

ooRexx

Documentation 5.2.0

Open Object Rexx

Rexx Extensions Library Reference



ooRexx Documentation 5.2.0 Open Object Rexx Rexx Extensions Library Reference Edition 2026.04.18 (last revised on 2026-04-17 with r13151)

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

Copyright © 2005-2026 Rexx Language Association. All rights reserved.

Portions Copyright © 1995, 2004 IBM Corporation and others. All rights reserved.

This documentation and accompanying materials are made available under the terms of the Common Public License v1.0 which accompanies this distribution. A copy is also available as an appendix to this document and at the following address: <https://www.oorexx.org/license.html>.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Rexx Language Association nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) 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 OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Preface	vi
1. Document Conventions	vi
1.1. Typographic Conventions	vi
1.2. Notes and Warnings	vi
2. How to Read the Syntax Diagrams	vii
3. Getting Help and Submitting Feedback	viii
3.1. The Open Object Rexx SourceForge Site	viii
3.2. The Rexx Language Association Mailing List	ix
4. Related Information	x
 1. csvStream Class	 1
1.1. Translation of data involved in the csvStream class	1
1.2. Methods The csvStream Class defines	1
1.3. Attributes of the csvStream Class	1
1.4. Methods Inherited from the Stream Class	1
1.5. Methods inherited from the Object class	2
1.6. Methods	2
1.6.1. Close Method	2
1.6.2. CSVLineIn Method	2
1.6.3. CSVLineOut Method	3
1.6.4. GetHeaders Method	3
1.6.5. INIT Method	3
1.6.6. OPEN Method	4
1.6.7. SetHeaders Method	5
1.7. Attributes	5
1.7.1. DELIMITER Attribute	5
1.7.2. HEADERS Attribute	5
1.7.3. QUALIFIER Attribute	5
1.7.4. SKIPHEADERS Attribute	5
1.8. Examples	5
 2. Host Emulator (HostEmu)	 7
2.1. EXECIO subcommand	8
2.1.1. Command Options	8
2.1.2. Return codes	9
2.2. HI subcommand	9
2.3. TE subcommand	9
2.4. TS subcommand	9
 3. JSON Support (Package "json.cls")	 10
3.1. Json Class	11
3.1.1. false (Class Attribute)	11
3.1.2. fromJSON	12
3.1.3. fromJsonFile (Class Method)	12
3.1.4. init	12
3.1.5. *NEW* jsonToXml (Class Method)	12
3.1.6. *NEW* jsonToXmlFile (Class Method)	13
3.1.7. *NEW* parseXml	13
3.1.8. *NEW* parseXmlFile	13
3.1.9. toJSON	14
3.1.10. toJsonFile (Class Method)	14
3.1.11. true (Class Attribute)	15
3.2. JsonBoolean Class	15
3.2.1. *NEW* activate (Class Method)	15
3.2.2. *NEW* compareTo	15

