ooRexx Documentation 5.1.0

Open Object Rexx

Rexx Extensions Library Reference

ooRexx Documentation 5.1.0 Open Object Rexx
Rexx Extensions Library Reference
Edition 2024.04.20 (last revised on 2024-01-06 with r12767)

Author	W. David Ashley
Author	Rony G. Flatscher
Author	Mark Hessling
Author	Rick McGuire
Author	Lee Peedin
Author	Oliver Sims
Author	Erich Steinböck
Author	Jon Wolfers

Preface

This book describes a number of extension classes to Open Object Rexx.

This book is intended for people who plan to develop applications using Rexx and the extension classes. Its users range from the novice to experienced ooRexx users.

This book is a reference rather than a tutorial. It assumes you are already familiar with object-oriented programming concepts.

Descriptions include the use and syntax of the language and explain how the language processor "interprets" the language as a program is running.

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

1.1. Typographic Conventions

Typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold` is used to highlight literal strings, class names, or inline code examples. For example:

> The `Class` class comparison methods return `.true` or `.false`, the result of performing the comparison operation.
>
> This method is exactly equivalent to `subWord(`*n*`, 1)`.

`Mono-spaced Normal` denotes method names or source code in program listings set off as separate examples.

> This method has no effect on the action of any `hasEntry`, `hasIndex`, `items`, `remove`, or `supplier` message sent to the collection.

```
-- reverse an array
a = .Array~of("one", "two", "three", "four", "five")

-- five, four, three, two, one
aReverse = .CircularQueue~new(a~size)~appendAll(a)~makeArray("lifo")
```

Proportional Italic is used for method and function variables and arguments.

> A supplier loop specifies one or two control variables, *index*, and *item*, which receive a different value on each repetition of the loop.
>
> Returns a string of length *length* with *string* centered in it and with *pad* characters added as necessary to make up length.

1.2. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed, like mandatory initialization. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. How to Read the Syntax Diagrams

Throughout this book, syntax is described using the structure defined below.

- Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

  The ►►— symbol indicates the beginning of a statement.

  The —··· symbol indicates that the statement syntax is continued on the next line.
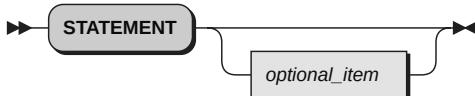
  The ···— symbol indicates that a statement is continued from the previous line.

  The —►◄ symbol indicates the end of a statement.
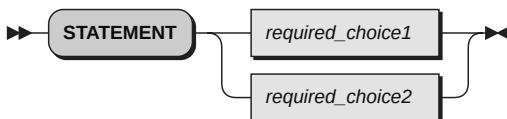
- Required items appear on the horizontal line (the main path).

  ►►—(**STATEMENT**)——(*required_item*)——►◄
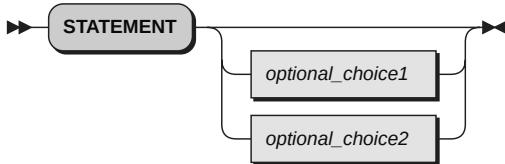
- Optional items appear below the main path.

  ►►—(**STATEMENT**)—————————►◄
                      └—(*optional_item*)—┘
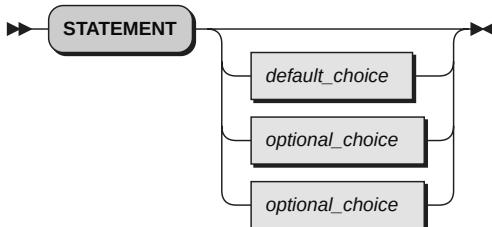
- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.

  ►►—(**STATEMENT**)——(*required_choice1*)——►◄
                      └—(*required_choice2*)—┘
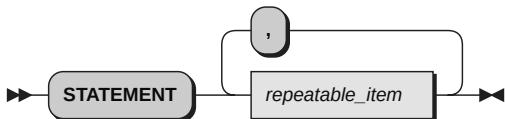
- If choosing one of the items is optional, the entire stack appears below the main path.

- If one of the items is the default, it is usually the topmost item of the stack of items below the main path.

- A path returning to the left above the main line indicates an item that can be repeated.
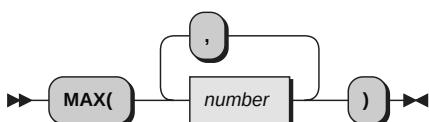
A repeat path above a stack indicates that you can repeat the items in the stack.

- A pointed rectangle around an item indicates that the item is a fragment, a part of the syntax diagram that appears in greater detail below the main diagram.

- Keywords appear in uppercase (for example, **SIGNAL**). They must be spelled exactly as shown but you can type them in upper, lower, or mixed case. Variables appear in all lowercase letters (for example, *index*). They represent user-supplied names or values.

- If punctuation marks, parentheses, arithmetic operators, or such symbols are shown, you must enter them as part of the syntax.

The following example shows how the syntax is described:

3. Getting Help and Submitting Feedback

The Open Object Rexx Project has a number of methods to obtain help and submit feedback for ooRexx and the extension packages that are part of ooRexx. These methods, in no particular order of preference, are listed below.

3.1. The Open Object Rexx SourceForge Site

Open Object Rexx utilizes SourceForge to house its source repositories, mailing lists and other project features at *https://sourceforge.net/projects/oorexx*. ooRexx uses the Developer and User mailing lists at *https://sourceforge.net/p/oorexx/mailman* for discussions concerning ooRexx. The ooRexx user is most likely to get timely replies from one of these mailing lists.

