WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Department of Information Systems
and Operations Management

OoRexx 5.1.0 Release Update

The 2025 International Rexx Symposium
Vienna, Austria
May4th – May 7th 2025

Vienna University of Economics and Business ▪ Welthandelsplatz 1, D2-C ▪ A-1020 Vienna

Rony G. Flatscher

Overiew

- Release of ooRexx 5.1.0
- Changes and additions
- Focusing on the new TraceObject class
- Roundup

Release of ooRexx 5.1.0

- Released at the beginning of May 2025 to coincide with the International Rexx Symposium 2025
  - ooRexx 5.1.0 GA (general availability)
- Changes
  - 72 bugs fixed
  - 36 feature enhancements
  - 35 documentation bugs fixed
  - 1 patch incorporated
- Passes the comprehensive ooRexx test suite on all major operating systems the Jenkins build and test system provides

Some Important Changes, Additions, 1

- New TraceObject class to improve tracing
- .context now understands for improved debugging
  - InvocationID
  - InstanceID
  - ThreadID
- The StackFrame class now supplies its context object
- json.cls
  - Add class and instance methods to the JSON class to simplify usage
  - Allow for human friendly inspection of JSON data, e.g. sorting attributes, optionally inserting whitespace for better legibility

Some Important Changes, Additions, 2

- Improve samples/complex.cls and some more samples
- Added oleinfo scripts from rony/sandbox/oleinfo (Windows)
  - Useful utilities to learn about installed COM/OLE classes, allows to create html documentation on the fly
  - Placed into samples/ole/oleinfo

New TraceObject Class, Overview

- Classic Rexx tracing does not allow to gain insights into some of the execution context, e.g.
  - Rexx interpreter instance currently executing (there may be many running)
  - The thread ID used to run Rexx code
  - The invocation ID in case there are parallel invocations of the same code on different Rexx interpreter instances
  - The current stackframe and its information
- The TraceObject class is a subclass of the StringTable class
  - Allows for storing context information using string indexes
  - Each tracing will cause the creation of an instance of the TraceObject class and store all relevant context information with it
  - It is possible for ooRexx programmers to intercept them in real-time or have the generated trace objects collected

TRACE Keyword Instruction and Built-in Function (BIF)

- REXX (1979, IBM)
  - A dynamic and dynamically typed language
  - Three instruction types
    - Assignment (second token an equal sign =)
    - Keyword instruction (first word is a defined REXX keyword)
    - Command instruction (any other string)
  - Allows for tracing all instruction types
    - TRACE keyword instruction and built-in function (BIF) with possible options
      - Normal: traces failures in command instructions (on by default)
      - All: displays instructions before executing them
      - Results: displays the instruction's result, if any
      - Intermediates: displays all intermediate evaluations in an expression

REXX Instructions and TRACE Normal (Default)

```
a="hello, world"        -- an assignement instruction
say "a="a               -- a SAY keyword instruction
say "TRACE option in effect:" trace() -- 'N'='Normal'
say
'echo' a                -- a known system command
say "return code:" rc -- return code
say
address nope 'nixi' a -- a non-existing command environment
say "return code:" rc -- return code
```

Output:

```
a="hello, world"
TRACE option in effect: N

hello, world
return code: 0

     8 *-* address nope 'nixi' -- a non-existing command environment
       >>> "nixi hello, world"
       +++ "RC (30)"
return code: 30
```

"trace prefix"

TRACE Options All, Results, Intermediates

```
do opt over 'All', 'Results', 'Intermediates'
   say "setting TRACE to '"opt"':"
   currOpt=trace(opt) -- TRACE (function)
   a=100+random() -- line # 4
   say "a:" a      -- line # 5
   TRACE N         -- line # 6 (keyword)
   say "---"
end
```

