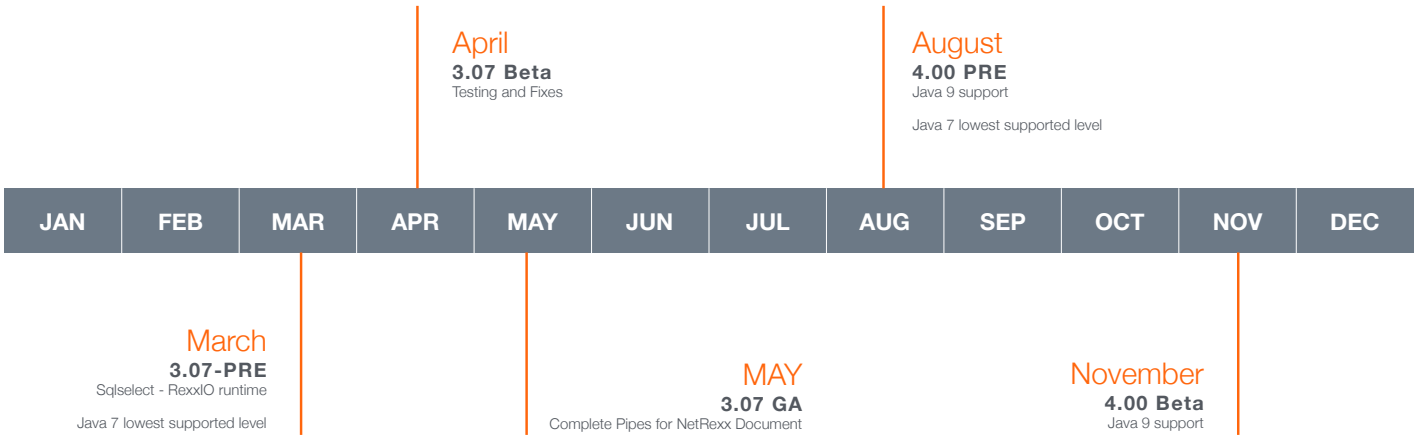
From 3.07

Method soundex() for Rexx strings

```
barre      B600 = B600
Wheaton    W350 = W350
Knuth      K530 = K530
auerbach   A612 = A612
Ekzampul   E251 = E251
D-day      D000 = D000
example    E251 = E251
4-H        H000 = H000
Burroughs  B620 = B620
d jay      D200 = D200
F.B.I.     F000 = F000
Lissajous  L222 = L222
```

Agenda

Release Timeline for NetRexx 3.0x - 4.00



SQLSelect stage of pipelines

- One of the first NetRexx programs I wrote
- It only accepted input from its commandline input
- It needed to accept input from a previous stage in the pipeline
- It nows does, after some 20 years

- This also prompted some experimentation with SQLite
- Which works wonderful with NetRexx

```
jdbcdriver=org.sqlite.JDBC
url=jdbc:sqlite:flightroute-iata.sqb
```

```
pipe (testflight2)

literal * from FlightRoute where flight = 'KLM765' ! sqlselect ! console
```


Rexx() constructor unshared

- Admittedly, this is not really useful for NetRexx but makes for a much better first impression when using the Rexx class in **Kotlin**
- Kotlin: upcoming, en-vogue language
  - It has a lot of the good things we know in NetRexx
  - Needs more investigation,
    - at least the first thing you try does not fail
  - If you are hired for a Kotlin project: yes, you can use Rexx
    - All the string functions we know from the 1980's there

```
fun main(args: Array<String>) {
  File("input.txt").forEachLine { handleLine(Rexx(it)) }

  // empty Rexx constructor
  val foo = Rexx()
  var bar:  Rexx
  bar = Rexx("test")
  RexxIO.Say(bar.reverse())
  RexxIO.Say(bar.hashCode())

  Val one: Int = 1
  val two: Int = 2
  RexxIO.Say(one + two)

}

fun handleLine(inp: Rexx) {
  var bar = Rexx(inp)
```