Here is a list of some of the most useful facilities provided by SourceForge.

The Developer Mailing List

Subscribe to the oorexx-devel mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-devel* to discuss ooRexx project development activities and future interpreter enhancements. You can find its archive of past messages at *http://sourceforge.net/mailarchive/forum.php?forum_name=oorexx-devel*.

The Users Mailing List

Subscribe to the oorexx-users mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-users* to discuss how to use ooRexx. It also supports a historical archive of past messages.

The Announcements Mailing List

Subscribe to the oorexx-announce mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-announce* to receive announcments of significant ooRexx project events.

The Bug Mailing List

Subscribe to the oorexx-bugs mailing list at *https://lists.sourceforge.net/lists/listinfo/oorexx-bugs* to monitor changes in the ooRexx bug tracking system.

Bug Reports

You can view ooRexx bug reports at *https://sourceforge.net/p/oorexx/bugs*. To be able to create new bug reports, you will need to first register for a SourceForge userid at *https://sourceforge.net/user/registration*. When reporting a bug, please try to provide as much information as possible to help developers determine the cause of the issue. Sample program code that can reproduce your problem will make it easier to debug reported problems.

Documentation Feedback

You can submit feedback for, or report errors in, the documentation at *https://sourceforge.net/p/oorexx/documentation*. Please try to provide as much information in a documentation report as possible. In addition to listing the document and section the report concerns, direct quotes of the text will help the developers locate the text in the source code for the document. (Section numbers are generated when the document is produced and are not available in the source code itself.) Suggestions as to how to reword or fix the existing text should also be included.

Request For Enhancement

You can new suggest ooRexx features or enhancements at *https://sourceforge.net/p/oorexx/feature-requests*.

Patch Reports

If you create an enhancement patch for ooRexx please post the patch at *https://sourceforge.net/p/oorexx/patches*. Please provide as much information in the patch report as possible so that the developers can evaluate the enhancement as quickly as possible.

Please do not post bug fix patches here, instead you should open a bug report at *https://sourceforge.net/p/oorexx/bugs* and attach the patch to it.

The ooRexx Forums

The ooRexx project maintains a set of forums that anyone may contribute to or monitor. They are located at *https://sourceforge.net/p/oorexx/discussion*. There are currently three forums available: Help, Developers and Open Discussion. In addition, you can monitor the forums via email.

3.2. The Rexx Language Association Mailing List

The Rexx Language Association maintains a forum at *http://www.rexxla.org/forum.html*.

3.3. comp.lang.rexx Newsgroup

The comp.lang.rexx newsgroup at *https://groups.google.com/forum/#!forum/comp.lang.rexx* is a good place to obtain help from many individuals within the Rexx community. You can obtain help on Open Object Rexx and other Rexx interpreters and tools.

4. Related Information

See also: *Open Object Rexx: Reference*

csvStream Class

The csvStream class extends the Stream class to read & write CSV files directly to Collection Objects.

The csvStream Class is a subclass of the Stream Class.

1.1. Translation of data involved in the csvStream class

CSV file literals are surrounded by quotes "". These are removed by CSVLineIn and inserted by CSVLineOut. Quotes within CSV data are represented self escaped ie: " appears as "". These are translated by the CSVLineIn and CSVLineOut methods. CSVLineOut encapsulates non-numeric fields in "" unless they already are. CSV literal strings can contain line-end sequences. To create multi-line fields use the line-end character provided by the operating system dependant ooRexx local variable .endofline.

1.2. Methods The csvStream Class defines

CLOSE (overrides stream class method)
CSVLINEIN
CSVLINEOUT
GETHEADERS
SETHEADERS
INIT (overrides stream class method)
OPEN (overrides stream class method)
STATE (overrides stream class method)
DESCRIPTION (overrides stream class method)

1.3. Attributes of the csvStream Class

HEADERS~FIELD(n)~NAME
HEADERS~FIELD(n)~LITERAL
SKIPHEADERS
DELIMITER
QUALIFIER
STRIPOPTION
STRIPCHAR

1.4. Methods Inherited from the Stream Class

ARRAYIN
ARRAYOUT
CHARIN
CHAROUT
CHARS
COMMAND
DESCRIPTION
FLUSH
LINEIN
LINEOUT

LINES
MAKEARRAY
POSITION
QUALIFY
QUERY
SAY
SEEK
STATE
SUPPLIER

1.5. Methods inherited from the Object class

NEW (Class method)
Operator methods: =, ==, \=, ><, <>, \==
CLASS
COPY
DEFAULTNAME>
HASMETHOD
OBJECTNAME
OBJECTNAME=
REQUEST
RUN
SETMETHOD
START
STRING
UNSETMETHOD

> **Note**
>
> The Stream class also has available class methods that its metaclass, the Class class, defines.

1.6. Methods

1.6.1. Close Method

▶▶──(close)──▶◀

Closes the stream that receives the message. CLOSE returns READY: if closing the stream is successful, or an appropriate error message. If you have tried to close an unopened file, then the CLOSE method returns a null string (""). If you specified headersExist when you created this instance then the headers will be updated to the stream at this point if they have been changed.

1.6.2. CSVLineIn Method

▶▶──(csvlinein)──▶◀

Reads and returns a row of CSV data from the stream. Note that a row of data may be stored in more than one logical line of the stream. An array is returned, the nth element of which contains the nth field from the Row.