Output:

```
setting TRACE to 'All':
     4 *-*   a=100+random() -- line # 4
     5 *-*   say "a:" a     -- line # 5
a: 1088
     6 *-*   TRACE N        -- line # 6 (keyword)
---
setting TRACE to 'Results':
     4 *-*   a=100+random() -- line # 4
       >>>     "436"
     5 *-*   say "a:" a     -- line # 5
       >>>     "a: 436"
a: 436
     6 *-*   TRACE N        -- line # 6 (keyword)
---
setting TRACE to 'Intermediates':
       >F>     TRACE => "N"
       >=>     CURROPT <= "N"
     4 *-*   a=100+random() -- line # 4
       >L>     "100"
       >F>     RANDOM => "514"
       >O>     "+" => "614"
       >>>     "614"
       >=>     A <= "614"
     5 *-*   say "a:" a     -- line # 5
       >L>     "a:"
       >V>     A => "614"
       >O>     " " => "a: 614"
       >>>     "a: 614"
a: 614
     6 *-*   TRACE N        -- line # 6 (keyword)
---
```


Rony G. Flatscher

ooRexx Program With Multithreading (TRACE All)

```
t1=.test~new        -- create an instance
t2=.test~new        -- create another instance
t1~hey
t2~ho
say "-"~copies(50)
t2~start('hey')
t1~start('ho')
say "-"~copies(50)

::class test        -- some Rexx class
::method hey        -- by default: guarded
  say 'hey (guarded)'

::method ho unguarded   -- unguarded
  say 'ho (unguarded)'

::options trace all
```

Output:

```
     1 *-* t1=.test~new      -- create an instance
     2 *-* t2=.test~new      -- create another instance
     3 *-* t1~hey
       >I> Method "HEY" with scope "TEST"
    12 *-* say 'hey (guarded)'
hey (guarded)
       <I< Method "HEY" with scope "TEST"
     4 *-* t2~ho
       >I> Method "HO" with scope "TEST"
    15 *-* say 'ho (unguarded)'
ho (unguarded)
       <I< Method "HO" with scope "TEST"
     5 *-* say "-"~copies(50)
--------------------------------------------------
     6 *-* t2~start('hey')
     7 *-* t1~start('ho')
     8 *-* say "-"~copies(50)
--------------------------------------------------
       >I> Method "HEY" with scope "TEST"
       >I> Method "HO" with scope "TEST"
    12 *-* say 'hey (guarded)'
hey (guarded)
    15 *-* say 'ho (unguarded)'
ho (unguarded)
       <I< Method "HEY" with scope "TEST"
       <I< Method "HO" with scope "TEST"
```

ooRexx and TRACE Information, 1

- ooRexx added new trace prefixes for new ooRexx features, e.g.
  - TRACE option All, Labels, Results
    - >I> (invocation entry)
    - <I< (invocation exit)
  - TRACE option Intermediates
    - >E> (name and value of an environment symbol)
    - >M> (name and result of a message)
    - >N> (name and result of a name-prefixed symbol)
    - >R> (name of argument and name of referenced variable)

Rony G. Flatscher

ooRexx and TRACE Information, 2

- Missing information about
  - Rexx interpreter instances
    - E.g. each JavaFX scene gets controlled by a separate ooRexx interpreter instance
  - Invocation ID
  - Thread ID
  - Attribute pool ID
    - Attributes and methods of the same class scope share the same instance attribute pool
    - Access of attribute pools is controlled by
      - Guard state of attributes and methods
      - Guard lock owner
      - Guard lock count (scope lock count)
- Overwhelming, not all information is always needed!

ooRexx Devising a TraceObject Class (ooRexx r12881), 1

- Subclassing StringTable (a map collection with a string index)

TraceObject
subclass StringTable

class attributes

+ collector = .nil
+ counter <<getter>>
- counter = 0
+ option = 'N'

class methods

+ activate
+ new: traceObject
- makeStringImpl(traceObject): String
+ setMakeString(makeStringMethod)
+ unsetMakeString

instance methods

+ compareTo: -1 | 0 | 1
+ makeString: String

Rony G. Flatscher

ooRexx Devising a TraceObject Class (ooRexx r12881), 2

- Class Attributes
  - collector, .nil (null) by default
    - If object is assigned it will get each created traceObject appended
      - Assigned object must understand the message append (any ordered collection is able to)
  - counter (getter) returns current count of trace objects created so far
  - option:
    - N ("normal", display traceline),
    - T ("thread", like N, but inject thread ID in trace prefix),
    - S ("standard", like N, but prepend extended trace information in brackets indicating the thread ID, the invocation ID, in case of a method in addition the attribute pool ID, the method's defined and current guard state, the lock count, guard lock reserved indicator and waiting state)
    - F ("full", like S, but include Rexx interpreter instance ID in addition)
    - P ("profiling", "probing", allow collecting trace objects, but do not display traceline)