This one I liked in Kotlin

- Open a file with its name and specify in on line how and where to handle each record
  - It tempted me to do some work (at work) in Kotlin
  - Until I realised we can to this in about the same manner in NetRexx

```
fun main(args: Array<String>) {
  File("input.txt").forEachLine { handleLine(Rexx(it)) }

  // empty Rexx constructor
  val foo = Rexx()
  var bar:  Rexx
  bar = Rexx("test")
  RexxIO.Say(bar.reverse())
  RexxIO.Say(bar.hashCode())

  Val one: Int = 1
  val two: Int = 2
  RexxIO.Say(one + two)

}

fun handleLine(inp: Rexx) {
  var bar = Rexx(inp)
```

```
class ZzFacilityReport
  properties inheritable
  bpSet = TreeSet()

  /**
   * Default constructor
   */
  method ZzFacilityReport()
    return

  method main(args=String[]) static
    z = ZzFacilityReport()

    RexxIO.setOutputStream(FileOutputStream('fiscfac.csv'))

    RexxIO().File('data/erp_wb_dpsob_bp_acc.txt').forEachline(z.file1())
    RexxIO().File('data/pr1_zzfacility.txt').forEachline(z.file2())



  class ZzFacilityReport.file1 dependent implements LineHandler
  method handle(in)
    parse in '|'.'|'bp'|' .
    parent.bpSet.add(bp.strip)

  class ZzFacilityReport.file2 dependent implements LineHandler
  method handle(in)
    z = ZzFacility()
    z.parse(in)
    if z.getPARTNER = '' then return
    if parent.bpSet.contains(z.getPARTNER.strip) then return
    if z.getPARTNER = 'PARTNER' then return
    if z.getZZACTVTSTART = '' then z.setZZACTVTSTART('01.01.1900')
```

Oneliner file handler

Using a minor class and inheritable properties

Support for this in RexxIO runtime class

- Previously not documented, contains Say(), Ask(), AskOne()
- Method file()
  - Accepts a filename and constructs a BufferedReader
  - Returns RexxIO (static) to be able to chain methods
- Method forEachLine()
  - accepts any implementation of the LineHandler interface

```
package netrexx.lang

class LineHandler interface
  method handle(in=Rexx)
```

```
method File(nm) returns RexxIO
  do
    fileIn = BufferedReader(FileReader(nm))
  catch IOException
    return null
  end
  return this

method forEachline(c=LineHandler)
  do
    loop forever
      line = Rexx fileIn.readLine()
      if line = null then leave
      c.handle(line)
    end
  catch IOException
  end -- do
```


Other RexxIO changes: OutputStream

- I noticed how everything that is prototyped with **say** always ends up needing to be written to a file
  - We can redirect, but that means all System.out and System.err ends up in between the output
  - We can open a PrintWriter and change all say statements to println()
  - Opening a file in a number of lines and changing all say statements is drudge work
  - How about if we could just **say** something (in)to a file
  - Thats is what the experiment is about

setOutputStream

- You can set an OutputStream on the RexxIO class (which is static)
  - For the first time, you can switch between stdout and stderr
  - You may also specify a FileOutputStream
  - All **say** output from that moment on will go to that file
  - Reset it by setting it back to System.out
  - Every **say** always flushes the output stream (and always did)
  - Even when this is taken into account:
    - On systems with slow consoles (read: windows):
      - The speedup is stunning when writing to a file

```
class ZzFacilityReport
  properties inheritable
  bpSet = TreeSet()

  /**
   * Default constructor
   */
  method ZzFacilityReport()
    return

  method main(args=String[]) static
    z = ZzFacilityReport()

    RexxIO.setOutputStream(FileOutputStream('fiscfac.csv'))

    RexxIO().File('data/erp_wb_dpsob_bp_acc.txt').forEachline(z.file1())
    RexxIO().File('data/pr1_zzfacility.txt').forEachline(z.file2())


  class ZzFacilityReport.file1 dependent implements LineHandler
  method handle(in)
    parse in '|'.'|'bp'|' .
    parent.bpSet.add(bp.strip)

  class ZzFacilityReport.file2 dependent implements LineHandler
  method handle(in)
    z = ZzFacility()
    z.parse(in)
    if z.getPARTNER = '' then return
    if parent.bpSet.contains(z.getPARTNER.strip) then return
    if z.getPARTNER = 'PARTNER' then return
    if z.getZZACTVTSTART = '' then z.setZZACTVTSTART('01.01.1900')
```


What if we want some **say** output going to more outputstreams?