Two other attributes exist after performing a CSVLineIn

Rawtext is a String Object containing the raw text that the row consists of.
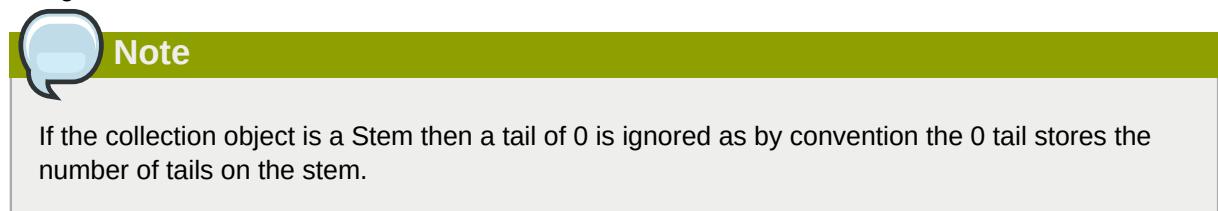
Values is a Table Object mapping field data onto field-names. This is only available if headersExist is specified on the init method.

Badly formed CSV data. Where the data read in by CSVLineIn is not well formed CSV data the results are unpredictable. The class can detect some errors in the incoming data, and where such an error is detected the STATE method will return ERROR and the DESCRIPTION method will give extra error information. Where the provenance of the data is outside your control it would be well to check the STATE after every CSVLineIn. Subsequent calls to CSVLineIn may be able to recover and return subsequent rows from the file but this should not be expected to be the norm. Subsequent calls to CSVLineIn after an error will not return the STATE to READY. It will remain at ERROR until the Stream class resets it (ie: when you close the CSVStream)

1.6.3. CSVLineOut Method

Writes a row of CSV data to a stream. Note that a row of data may be stored in more than one logical line of the stream. If the stream was instantiated with headersExist as .true then the collection-object may be a directory, table or stem object mapping headers onto CSV fields. Otherwise the collection-object must be an array or a collection with a makeArray method and the nth element of the collection will be placed in the nth field of the CSV file. Any occurrences of the Nil Object are stored as null strings in the file.

> **Note**
>
> If the collection object is a Stem then a tail of 0 is ignored as by convention the 0 tail stores the number of tails on the stem.
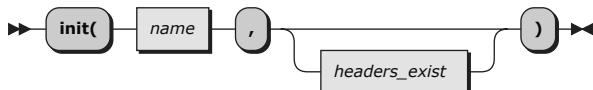
1.6.4. GetHeaders Method



Returns a csvStreamHeader object.

Get headers will return a csvStreamHeader object containing details of the column header names and whether they are literal values or not. Column header names that exist before the csvStream is opened are present as soon as the file is opened, but literal information will not be present till the first CSVLineIn or CSVLineOut is issued.

1.6.5. INIT Method

Initializes a stream object for a stream named name, but does not open the stream.

The second optional parameter if passed a value of 'H' (or .true) indicates that the first row of the stream is (or is to be) a row of headers containing the names of the CSV fields. Note that header fields are case sensitive. This means that 'name' and 'Name' and 'NAME' will all be treated as separate columns.

1.6.6. OPEN Method

Parameters are as the Stream class Open Method

Opens the stream to which you send the message and returns READY:. If the method is unsuccessful, it returns an error message string in the same form that the DESCRIPTION method uses. See the Stream Class Open Method for a fuller description.

1.6.6.1. Changing the behaviour of a csvStream object

Before issuing the Open message, you can affect the csvStream's behaviour by setting the attribute skipHeaders to .false. This will mean that the first row returned by CSVLineIn on a csvStream where headers exist is the header row, rather than the default behaviour which is to return the first row of data.

After issuing the OPEN message to a csvStream which has been opened with headers exist, the class will attempt to learn the nature of the fields by analysing the data. You can teach it by setting the headers field attributes name and literal. For instance:

Example 1.1. Describing header fields

```
/* set the name of the second field to 'Height' */
MyCsvStream~headers~field(2)~name='Height'

/* tell the stream to treat the
third column as literal data rather than numeric */
MyCsvStream~headers~field(3)~literal= .true
```

By default the delimiter csvStream expects is a comma (after all CSV stands for Comma Separated Variables) and literals are qualified by a double inverted comma. However you can create and read files with other delimiters or qualifiers by changing the attributes delimiter and qualifier after instantiating A csvStream object. For instance, to use ; as a delimiter and ' as a qualifier do the following:

Example 1.2. Describing field delimiters

```
MyCsvStream = .csvStream~new
MyCsvStream~delimiter=";"
MyCsvStream~qualifier="'"
```

If the attribute StripOption is set to 'L', 'T' or 'B' then data is stripped using that option before CSVLineIn inserts it in the returned array. The default of 'N' means no stripping is performed. One can specify which character to strip using the attribute stripChar which defaults to blank.

Example 1.3. Setting the strip option

```
MyCsvStream~StripOption = 'T' /* strip trailing blanks */
```

or

Example 1.4. Removing leading zeros

```
MyCsvStream~StripOption = 'L'
MyCsvStream~stripChar = '0' /* strip Leading zeroes */
```

1.6.7. SetHeaders Method

Passed a csvStreamHeader object will apply it to the csvStream. Together with Get Headers this allows you to base one CSV file on another.

1.7. Attributes

1.7.1. DELIMITER Attribute

This is the character which delimits the fields (as long as it does not appear within a literal). In a standard CSV file it is a comma ,. See Changing the behaviour of a csvStream object under the OPEN method for an example of changing the delimiter.