ooRexx Devising a TraceObject Class (ooRexx r12881), 3

- Class Methods
  - makeStringImpl, default formatting of trace output, handles all option settings
  - setMakeString, allows to change the makeString method to use
  - unsetMakeString, reverts to makeStringImpl
- Instance Methods
  - makeString: renders traceObject to a string using the currently set class method (makeStringImpl or the method supplied to setMakeString)
    - ooRexx will request a string rendering from objects under certain circumstances, e.g. when using a SAY keyword instruction on an object, by sending it the makeString message

ooRexx Devising a TraceObject Class (ooRexx r12881), 4

- Allows adding any number of trace related information in the trace object, like
  - ATTRIBUTEPOOL (ID number, makeStringImpl prefixes a 'A')
  - CALLERSTACKFRAME (a StringTable with the caller's stackframe information)
  - HASSCOPELOCK (boolean, if true makeStringImpl displays '*', a blank else)
  - INTERPRETER (ID number, makeStringImpl prefixes a 'R')
  - INVOCATION (ID number, makeStringImpl prefixes a 'I')
  - ISGUARDED (boolean, true if method is defined to be guarded, then makeStringImpl displays a 'G', a blank else)
  - ISWAITING (boolean, if true makeStringImpl displays a 'W', a blank else)
  - NUMBER (sequence number)
  - OPTION (the option character at time of creation)

ooRexx Devising a TraceObject Class (ooRexx r12881), 5

- RECEIVER (the receiver object for which the method runs)
- SCOPELOCKCOUNT (number, the current guard lock count, makeStringImpl prefixes a 'L')
- STACKFRAME (a StringTable with the stackframe information)
- THREAD (ID number, makeStringImpl prefixes a 'T')
- TIMESTAMP (DateTime of object creation)
- TRACELINE (the formatted trace line)
- VARIABLE (a StringTable with the name, value and usage)

Rony G. Flatscher

Collecting Trace Objects and Displaying with Option Thread

```
arrTraceLog=.array~new  -- array for collecting
.TraceObject~collector=arrTraceLog
.TraceObject~option='Profile'
t=.test~new          -- create object
say t~show_m1        -- synchronous message
msg=t~start(show_m1) -- asynchronous message
say "msg~result:" msg~result  -- waits for result
say

.TraceObject~collector=.nil   -- stop appending
say ".TraceObject~counter:" .TraceObject~counter
say "arrTraceLog has" arrTraceLog~items "trace
objects:"
.TraceObject~option='Thread'  -- set format option
do traceObject over arrTraceLog
  say traceObject   -- string value by makeString
end

::class test
::method m1      unguarded
  trace results
  return 'hello, my beloved world!'

::method show_m1
  trace results
  reply self~m1    -- return and new thread
  say "about to return from show_m1 ..."
```

Output:

```
hello, my beloved world!
about to return from show_m1 ...
about to return from show_m1 ...
msg~result: hello, my beloved world!

.TraceObject~counter: 24
arrTraceLog has 24 trace objects:
      >I1> Method "SHOW_M1" with scope "TEST"
   25 *-1* reply self~m1   -- return and new thread
      >I1> Method "M1" with scope "TEST"
   21 *-1* return 'hello, my beloved world!'
      >>1>   "hello, my beloved world!"
      <I1< Method "M1" with scope "TEST"
      >>1>   "hello, my beloved world!"
      <I1< Method "SHOW_M1" with scope "TEST"
      >I2> Method "SHOW_M1" with scope "TEST"
   26 *-2* say "about to return from show_m1 ..."
      >>2>   "about to return from show_m1 ..."
      <I2< Method "SHOW_M1" with scope "TEST"
      >I3> Method "SHOW_M1" with scope "TEST"
   25 *-3* reply self~m1   -- return and new thread
      >I3> Method "M1" with scope "TEST"
   21 *-3* return 'hello, my beloved world!'
      >>3>   "hello, my beloved world!"
      <I3< Method "M1" with scope "TEST"
      >>3>   "hello, my beloved world!"
      <I3< Method "SHOW_M1" with scope "TEST"
      >I2> Method "SHOW_M1" with scope "TEST"
   26 *-2* say "about to return from show_m1 ..."
      >>2>   "about to return from show_m1 ..."
      <I2< Method "SHOW_M1" with scope "TEST"
```