- To make **say** output go to more streams (stdout, a file, stderr) we can:
  - pushOutputStream
    - Add one outputstream
  - popOutputStream
    - Remove the latest added outputstream
  - StdOut in RexxIO is now a ConcurrentLinkedDeque
    - Which should make it reasonable thread safe

```
method setOutputStream(out=OutputStream) static
    StdOut.clear()
    StdOut.push(PrintWriter(out))

method pushOutputStream(out=OutputStream) static
    StdOut.push(PrintWriter(out))

method popOutputStream() static
  do
    StdOut.pop()
  catch java.util.NoSuchElementException
    StdOut.push(PrintWriter(System.out))
  end
```

Annotations (in 3.06)

- Adding annotations was not avoidable due to the large amount of Java classes using mandatory annotations - jUnit, vaadin, Jakarta Spring
- Unlike generics, the way to handle these in NetRexx without language support would be much more complex (though not impossible, everything becomes a method call in the end)
- For this reason, the parser was adapted to recognise and pass through @annotations
- This was not easy and there still are some snags
- Most of the things you need do work, though

```
options binary
@Author(name="Class Author")
class AnnotateTest

properties private unused
propz
a = ArrayList()
test = TreeMap()


  @SuppressWarnings("unchecked")
  method main(args=String[]) static
    say 'hello annotations'
    t=AnnotateTest()
    t.old()

    @Override
    @Deprecated /* just to illustrate a comment */
  method toString() returns String
    return 'Annotations'

    @Deprecated
  method old() /* a comment with an @ in it */
    say 'do no use anymore'
```


OSProcess - Runtime support for ADDRESS and OUTTRAP (since 3.06)

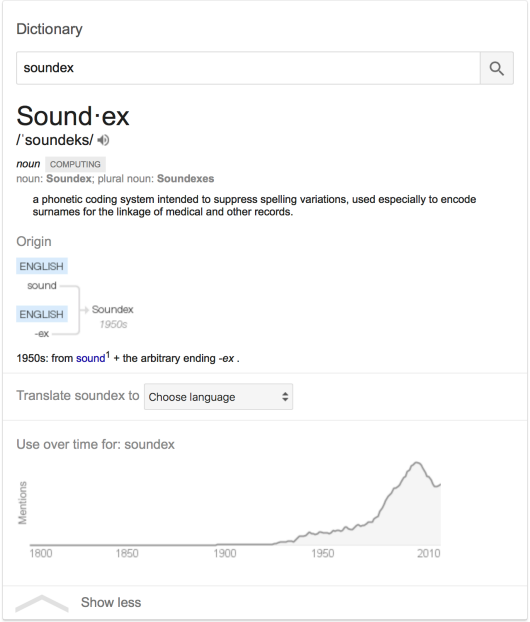
- NetRexx was designed with the following assumptions
  - Java is going to be used for I/O
  - Java interfaces are going to be used for native functionality
  - Java handles pretty much everything and native is not needed
  - Here NetRexx diverges from other dialects
  - Scripting is closely related to the (OS/Platform) environment
  - These can be building blocks for an ADDRESS command
  - Let's see what ooRexx is doing with ADDRESS WITH

```
 /**
  * Method cmp compares two binary files
  * @param sha1 is a ObjectId
  * @param sha2 is a ObjectId
  */
 method cmp(sha1=ObjectId,sha2=ObjectId) protect
   retrieveFileFromSHA(sha1,'tmpf1')
   retrieveFileFromSHA(sha2,'tmpf2')

   command    = ArrayList()
   command.add('cmp')
   command.add('tmpf1')
   command.add('tmpf2')
   os = OSProcess()
   a  = os.outtrap(command)
   i  = a.iterator()
   loop while i.hasNext()
     line     = Rexx i.next()
     say line
   end
```