1.7.2. HEADERS Attribute

Access is available to the Field definition table for files with headers. There are two entries, NAME & LITERAL. NAME is the Name for that particular column. If LITERAL is .true then that column will be treated as a literal even if the data in it is numeric. If any entry in a column is non-numeric then the entire column is treated as a literal. See Changing the behaviour of a csvStream object under the OPEN method for an example of accessing the table.

1.7.3. QUALIFIER Attribute

The Qualifier is the character that surrounds literal fields. Delimiters that appear within literal fields are ignored. In a standard CSV file the qualifier is a double quotation mark ("). See Changing the behaviour of a csvStream object under the OPEN method for an example of changing the qualifier.

1.7.4. SKIPHEADERS Attribute

See *Changing the behaviour of a csvStream object* under the OPEN method.

1.8. Examples

Example 1.5. Files without headers

```
csv = .csvStream~new('MyData.csv') /* 2nd arg defaults to no headers */
/* csv~skipHeaders = .false          UnNoOp to return header line */

csv~open('write')                      /*=File looks like this=*/
csv~CSVLineOut(.array~of('red','stop')) /* "red","stop"        */
csv~CSVLineOut(.array~of('green','go')) /* "green","go"        */
csv~close                              /*====================*/

csv~open('read')                       /*======Returns=======*/
do while csv~chars > 0                 /* New record         */
   dataArr = csv~CSVLineIn             /* field 1: red       */
   say 'New record'                    /* field 2: stop      */
   do I = 1 to dataArr~last           /* New record         */
   say 'field' I':' dataArr[I]        /* field 1: green     */
   end                                 /* field 2: go        */
end                                    /*====================*/
csv~close

::requires 'csvStream.cls'
```

Example 1.6. Files with headers

```
csv = .csvStream~new('headered.csv', .true)
csv~open        /* Stream class defaults to both ie:readWrite */
myTable = .table~new
myTable~put('red','colour')
myTable~put('stop','action')
csv~CSVLineout(myTable)
myTable~put('green','colour')
myTable~put('go','action')
csv~CSVLineout(myTable)
csv~close

Csv~open('read')                       /*======Returns=======*/
Do while csv~chars > 0                 /* new record         */
   Csv~csvLineIn                       /* colour: red        */
   Say 'new record'                    /* action: stop       */
   Do field over csv~values           /* new record         */
      Say field':' csv~values~at(field) /* colour: green     */
   End                                 /* action: go         */
End                                    /*====================*/
csv~close

::requires 'csvStream.cls'
```

Example 1.7. Example with error checking

```
csv = .csvStream~new('BadData.csv')

csv~open('read')
if csv~state = 'READY'
then do
   do while csv~chars > 0
      dataArr = csv~CSVLineIn
      if csv~state = 'ERROR'
```

```
          then do
             say 'BAD DATA IN CSV FILE -' csv~description
             leave
          end
          say 'New record'
          do I = 1 to dataArr~last
          say 'field' I':' dataArr[I]
          end
    end
    csv~close
end
else say 'COULD NOT OPEN CSV FILE -' csv~description

::requires 'csvStream.cls'
```

JSON Support

The **_JSON_** class allows to create and process *JSON (JavaScript Object Notation)* strings. The *JSON* definition used for this implementation can be found at *https://www.ietf.org/rfc/rfc4627.txt*.

> ### 💬 Note
>
> This implementation does not transform into or from Unicode escape sequences in the form of "\uXXXX".

To use the **_JSON_** class, you must place the following directive in your script:

```
::requires "json.cls"         -- get JSON support
```

This will load the classes **_JSON_** and **_JSONBoolean_**.

Here is an example of using the JSON support of ooRexx for encoding a directory as a JSON string.

Example 2.1. Using the JSON support.

```
obj=.Directory~new           -- a directory (a MapCollection)
obj['Name']    ="Eli"        -- a string value
obj['Children']=3            -- a number value
obj['Parents']= .array~of('Eve', 'Adam')  -- an array value (an OrderedCollection)
obj['Salary']  =.nil         -- no value at all (not employed)
obj['rich']    =.json~false  -- use .Json's false (a JSONBoolean)

say .JSON~toJson(obj)        -- show minimized JSON string
say "---"
say .JSON~toJson(obj,.true)  -- create and show legible JSON string

::requires "json.cls"        -- get JSON support
```

The program yields the following output:

```
{"Children":3,"Name":"Eli","Parents":["Eve","Adam"],"Salary":null,"rich":false}
---
{
    "Children": 3,
    "Name": "Eli",
    "Parents": [
        "Eve",
        "Adam"
    ],
    "Salary": null,
    "rich": false
}
```

Here is an example of using the JSON support of ooRexx for decoding JSON strings.

Example 2.2. Using the JSON support.

```
do i=1 to .resources~items
    resName = "JSON_STRING_"i
```

```
      jsonString = .resources~entry(resName)~makeString
      say "round #" i", resource" resName":"
      say
      say "jsonString:" jsonString  -- show JSON string
      say
      d=.JSON~fromJSON(jsonString)  -- parse JSON string
      say "after parsing (fromJSON), result in variable 'd' ("d"):"
      say
      say "Name      :" d["Name"]
      say "Children  :" d["Children"]
      parents = d["Parents"]
      say "Parents   :" parents~toString(,',') "("parents~string")"
      say "Salary    :" d["Salary"]
      rich = d["rich"]
      say "rich      :" rich "(a" rich~class~id")"
      say "-"~copies(80)
   end

   ::requires "json.cls"         -- get JSON support

   ::resource JSON_STRING_1      -- minimized JSON string
   {"Children":3,"Name":"Eli","Parents":["Eve","Adam"],"Salary":null,"rich":false}
   ::END

   ::resource JSON_STRING_2      -- legible JSON string
   {
      "Children": 3,
      "Name": "Eli",
      "Parents": [
         "Eve",
         "Adam"
      ],
      "Salary": null,
      "rich": false
   }
   ::END
```