3.2.3. *NEW* = (Equal)	16
3.2.4. false (Class Attribute)	16
3.2.5. *NEW* init	16
3.2.6. *NEW* makeJSON	16
3.2.7. *NEW* makeString	16
3.2.8. *NEW* \= (Not Equal)	17
3.2.9. true (Class Attribute)	17
3.2.10. *NEW* unknown	17
3.2.11. *NEW* value	17
3.3. *NEW* jsonString Class	17
3.3.1. *NEW* makeJSON	18
3.4. *NEW* JsonError Class	18
3.4.1. *NEW* column (Attribute)	18
3.4.2. *NEW* contextLine (Attribute)	18
3.4.3. *NEW* init	18
3.4.4. *NEW* lineNumber (Attribute)	19
3.4.5. *NEW* makeString	19
3.4.6. *NEW* message (Attribute)	19
3.5. *NEW* JsonXmlEmitter Class	19
3.5.1. *NEW* init	20
3.5.2. *NEW* emit	20
3.6. *NEW* JsonXmlParser Class	20
3.6.1. *NEW* parse	20
3.7. *NEW* string2json Public Routine	21
4. YAML Support (Package "yaml.cls")	22
4.1. *NEW* Yaml Class	23
4.1.1. *NEW* anchorMap (Attribute)	24
4.1.2. *NEW* false (Class Attribute)	24
4.1.3. *NEW* init (Class Method)	24
4.1.4. *NEW* mergeSourceMap (Attribute)	24
4.1.5. *NEW* directivesMap (Attribute)	24
4.1.6. *NEW* init	25
4.1.7. *NEW* parseAll	25
4.1.8. *NEW* parseAllFile	26
4.1.9. *NEW* parseArray	26
4.1.10. *NEW* parseFile	26
4.1.11. *NEW* parseFrontMatter	26
4.1.12. *NEW* parseFrontMatterFile	26
4.1.13. *NEW* parseString	27
4.1.14. *NEW* parseXml	27
4.1.15. *NEW* parseXmlFile	27
4.1.16. *NEW* resolveTagHandle	27
4.1.17. *NEW* readFileLines	28
4.1.18. *NEW* toYaml (Class Method)	28
4.1.19. *NEW* toYamlAll (Class Method)	29
4.1.20. *NEW* toYamlArray (Class Method)	29
4.1.21. *NEW* toYamlFile (Class Method)	30
4.1.22. *NEW* toYamlFMFile (Class Method)	30
4.1.23. *NEW* true (Class Attribute)	31
4.1.24. *NEW* yamlToXml (Class Method)	31
4.1.25. *NEW* yamlToXmlFile (Class Method)	32
4.2. *NEW* YamlBoolean Class	32
4.2.1. *NEW* compareTo	33

4.2.2. *NEW* = (Equal)	33
4.2.3. *NEW* false (Class Attribute)	33
4.2.4. *NEW* init	33
4.2.5. *NEW* makeString	33
4.2.6. *NEW* makeYAML	34
4.2.7. *NEW* != (Not Equal)	34
4.2.8. *NEW* true (Class Attribute)	34
4.2.9. *NEW* unknown	34
4.2.10. *NEW* value	34
4.3. *NEW* YamlTagged Class	35
4.3.1. *NEW* init	35
4.3.2. *NEW* makeString	35
4.3.3. *NEW* tag (Attribute)	35
4.3.4. *NEW* value (Attribute)	36
4.4. *NEW* YamlError Class	36
4.4.1. *NEW* column (Attribute)	36
4.4.2. *NEW* contextLine (Attribute)	36
4.4.3. *NEW* init	36
4.4.4. *NEW* lineNumber (Attribute)	37
4.4.5. *NEW* makeString	37
4.4.6. *NEW* message (Attribute)	37
4.5. *NEW* YamlEmitter Class	37
4.5.1. *NEW* emit	37
4.5.2. *NEW* init	38
4.6. *NEW* YamlXmlEmitter Class	38
4.6.1. *NEW* emit	39
4.6.2. *NEW* init	39
4.7. *NEW* YamlXmlParser Class	40
4.7.1. *NEW* directivesMap (Attribute)	40
4.7.2. *NEW* mergeSourceMap (Attribute)	40
4.7.3. *NEW* xmlAnchors (Attribute)	40
4.7.4. *NEW* parse	40
4.8. *NEW* yaml.deepEqual Public Routine	41
A. Notices	42
A.1. Trademarks	42
A.2. Source Code For This Document	43
B. Common Public License Version 1.0	44
B.1. Definitions	44
B.2. Grant of Rights	44
B.3. Requirements	45
B.4. Commercial Distribution	45
B.5. No Warranty	46
B.6. Disclaimer of Liability	46
B.7. General	46
C. Revision History	48
Index	49

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.