Soundex (3.07)

- Rexx variables have to ways for comparison
  - A strict (==) comparator
  - A less strict (more what a human would do) comparator (=)
  - But it misses a loose comparator
  - For this, the Soundex algorithm is the standard
  - For data cleansing operations this was needed so often, it was put as a method on the Rexx string
- Why put it in the runtime
  - the algorithm is just not trivial enough to assume that language users will easily roll their own
  - It is a good addition to the other two comparators



Soundex example & testset

- We need to normalize a database that has a free field for street name
- We know people have put in various forms of ‘unknown’
- We know that **‘unknown’.soundex()** is U525
- We now find:

Unkown/ Onbekend
Unknown\ Onbekend
Unknown/Onbekend
Unknown/Onbeken
Unknown/ Onbekend
Unknown Onbekend
UNKNOWN /ONBEKND
Unknown /Onbekend
Unknown / Onbekend
unknown /onbekend
Unknown
Unknowm/ Onbekend
Unknnown/Onbekend

```
barre      B600 = B600
Wheaton    W350 = W350
Knuth      K530 = K530
auerbach   A612 = A612
Ekzampul   E251 = E251
D-day      D000 = D000
example    E251 = E251
4-H        H000 = H000
Burroughs  B620 = B620
d jay      D200 = D200
F.B.I.     F000 = F000
Lissajous  L222 = L222
Burrows    B620 = B620
coöp       C100 = C100
de la Rosa D462 = D462
Gauss      G200 = G200
Donnell    D540 = D540
Ghosh      G200 = G200
Dracula    D624 = D624
Ellery     E460 = E460
he         H000 = H000
Gutierrez  G362 = G362
Drakula    D624 = D624
Williams   W452 = W452
Heilbronn  H416 = H416
Du Pont    D153 = D153
Robert     R163 = R163
Pfister    P236 = P236
Moskowitz  M232 = M232
Euler      E460 = E460
Hilbert    H416 = H416
Rupert     R163 = R163
Uhrbach    U612 = U612
Moskovitz  M213 = M213
Lukasiewic L222 = L222
Woolcock   W422 = W422
Tymczak    T522 = T522
Rubin      R150 = R150
Swhgler    S460 = S460
```

Soundex implementation

- Somewhat dependent on language
- The canonical form is for English
- The numbers are dependent on pronunciation
- In case of popular demand:
  - We need to make these strings swappable

```
/** soundex returns the normalized soundex value of the string */
method soundex() returns Rexx
  in = this.upper()
  old_alphabet= 'AEIOUYHWBFPVCGJKQSXZDTLMNR'
  new_alphabet= '@@@@@@**111122222222334556'
  word=Rexx('')
  loop i=1 for intlength()
    tmp_=in.substr(i, 1)
    if tmp_.datatype('M')  then word=word||tmp_
  end
  value=word.strip().left(1)
  word=word.translate(new_alphabet, old_alphabet)
  prev=value.translate(new_alphabet, old_alphabet)
  loop j=2  to word.length()
    q=word.substr(j, 1)
    if q\==prev & q.datatype('W')  then do
      value=value || q;  prev=q
    end
    else if q=='@'  then prev=q
  end
  return value.left(4,0)
```


NetRexx 4.00

- NetRexx 3.X does not run on Java 9
- This is due to an incompatible change by Java - the Oracle team
- Reason for the change is the module system
- NetRexx reads all jars and zip, and classes on the classpath for every compilation
  - This has become impossible now
- Later this week we will have a workshop on reflection and method handles
- Results of this workshop will be highly important to the future of NetRexx

Thank you for your attention

- Q? rvjansen@xs4all.nl or president@rexxla.org

29th International Rexx Language Symposium

Aruba, Dutch West Indies

2

Integrating NetRexx code in ooRexx 5.0 – Rony G. Flatscher

Date and Time

26 Mar 2018, 15:00:00 CET

Presenter

Rony G. Flatscher

Presenter Details

Rony works as a professor for Business informatics ("Wirtschaftsinformatik") at the Vienna University of Economics and Business Administration (Wirtschaftsuniversität Wien) and uses Open Object Rexx for teaching Business Administration and MIS students the object-oriented paradigm, as well as remote-controlling (automating) Windows and Windows end-user applications (e.g. MS Office, Open Office) as well as Java and Java applications (he is the author of BSF4ooRexx, the ooRexx-Java bridge, which uses Apache BSF and had Rony invited to become an ASF member). He consults and trains in all of his research fields.