The program yields the following output:

```
round # 1, resource JSON_STRING_1:

jsonString: {"Children":3,"Name":"Eli","Parents":["Eve","Adam"],"Salary":null,"rich":false}

after parsing (fromJSON), result in variable 'd' (a Directory):

Name      : Eli
Children  : 3
Parents   : Eve,Adam (an Array)
Salary    : The NIL object
rich      : 0 (a JSONBoolean)
--------------------------------------------------------------------------------
round # 2, resource JSON_STRING_2:

jsonString: {
   "Children": 3,
   "Name": "Eli",
   "Parents": [
      "Eve",
      "Adam"
   ],
   "Salary": null,
   "rich": false
}

after parsing (fromJSON), result in variable 'd' (a Directory):
```

```
Name      : Eli
Children  : 3
Parents   : Eve,Adam (an Array)
Salary    : The NIL object
rich      : 0 (a JSONBoolean)
---------------------------------------------------------------------------
```

2.1. JSONBoolean

The **JSONBoolean** class can be used to test whether a boolean value is an instance of it and provides the following public class methods:

false
true

2.1.1. false

This class method returns the *false* value which behaves likes ooRexx' *.false* and can be therefore used interchangeably. Using this value allows the methods *toJSON* and *toJSONFile* to correctly create the JSON boolean *false* encodings.

2.1.2. true

This class method returns the *true* value which behaves likes ooRexx' *.true* and can be thererfore used interchangeably. Using this value allows the methods *toJSON* and *toJSONFile* to correctly create the JSON boolean *true* encodings.

2.2. JSON

The **JSON** class provides the following public methods:

init
false
fromJSON
fromJSONFile
toJSON
toJSONFile
true

> **Note**
>
> The methods *toJSON* and *toJSONFile* will render Rexx objects as follows:
>
> - An instance of type **String**: replaces all double-quotes with the escape sequence \" and encloses the string value in double quotes, unless the string is a valid Rexx number in which case the string is used as is and no enquoting takes place
>
> - An instance of type **MapCollection**: creates a comma separated list of name/value pairs enclosed in curly brackets.
>
> - An instance of type **OrderedCollection**: creates a comma separated list of ordered values enclosed in square brackets.
>
> - An object that is *.nil* as the unquoted string *null*.
>
> - An object that has a method named *makeJson* (supporting the ooRexx *request* protocol for conversion to *JSON*): uses the string that *makeJson* is expected to return.
>
>   The **JSONBoolean** *false* value will be rendered as the unquoted string *false*, the *true* value will be rendered as the unquoted string *true*.
>
> - An object that has a method named *makeArray* (supporting the ooRexx *request* protocol for conversion to an ooRexx *Array*): uses the returned array that *makeArray* is expected to return and renders it as an **OrderedCollection** (see above).
>
> - An object that has a method named *makeString* (supporting the ooRexx *request* protocol for conversion to an ooRexx *String*): uses the string that *makeString* is expected to return and renders it as a **String** (see above).
>
> - An object for which a conversion to JSON is unknown: sends the `string` message and renders the resulting string as a **String** (see above).

2.2.1. init

▶▶─(**init**)─▶◀

The `init` method initializes the JSON parser.

2.2.2. false

▶▶─(**false**)─▶◀

This class method returns the *JSONBoolean false* value which behaves likes ooRexx' *.false* and can be therefore used interchangeably. Using this value allows the methods *toJSON* and *toJSONFile* to correctly create the JSON boolean *false* encoding.

2.2.3. fromJSON

The `fromJSON` class and instance method creates and returns a Rexx object from the *JSON* encoded *string*.

> **Note**
>
> If the JSON string contained boolean values they will be represented by the respective **JSONBoolean** *true* or *false* values which are compatible with ooRexx' *.true* (the string/number *1*) or *.false* (the string/number *0*) values.

2.2.4. fromJSONFile
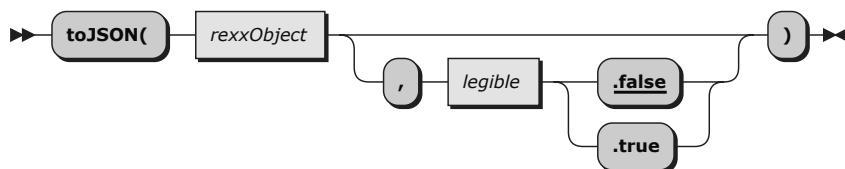

The `fromJSONFile` class method reads the JSON encoded data from *fileName*, creates and returns a Rexx object from the *JSON* encoded *string*.
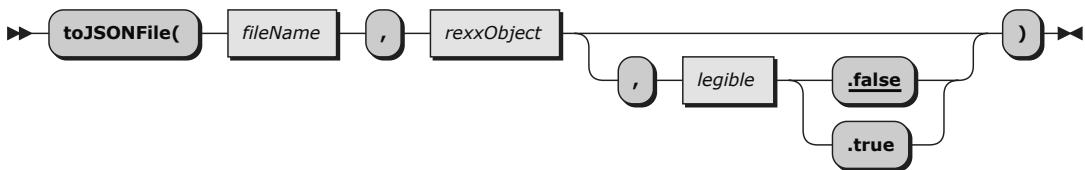
> **Note**
>
> If the JSON string contained boolean values they will be represented by the respective **JSONBoolean** *true* or *false* values which are compatible with ooRexx' *.true* (the string/number *1*) or *.false* (the string/number *0*) values.