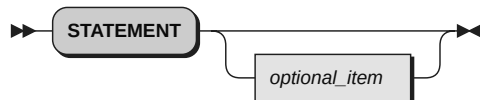
The  symbol indicates that a statement is continued from the previous line.

The  symbol indicates the end of a statement.

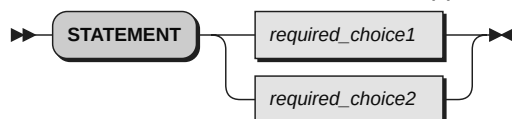
- Required items appear on the horizontal line (the main path).



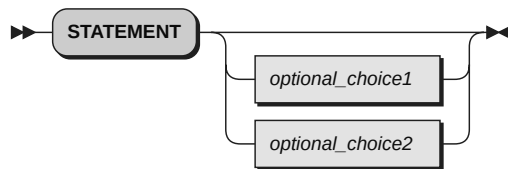
- Optional items appear below the main path.



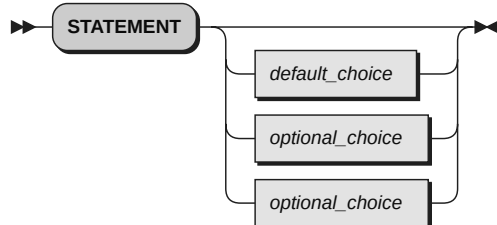
- If you can choose from two or more items, they appear vertically, in a stack. If you must choose one of the items, one item of the stack appears on the main path.



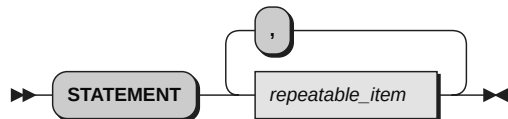
- If choosing one of the items is optional, the entire stack appears below the main path.



- 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.



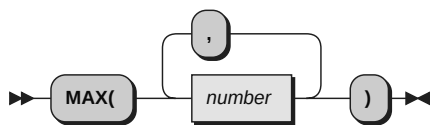
A repeat path above a stack indicates that you can repeat the items in the stack.

- 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://sourceforge.net/projects/oorexx/lists/oorexx-devel> to discuss ooRexx project development activities and future interpreter enhancements. You can find its archive of past messages at <https://sourceforge.net/p/oorexx/mailman/oorexx-devel>.

The Users Mailing List

Subscribe to the oorexx-users mailing list at <https://sourceforge.net/projects/oorexx/lists/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://sourceforge.net/projects/oorexx/lists/oorexx-announce> to receive announcements of significant ooRexx project events.

The Bug Mailing List

Subscribe to the oorexx-bugs mailing list at <https://sourceforge.net/projects/oorexx/lists/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 suggest new 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 <https://groups.io/g/rexxla-members/topics>.

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



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



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.

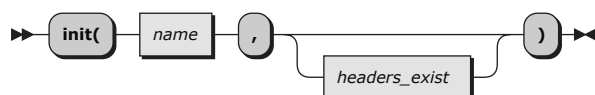
1.6.4. GetHeaders Method



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'

```

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
```