Rony G. Flatscher

ooRexx (r12881)

.TraceObject~option="Normal"

```
t1=.test~new       -- create an instance
t2=.test~new       -- create another instance
t1~hey
t2~ho
say "-"~copies(50)
t2~start('hey')
t1~start('ho')
say "-"~copies(50)

::class test       -- some Rexx class
::method hey       -- by default: guarded
  say 'hey (guarded)'

::method ho unguarded    -- unguarded
  say 'ho (unguarded)'

::options trace all
```

Output:

```
      1 *-* t1=.test~new      -- instance
      2 *-* t2=.test~new      -- instance
      3 *-* t1~hey
        >I> Method "HEY" with scope "TEST"
     12 *-* say 'hey (guarded)'
hey (guarded)
        <I< Method "HEY" with scope "TEST"
      4 *-* t2~ho
        >I> Method "HO" with scope "TEST"
     15 *-* say 'ho (unguarded)'
ho (unguarded)
        <I< Method "HO" with scope "TEST"
      5 *-* say "-"~copies(50)
--------------------------------------------------
      6 *-* t2~start('hey')   -- async
      7 *-* t1~start('ho')    -- async
      8 *-* say "-"~copies(50)
--------------------------------------------------
        >I> Method "HEY" with scope "TEST"
     12 *-* say 'hey (guarded)'
hey (guarded)
        >I> Method "HO" with scope "TEST"
     15 *-* say 'ho (unguarded)'
ho (unguarded)
        <I< Method "HEY" with scope "TEST"
        <I< Method "HO" with scope "TEST"
```


Rony G. Flatscher

.TraceObject~option="Thread"

```
t1=.test~new       -- create an instance
t2=.test~new       -- create another instance
t1~hey
t2~ho
say "-"~copies(50)
t2~start('hey')
t1~start('ho')
say "-"~copies(50)

::class test       -- some Rexx class
::method hey       -- by default: guarded
  say 'hey (guarded)'

::method ho unguarded    -- unguarded
  say 'ho (unguarded)'

::options trace all
```

Output:

```
     1 *-2* t1=.test~new      -- instance
     2 *-2* t2=.test~new      -- instance
     3 *-2* t1~hey
       >I2> Method "HEY" with scope "TEST"
    12 *-2* say 'hey (guarded)'
hey (guarded)
       <I2< Method "HEY" with scope "TEST"
     4 *-2* t2~ho
       >I2> Method "HO" with scope "TEST"
    15 *-2* say 'ho (unguarded)'
ho (unguarded)
       <I2< Method "HO" with scope "TEST"
     5 *-2* say "-"~copies(50)
--------------------------------------------------
     6 *-2* t2~start('hey')   -- async
     7 *-2* t1~start('ho')    -- async
     8 *-2* say "-"~copies(50)
--------------------------------------------------
       >I3> Method "HEY" with scope "TEST"
    12 *-3* say 'hey (guarded)'
hey (guarded)
       >I4> Method "HO" with scope "TEST"
    15 *-4* say 'ho (unguarded)'
ho (unguarded)
       <I3< Method "HEY" with scope "TEST"
       <I4< Method "HO" with scope "TEST"
```


Rony G. Flatscher

.TraceObject~option="Standard"

```
t1=.test~new       -- create an instance
t2=.test~new       -- create another instance
t1~hey
t2~ho
say "-"~copies(50)
t2~start('hey')
t1~start('ho')
say "-"~copies(50)

::class test       -- some Rexx class
::method hey       -- by default: guarded
  say 'hey (guarded)'

::method ho unguarded    -- unguarded
  say 'ho (unguarded)'

::options trace all
```