2.2.5. toJSON

The `toJSON` class and instance method creates and returns a *JSON* encoded string representing *rexxObject*. If the optional *legible* argument is omitted then a minimized JSON string (does not contain any ignorable whitespace) gets created (defaults to *.false*). If the *legible* argument is supplied with a value of *.true* a legible JSON string gets created, which includes ignorable whitespace to ease reading for humans.

2.2.6. toJSONFile

```
►► toJSONFile( fileName , rexxObject )  ►◄
                          , legible .false
                                    .true
```

The `toJSONFile` class method writes the *JSON* encoded string representing *rexxObject* to *fileName*. If the optional *legible* argument is omitted then a minimized JSON string (does not contain any ignorable whitespace) gets created (defaults to *.false*). If the *legible* argument is supplied with a value of *.true* a legible JSON string gets created, which includes ignorable whitespace to ease reading for humans.

2.2.7. true

```
►► true ►◄
```

This class method returns the ***JSONBoolean*** *true* value which behaves likes ooRexx' *.true* and can be therefore used interchangeably. Using this value allows the methods *toJSON* and *toJSONFile* to correctly create the JSON boolean *true* encoding.

Host Emulator (HostEmu)

HostEmu is a subcommand environment that partially emulates a TSO/CMS environment. It provides a small subset of commands available in those environments which make the transition from a real host Rexx programming environment to a Linux/Windows ooRexx environment much easier. The following subcommands are available:

EXECIO

an I/O mechanism.

HI

halts the current Rexx program.

TE

deactivate the Rexx trace mechanism.

TS

activate the Rexx trace mechanism.

The HostEmu HI, TS, TE commands have no arguments that are acceptable in the HostEmu environment.

> **Note**
>
> The HI, TS, and TE commands can not be issued directly from a Linux/Windows command prompt. The only way to issue these commands is from within the currently running Rexx program (e. g. **address 'hostemu' 'hi'**) and they will only affect this Rexx program. Other running Rexx programs will be unaffected. As such, these command are only a very limited, much less useful substitute of the original TSO and CMS immediate commands.

The EXECIO subcommand is a simplified version of the mainframe command with only a small subset of the original EXECIO functionality supported.

To include and use the HostEmu subcommand environment you must place a ooRexx directive in your script. The following shows how to accomplish this.

```
::requires "hostemu" LIBRARY
```
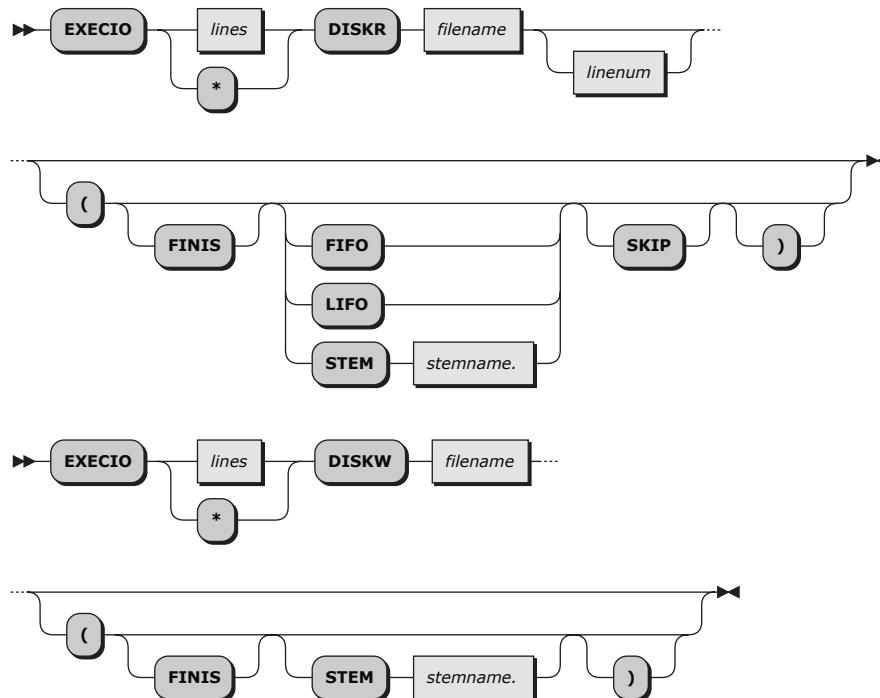
This will activate the environment. The subcommand name is "HostEmu" (the case of this string is not important). You can send commands to this environment via the ooRexx address statement. Here is an example.

```
address hostemu 'execio * diskr "./inputfile.txt" (finis stem in.'
```

Note that the file name MUST be placed within a set of quotation marks.

The example above should look very familiar to a mainframe Rexx programmer. The big difference is that a real file name is used instead of a DDNAME and the HostEmu environment is not the default address environment, thus the requirement that you either include the 'HostEmu' environment name in the address statement or you make the 'HostEmu' environment the default environment.

3.1. EXECIO subcommand

3.1.1. Command Options

lines

> Specifies the number of records (text lines) to read or write.

*

> Specifies that all remaining records are to be read or written.

DISKR

> The operation is a disk read operation.