Integrating NetRexx Code in ooRexx 5.0

Rony G. Flatscher, WU
2018 International Rexx Symposium

Overview

- NetRexx integration with BSF4ooRexx
  - Example
- ooRexx 5.0
- Integrating NetRexx into ooRexx 5.0
  - Example
- Roundup and outlook

NetRexx Integration with BSF4ooRexx, 1

- BSF4ooRexx
  - If NetRexx present, then it automatically turns NetRexx objects into Rexx strings
  - ooRexx can be used by NetRexx
    - Apache's BSF framework
    - Java's javax.script framework
    - See samples coming with BSF4ooRexx!
  - Each NetRexx class can be used by ooRexx
    - Apache's BSF framework: must be precompiled!
    - Java's javax.script framework: must supply source

NetRexx Integration with BSF4ooRexx, 2

- Example 1
  - NetRexx source stored in a file
  - JSR-223 (javax.script) to fetch NetRexx engine
  - Getting default ScriptContext
    - Adding an entry "javax.script.filename" according to the JSR-223 specifications
  - Using NetRexx engine to put an attribute into its default ScriptContext
    - Attribute "hi" gets a string from ooRexx
  - NetRexx engine is used to run the NetRexx program which fetches and shows the entries

NetRexx Integration with BSF4ooRexx, 3

```
manager = .bsf~new("javax.script.ScriptEngineManager")
nre=manager~getEngineByName("NetRexx")  -- fetch NetRexx via JSR-223

signal on syntax
sc=nre~getContext                -- get its default ScriptContext
filename="someNetRexxCode.nrx"
sc~setAttribute(nre~FILENAME, filename, sc~ENGINE_SCOPE)
nre~put("sc", sc)
nre~put("hi", "hello at" .dateTime~new)
say "(ooRexx) executing ["filename"] ..."
res=nre~eval(.bsf~new("java.io.FileReader", filename), sc)
exit

syntax:
   co=condition(o)
   say ppCondition2(co)
   exit -1

::requires BSF.CLS          -- get Java bridge
::requires "rgf_util2.rex"  -- get access to ppCondition2()
```


NetRexx Integration with BSF4ooRexx, 4

```
parse source s
say '(nrx) this is NetRexx speaking from:' s
say '(nrx) this is what got sent to me:  ' hi
name="javax.script.filename"
say '(nrx)' name':' sc.getAttribute(name)
```

- Output:

```
(ooRexx) executing [someNetRexxCode.nrx] ...
(nrx) this is NetRexx speaking from: Java method someNetRexxCode
(nrx) this is what got sent to me:   hello at 2018-03-25T17:04:47.514436
(nrx) javax.script.filename: someNetRexxCode.nrx
```

ooRexx 5.0

- Many new and useful features!
- ::RESOURCE directive
  - Allows for storing any kind of text
    - Possible to store code, e.g. NetRexx source code!
  - .resources serves as the directory of resources
  - Fetching a resource entry returns an array of strings
  - Using ooRexx Array's toString method returns a plain string

Integrating NetRexx Into ooRexx 5.0, 1

- Use a ::resource directive instead of a file
- Allows one to have all resources in one package, the ooRexx program
- Will use JSR-223 to run the NetRexx program