Output:

```
[T2 I2]                      1 *-* t1=.test~new      -- instance
[T2 I2]                      2 *-* t2=.test~new      -- instance
[T2 I2]                      3 *-* t1~hey
[T2 I3 Gu A1 L0   ]            >I> Method "HEY" with scope "TEST"
[T2 I3 G  A1 L1 * ]         12 *-* say 'hey (guarded)'
hey (guarded)
[T2 I3 Gu A1 L0   ]            <I< Method "HEY" with scope "TEST"
[T2 I2]                      4 *-* t2~ho
[T2 I4 U  A2 L0   ]            >I> Method "HO" with scope "TEST"
[T2 I4 U  A2 L0   ]         15 *-* say 'ho (unguarded)'
ho (unguarded)
[T2 I4 U  A2 L0   ]            <I< Method "HO" with scope "TEST"
[T2 I2]                      5 *-* say "-"~copies(50)
--------------------------------------------------
[T2 I2]                      6 *-* t2~start('hey')   -- async
[T2 I2]                      7 *-* t1~start('ho')    -- async
[T2 I2]                      8 *-* say "-"~copies(50)
--------------------------------------------------
[T3 I5 Gu A2 L0   ]            >I> Method "HEY" with scope "TEST"
[T3 I5 G  A2 L1 * ]         12 *-* say 'hey (guarded)'
hey (guarded)
[T4 I6 U  A1 L0   ]            >I> Method "HO" with scope "TEST"
[T4 I6 U  A1 L0   ]         15 *-* say 'ho (unguarded)'
ho (unguarded)
[T3 I5 Gu A2 L0   ]            <I< Method "HEY" with scope "TEST"
[T4 I6 U  A1 L0   ]            <I< Method "HO" with scope "TEST"
```

.TraceObject~option="Full"

```
t1=.test~new        -- create an instance
t2=.test~new        -- create another instance
t1~hey
t2~ho
say "-"~copies(50)
t2~start('hey')
t1~start('ho')
say "-"~copies(50)

::class test        -- some Rexx class
::method hey        -- by default: guarded
  say 'hey (guarded)'

::method ho unguarded   -- unguarded
  say 'ho (unguarded)'

::options trace all
```

Output:

```
[R1 T2 I2]                           1 *-* t1=.test~new      -- instance
[R1 T2 I2]                           2 *-* t2=.test~new      -- instance
[R1 T2 I2]                           3 *-* t1~hey
[R1 T2 I3 Gu A1 L0   ]                 >I> Method "HEY" with scope "TEST"
[R1 T2 I3 G  A1 L1 * ]              12 *-* say 'hey (guarded)'
hey (guarded)
[R1 T2 I3 Gu A1 L0   ]                 <I< Method "HEY" with scope "TEST"
[R1 T2 I2]                           4 *-* t2~ho
[R1 T2 I4 U  A2 L0   ]                 >I> Method "HO" with scope "TEST"
[R1 T2 I4 U  A2 L0   ]              15 *-* say 'ho (unguarded)'
ho (unguarded)
[R1 T2 I4 U  A2 L0   ]                 <I< Method "HO" with scope "TEST"
[R1 T2 I2]                           5 *-* say "-"~copies(50)
--------------------------------------------------
[R1 T2 I2]                           6 *-* t2~start('hey')   -- async
[R1 T2 I2]                           7 *-* t1~start('ho')    -- async
[R1 T2 I2]                           8 *-* say "-"~copies(50)
--------------------------------------------------
[R1 T3 I5 Gu A2 L0   ]                 >I> Method "HEY" with scope "TEST"
[R1 T3 I5 G  A2 L1 * ]              12 *-* say 'hey (guarded)'
hey (guarded)
[R1 T4 I6 U  A1 L0   ]                 >I> Method "HO" with scope "TEST"
[R1 T4 I6 U  A1 L0   ]              15 *-* say 'ho (unguarded)'
ho (unguarded)
[R1 T3 I5 Gu A2 L0   ]                 <I< Method "HEY" with scope "TEST"
[R1 T4 I6 U  A1 L0   ]                 <I< Method "HO" with scope "TEST"
```

Analyzing a Multithreaded ooRexx Program by Itself, 1

- Purpose of the program
  - Defines class Test
    - The guarded method init (constructor) initializes the lock attribute to .true
    - The unguarded method wait uses the guard on keyword instruction to wait until the lock attribute turns .false
    - The guarded method do carries out some work and at the end clears the lock attribute by setting it to .false
  - Defines an options directive to trace the entire program with trace results
  - Main program ("prolog")
    - Configures the .TraceObject class to collect all created trace objects
    - Creates an instance of the .Test class and sends it the message do asynchronously
    - Sends the wait message to wait until the asynchronously running method do has ended
    - Displays all collected trace objects in Standard format