DISKW

> The operation is a disk write operation.

filename

> The name of the file for the disk operation. If *filename* contains special characters (e. g. "/"), it will have to be enclosed in double quotes. For Unix systems, this means that *filename* will generally have to be quoted.

linenum

> The relative line number within the specified file where a DISKR operation is to begin. If *linenum* is not specified reading begins at the current position.

FINIS

> The file will be closed at the end of the operation. The default is to leave the file open.

STEM stemname.

> Specifies to read from or write to the specified stem. A trailing period is required or the name will be used as the root of a standard Rexx variable name. If *stemname* contains special characters, it will have to be enclosed in double quotes.

For DISKW operation, if STEM isn't specified, data is taken from the Rexx SESSION queue.

FIFO

Specifies to write to the Rexx SESSION queue in a first-in first-out order. This is the default.

LIFO

Specifies to write to the Rexx SESSION queue in a last-in first-out order.

SKIP

Specifies the number of records (text lines) to be skipped. No stem or queue operations will be performed in this case.

Note that the options between the brackets can be specified in any order.

3.1.2. Return codes

2

End-of-file was reached before the specified number of lines were read.

24

Bad parameter list, a wrong option, a non-numeric line number, or an invalid file name was specified.
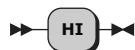
41

Out-of-memory

2008
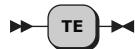
The variable name specified with the STEM option was invalid, or could not be read, or written to.

3.2. HI subcommand
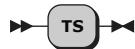
Halts the current Rexx program.

3.3. TE subcommand

Deactivate the Rexx trace mechanism.

3.4. TS subcommand

Activate the Rexx trace mechanism.

Appendix A. Notices

Any reference to a non-open source product, program, or service is not intended to state or imply that only non-open source product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any Rexx Language Association (RexxLA) intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-open source product, program, or service.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-open source products was obtained from the suppliers of those products, their published announcements or other publicly available sources. RexxLA has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-RexxLA packages. Questions on the capabilities of non-RexxLA packages should be addressed to the suppliers of those products.

All statements regarding RexxLA's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

A.1. Trademarks

Open Object Rexx™ and ooRexx™ are trademarks of the Rexx Language Association.

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both:

1-2-3
AIX
IBM
Lotus
OS/2
S/390
VisualAge

AMD is a trademark of Advance Micro Devices, Inc.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the Unites States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

A.2. Source Code For This Document

The source code for this document is available under the terms of the Common Public License v1.0 which accompanies this distribution and is available in the appendix *Appendix B, Common Public License Version 1.0*. The source code is available at *https://sourceforge.net/p/oorexx/code-0/HEAD/ tree/docs/*.

The source code for this document is maintained in DocBook SGML/XML format.

The railroad diagrams were generated with the help of "Railroad Diagram Generator" located at *http:// bottlecaps.de/rr/ui*. Special thanks to Gunther Rademacher for creating and maintaining this tool.

Appendix B. Common Public License Version 1.0

THE ACCOMPANYING PROGRAM IS PROVIDED UNDER THE TERMS OF THIS COMMON PUBLIC LICENSE ("AGREEMENT"). ANY USE, REPRODUCTION OR DISTRIBUTION OF THE PROGRAM CONSTITUTES RECIPIENT'S ACCEPTANCE OF THIS AGREEMENT.

B.1. Definitions

"Contribution" means:

1. in the case of the initial Contributor, the initial code and documentation distributed under this Agreement, and

2. in the case of each subsequent Contributor:
   a. changes to the Program, and

   b. additions to the Program;

where such changes and/or additions to the Program originate from and are distributed by that particular Contributor. A Contribution 'originates' from a Contributor if it was added to the Program by such Contributor itself or anyone acting on such Contributor's behalf. Contributions do not include additions to the Program which: (i) are separate modules of software distributed in conjunction with the Program under their own license agreement, and (ii) are not derivative works of the Program.

"Contributor" means any person or entity that distributes the Program.

"Licensed Patents " mean patent claims licensable by a Contributor which are necessarily infringed by the use or sale of its Contribution alone or when combined with the Program.

"Program" means the Contributions distributed in accordance with this Agreement.

"Recipient" means anyone who receives the Program under this Agreement, including all Contributors.

B.2. Grant of Rights

1. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free copyright license to reproduce, prepare derivative works of, publicly display, publicly perform, distribute and sublicense the Contribution of such Contributor, if any, and such derivative works, in source code and object code form.

2. Subject to the terms of this Agreement, each Contributor hereby grants Recipient a non-exclusive, worldwide, royalty-free patent license under Licensed Patents to make, use, sell, offer to sell, import and otherwise transfer the Contribution of such Contributor, if any, in source code and object code form. This patent license shall apply to the combination of the Contribution and the Program if, at the time the Contribution is added by the Contributor, such addition of the Contribution causes such combination to be covered by the Licensed Patents. The patent license shall not apply to any other combinations which include the Contribution. No hardware per se is licensed hereunder.

3. Recipient understands that although each Contributor grants the licenses to its Contributions set forth herein, no assurances are provided by any Contributor that the Program does not infringe the patent or other intellectual property rights of any other entity. Each Contributor disclaims any liability to Recipient for claims brought by any other entity based on infringement

of intellectual property rights or otherwise. As a condition to exercising the rights and licenses granted hereunder, each Recipient hereby assumes sole responsibility to secure any other intellectual property rights needed, if any. For example, if a third party patent license is required to allow Recipient to distribute the Program, it is Recipient's responsibility to acquire that license before distributing the Program.