Integrating NetRexx Into ooRexx 5.0, 2

```
manager = .bsf~new("javax.script.ScriptEngineManager")
nre=manager~getEngineByName("NetRexx")  -- fetch NetRexx via JSR-223

signal on syntax
sc=nre~getContext                -- get its default ScriptContext
filename="someNetRexxCode.nrx (really, it is a string)"
sc~setAttribute(nre~FILENAME, filename, sc~ENGINE_SCOPE)
nre~put("sc", sc)
nre~put("hi", "hello at" .dateTime~new)
say "(ooRexx) executing ["filename"] ..."
code=.resources~netRexxCode~toString
res=nre~eval(code, sc)
exit

syntax:
   co=condition(o)
   say ppCondition2(co)
   exit -1

::requires BSF.CLS          -- get Java bridge
::requires "rgf_util2.rex"  -- get access to ppCondition2()

::resource netRexxCode
    parse source s
    say '(nrx) this is NetRexx speaking from:' s
    say '(nrx) this is what got sent to me:  ' hi
    name="javax.script.filename"
    say '(nrx)' name':' sc.getAttribute(name)
::END
```


Integrating NetRexx Into ooRexx 5.0, 3

```
::resource netRexxCode
    parse source s
    say '(nrx) this is NetRexx speaking from:' s
    say '(nrx) this is what got sent to me:  ' hi
    name="javax.script.filename"
    say '(nrx)' name':' sc.getAttribute(name)
::END
```

- Output:

```
(ooRexx) executing [someNetRexxCode.nrx (really, it is a string)] ...
(nrx) this is NetRexx speaking from: Java method someNetRexxCode
(nrx) this is what got sent to me:   hello at 2018-03-25T17:34:31.229503
(nrx) javax.script.filename: someNetRexxCode.nrx (really, it is a string)
```

Roundup and Outlook

- BSF4ooRexx allows
  - Full access to compiled NetRexx programs/classes
  - Full access to JSR-223
    - ooRexx programs can run NetRexx programs from source using BSF4ooRexx
  - ooRexx 5.0 programs can embed the source
- Outlook
  - If possible to compile and access NetRexx classes
    - ooRexx programs can use NetRexx e.g. for lambdas and any unforeseen need to use NetRexx classes on the fly!

3

IBM Update: Ask the development team – Virgil Hein

Date and Time

26 Mar 2018, 18:30:00 CET

Presenter

Virgil Hein

Presenter Details

Virgil has been with IBM for 38+ years working in software development. In his current position as an IBM Business Manager he is responsible for all facets of a set of mature technology products. This includes responsibility for strategy, business management, development, marketing, sales, service, and support. Main products include Office Vision products, BookManager, Rexx, and OS/2. In this position the main goals are focused on maintaining/increasing customer satisfaction, supporting customer efforts to migrate to follow-on solutions, and finding creative means of increasing mature/growth product revenue. As the product owner for the IBM Rexx Compiler, Virgil is closely involved with a variety of Rexx activities both inside and outside of IBM.

IBM Rexx Language Update: Classic Rexx and The Rexx Compiler

Virgil Hein IBM
vhein@us.ibm.com

March 2018

Disclaimers

- The information contained in this presentation is provided for informational purposes only.
- While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided “as is”, without warranty of any kind, express or implied.
- In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice.
- IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other documentation.
- Nothing contained in this presentation is intended to, or shall have the effect of:
  - Creating any warranty or representation from IBM (or its affiliates or its or their suppliers and/or licensors); or
  - Altering the terms and conditions of the applicable license agreement governing the use of IBM software.

 © 2014, 2016 IBM Corporation

Agenda

- HLASM TextBook
- REXX products
- External environments and interfaces
- Key functions and instructions
- REXX compound variables vs. data stack
- I/O
- Troubleshooting
- Programming style and techniques
- REXX Enhancements (z/OS)

© 2014, 2016 IBM Corporation

HLASM TextBook

- HLASM TextBook V 1.00
  - Marist College web site:
    http://idcp.marist.edu/enterprisesystemseducation/Assembler%20Language%20Programming%20for%20IBM%20z%20System%20Servers.pdf
- HLASM TextBook V 2.00
  - PDF: Assembler Language Programming for IBM Systems Z Servers V2.00

Assemb Lang
Programming

© 2014, 2016 IBM Corporation