Analyzing a Multithreaded ooRexx Program by Itself, 2

```
trace off        -- do not trace prolog
.TraceObject~option="P" -- Profile/Probe
.TraceObject~collector=.array~new
o=.Test~new      -- create an instance
say "starting worker asynchronously ..."
o~start("do")   -- dispatched on a new thread
say "about to wait for worker to end ..."
o~wait           -- synchronize with worker
say "waiting is over"
trace off
arr=.TraceObject~collector   -- get collection
.TraceObject~collector=.nil -- stop collecting
say
say 'setting: .TraceObject~option="Standard"'
.TraceObject~option="S"
say "displaying" arr~items "collected trace objects:"
do traceObj over arr
   say traceObj
end

::class Test
::method init   -- constructor
  expose lock   -- access attribute
  lock=.true    -- set default

::method wait unguarded
  expose lock   -- access attribute
  guard on when lock=.false

::method do     -- do some work
  expose lock   -- access attribute
  t=random(1,999)/1000
  say "do: working for" t "secs"
  call sysSleep t
  say "do: ending work"
  lock=.false   -- release lock

::options trace results   -- trace everything
```

Output:

```
     1 *-* trace off       -- do not trace prolog
starting worker asynchronously ...
about to wait for worker to end ...
do: working for 0.422 secs
do: ending work
waiting is over

setting: .TraceObject~option="Standard"
displaying 24 collected trace objects:
[T1  I2  Gu A1  L0    ]        >I> Method "INIT" with scope "TEST"
[T1  I2  G  A1  L1  * ]     23 *-* expose lock   -- access attribute
[T1  I2  G  A1  L1  * ]     24 *-* lock=.true    -- set default
[T1  I2  G  A1  L1  * ]        >>>   "1"
[T1  I2  Gu A1  L0    ]        <I< Method "INIT" with scope "TEST"
[T1  I3  U  A1  L0    ]        >I> Method "WAIT" with scope "TEST"
[T1  I3  U  A1  L0    ]     27 *-* expose lock   -- access attribute
[T1  I3  U  A1  L0    ]     28 *-* guard on when lock=.false
[T1  I3  Ug A1  L1  *W]        >K>   "WHEN" => "0"
[T2  I4  Gu A1  L0    ]        >I> Method "DO" with scope "TEST"
[T2  I4  G  A1  L1  * ]     31 *-* expose lock   -- access attribute
[T2  I4  G  A1  L1  * ]     32 *-* t=random(1,999)/1000
[T2  I4  G  A1  L1  * ]        >>>   "0.422"
[T2  I4  G  A1  L1  * ]     33 *-* say "do: working for" t "secs"
[T2  I4  G  A1  L1  * ]        >>>   "do: working for 0.422 secs"
[T2  I4  G  A1  L1  * ]     34 *-* call sysSleep t
[T2  I4  G  A1  L1  * ]        >>>   "0"
[T2  I4  G  A1  L1  * ]     35 *-* say "do: ending work"
[T2  I4  G  A1  L1  * ]        >>>   "do: ending work"
[T2  I4  G  A1  L1  * ]     36 *-* lock=.false   -- release lock
[T2  I4  G  A1  L1  * ]        >>>   "0"
[T2  I4  Gu A1  L0    ]        <I< Method "DO" with scope "TEST"
[T1  I3  Ug A1  L1  * ]        >K>   "WHEN" => "1"
[T1  I3  U  A1  L0    ]        <I< Method "WAIT" with scope "TEST"
```

Roundup

- ooRexx 5.1, released May 4th, 2025
  - Bug fixes, new features
  - Introduces the .TraceObject class as a subclass of .StringTable
    - Global configuration
    - Each traced instruction causes a trace object to be created with the trace information
    - Allows creating custom makeString methods for formatting the tracelines
  - Allows for gaining full insight into multithreaded execution and locking
    - For the first time possible to analyze hanging ooRexx programs in depth
      - Run the hanging program on a separate thread after configuring .TraceObject
      - After a predefined timeout, analyze the collected trace objects
  - E.g. allows for creating trace logs for later analysis