4. Each Contributor represents that to its knowledge it has sufficient copyright rights in its Contribution, if any, to grant the copyright license set forth in this Agreement.

B.3. Requirements

A Contributor may choose to distribute the Program in object code form under its own license agreement, provided that:

1. it complies with the terms and conditions of this Agreement; and

2. its license agreement:

   a. effectively disclaims on behalf of all Contributors all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose;

   b. effectively excludes on behalf of all Contributors all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits;

   c. states that any provisions which differ from this Agreement are offered by that Contributor alone and not by any other party; and

   d. states that source code for the Program is available from such Contributor, and informs licensees how to obtain it in a reasonable manner on or through a medium customarily used for software exchange.

When the Program is made available in source code form:

1. it must be made available under this Agreement; and

2. a copy of this Agreement must be included with each copy of the Program.

Contributors may not remove or alter any copyright notices contained within the Program.

Each Contributor must identify itself as the originator of its Contribution, if any, in a manner that reasonably allows subsequent Recipients to identify the originator of the Contribution.

B.4. Commercial Distribution

Commercial distributors of software may accept certain responsibilities with respect to end users, business partners and the like. While this license is intended to facilitate the commercial use of the Program, the Contributor who includes the Program in a commercial product offering should do so in a manner which does not create potential liability for other Contributors. Therefore, if a Contributor includes the Program in a commercial product offering, such Contributor ("Commercial Contributor") hereby agrees to defend and indemnify every other Contributor ("Indemnified Contributor") against any losses, damages and costs (collectively "Losses") arising from claims, lawsuits and other legal actions brought by a third party against the Indemnified Contributor to the extent caused by the acts or omissions of such Commercial Contributor in connection with its distribution of the Program in a commercial product offering. The obligations in this section do not apply to any claims or Losses relating to any actual or alleged intellectual property infringement. In order to qualify, an Indemnified

Contributor must: a) promptly notify the Commercial Contributor in writing of such claim, and b) allow the Commercial Contributor to control, and cooperate with the Commercial Contributor in, the defense and any related settlement negotiations. The Indemnified Contributor may participate in any such claim at its own expense.

For example, a Contributor might include the Program in a commercial product offering, Product X. That Contributor is then a Commercial Contributor. If that Commercial Contributor then makes performance claims, or offers warranties related to Product X, those performance claims and warranties are such Commercial Contributor's responsibility alone. Under this section, the Commercial Contributor would have to defend claims against the other Contributors related to those performance claims and warranties, and if a court requires any other Contributor to pay any damages as a result, the Commercial Contributor must pay those damages.

B.5. No Warranty

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, THE PROGRAM IS PROVIDED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES OR CONDITIONS OF TITLE, NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Each Recipient is solely responsible for determining the appropriateness of using and distributing the Program and assumes all risks associated with its exercise of rights under this Agreement, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

B.6. Disclaimer of Liability

EXCEPT AS EXPRESSLY SET FORTH IN THIS AGREEMENT, NEITHER RECIPIENT NOR ANY CONTRIBUTORS SHALL HAVE ANY LIABILITY FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING WITHOUT LIMITATION LOST PROFITS), HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OR DISTRIBUTION OF THE PROGRAM OR THE EXERCISE OF ANY RIGHTS GRANTED HEREUNDER, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

B.7. General

If any provision of this Agreement is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this Agreement, and without further action by the parties hereto, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

If Recipient institutes patent litigation against a Contributor with respect to a patent applicable to software (including a cross-claim or counterclaim in a lawsuit), then any patent licenses granted by that Contributor to such Recipient under this Agreement shall terminate as of the date such litigation is filed. In addition, if Recipient institutes patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Program itself (excluding combinations of the Program with other software or hardware) infringes such Recipient's patent(s), then such Recipient's rights granted under Section 2(b) shall terminate as of the date such litigation is filed.

All Recipient's rights under this Agreement shall terminate if it fails to comply with any of the material terms or conditions of this Agreement and does not cure such failure in a reasonable period of time after becoming aware of such noncompliance. If all Recipient's rights under this Agreement terminate, Recipient agrees to cease use and distribution of the Program as soon as reasonably practicable.

However, Recipient's obligations under this Agreement and any licenses granted by Recipient relating to the Program shall continue and survive.

Everyone is permitted to copy and distribute copies of this Agreement, but in order to avoid inconsistency the Agreement is copyrighted and may only be modified in the following manner. The Agreement Steward reserves the right to publish new versions (including revisions) of this Agreement from time to time. No one other than the Agreement Steward has the right to modify this Agreement. IBM is the initial Agreement Steward. IBM may assign the responsibility to serve as the Agreement Steward to a suitable separate entity. Each new version of the Agreement will be given a distinguishing version number. The Program (including Contributions) may always be distributed subject to the version of the Agreement under which it was received. In addition, after a new version of the Agreement is published, Contributor may elect to distribute the Program (including its Contributions) under the new version. Except as expressly stated in Sections 2(a) and 2(b) above, Recipient receives no rights or licenses to the intellectual property of any Contributor under this Agreement, whether expressly, by implication, estoppel or otherwise. All rights in the Program not expressly granted under this Agreement are reserved.

This Agreement is governed by the laws of the State of New York and the intellectual property laws of the United States of America. No party to this Agreement will bring a legal action under this Agreement more than one year after the cause of action arose. Each party waives its rights to a jury trial in any resulting litigation.

Appendix C. Revision History

Revision 0-0   Aug 2016

Initial creation for 5.0

Index