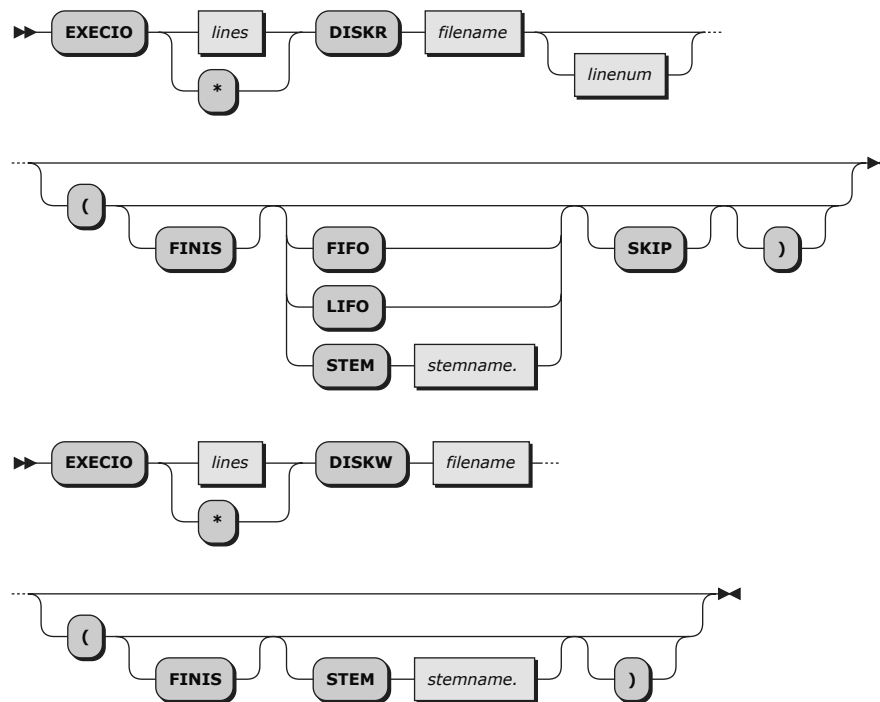
This will activate the environment. The subcommand name is "HostEmu" (the case of this string is not important). You can send commands to this environment via the ooRexx address statement. Here is an example.

```
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.

2.1. EXECIO subcommand



2.1.1. 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.

2.1.2. 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

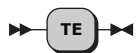
The variable name specified with the STEM option was invalid, or could not be read, or written to.

2.2. HI subcommand



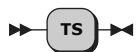
Halts the current Rexx program.

2.3. TE subcommand



Deactivate the Rexx trace mechanism.

2.4. TS subcommand



Activate the Rexx trace mechanism.

JSON Support (Package "json.cls")

The `json.cls` package provides a JSON encoder and decoder for ooRexx, conforming to RFC 8259 (<https://www.rfc-editor.org/rfc/rfc8259>). It encodes ooRexx objects as JSON text and decodes JSON text back into ooRexx objects. Any JSON value (object, array, string, number, boolean, null) is accepted at the top level, as permitted by RFC 8259.

The package defines the following public classes and routines:

Json Class

The main encoder/decoder class. Provides both class methods for convenience and instance methods for full control.

JsonBoolean Class

Sentinel class for JSON boolean values. Two singletons represent **true** and **false**.

**NEW* JsonString Class*

Subclass of **String** that forces numeric values to be encoded as JSON strings rather than JSON numbers.

**NEW* JsonError Class*

Represents a JSON parsing error, carrying the error message, source location, and offending input line.

**NEW* string2json Public Routine*

Public routine that encodes a Rexx string as a JSON value and appends it to a **MutableBuffer**.

Encoding (ooRexx → JSON): **MapCollection** objects are encoded as JSON objects (keys sorted alphabetically), **OrderedCollection** objects as JSON arrays, **.nil** as **null**, **JsonBoolean** singletons as **true** or **false**, **JsonString** values as quoted strings (even when numeric), and plain strings as numbers when numeric or as quoted strings otherwise. Objects that provide a `makeJSON`, `makeArray`, or `makeString` method are handled transparently.

Decoding (JSON → ooRexx): JSON objects are decoded as **Directory** objects, JSON arrays as **Array** objects, JSON strings as **JsonString** instances, JSON numbers as plain ooRexx strings, JSON booleans as **JsonBoolean** singletons, and JSON null as **.nil**.

\uXXXX handling: Since ooRexx has no native Unicode support, **\u00XX** escape sequences are decoded to the corresponding single-byte character. Non-**\u00XX** sequences (e.g. **\u4E16**) are kept as literal **\uXXXX** text. During encoding, any literal **\uXXXX** pattern already present in a source string is passed through unchanged, so that a Rexx string containing the six characters **\u0041** will encode as **\u0041** (decoded as **A** by a JSON consumer), rather than escaping the backslash.

Quick start – Parse a JSON string and re-encode it:

```
json = .json~new
doc = json~fromJSON('{"name": "Alice", "age": 30, "active": true}')
say doc["name"]           -- Alice
say doc["age"]            -- 30
say doc["active"]         -- 1 (a JsonBoolean singleton)
say .json~toJSON(doc)     -- {"active":true,"age":30,"name":"Alice"}

::requires "json.cls"
```

Example 1 – Build an ooRexx tree and emit JSON:

```
doc = .directory~new
```

```

doc["title"] = "My Application"
doc["version"] = .jsonString-new("2") -- force quoted
authors = .array-of("Alice", "Bob")
doc["authors"] = authors
say .json-toJSON(doc, .true)
-- Output:
-- {
--   "authors": [
--     "Alice",
--     "Bob"
--   ],
--   "title": "My Application",
--   "version": "2"
-- }

::requires "json.cls"

```

Example 2 – Read a JSON file and write it back:

```

doc = .json~fromJsonFile("config.json")
say doc["title"]
.json~toJsonFile("config_pretty.json", doc, .true)

::requires "json.cls"

```

Example 3 – Round-trip with booleans:

```

json = .json-new
doc = json~fromJSON('{"active": true, "deleted": false}')
say doc["active"]~isA(.JsonBoolean) -- 1
say doc["active"] -- 1
say .json~toJSON(doc)
-- Output: {"active":true,"deleted":false}

::requires "json.cls"

```

3.1. Json Class

The **Json** class is the main entry point of the **json.cls** package. Encoding and decoding are available both as class methods (convenience wrappers that create a temporary instance) and as instance methods. Each instance initialises its own escape/unescape tables and maintains internal parsing state.

Table 3.1. Json Class

Object		
Methods inherited from the Object class		
Json		
<i>false (Class Attribute)</i>	<i>init</i>	<i>true (Class Attribute)</i>
<i>fromJSON</i>	<i>toJSON</i>	
<i>fromJsonFile (Class Method)</i>	<i>toJsonFile (Class Method)</i>	

3.1.1. false (Class Attribute)



Returns the **JsonBoolean** proxy for **.false**. It can be used interchangeably with ooRexx' **.false** or **0** values.

3.1.2. fromJSON



Decodes a JSON text into an ooRexx object tree. JSON objects become **Directory** instances, JSON arrays become **Array** instances, JSON strings become **JsonString** instances, JSON numbers become plain ooRexx strings, JSON booleans become **JsonBoolean** singletons, and JSON null becomes **.nil**.

This method is available both as an instance method and as a class method. When called as a class method, a temporary instance is created internally.

Numbers are validated against the RFC 8259 grammar: leading plus signs, leading zeros, trailing dots, and leading dots are rejected. Strings are validated per RFC 8259 §7: raw control characters U+0000 through U+001F inside quoted strings are rejected (they must be escaped using **\uXXXX** notation). Invalid JSON raises a **Syntax** condition with a **JsonError** description in the additional information.

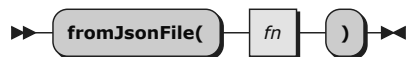
Arguments:

jsonString

A JSON text string.

Returns: the ooRexx object tree representing the decoded JSON.

3.1.3. fromJsonFile (Class Method)



Convenience class method that reads a JSON file and decodes it into an ooRexx object tree. Opens the file, reads its entire contents, creates a **Json** instance, and delegates to **fromJSON**.

Arguments:

fn

The file name or file object to read JSON data from.

Returns: the ooRexx object tree representing the decoded JSON.

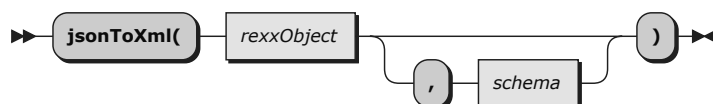
3.1.4. init



Creates a new **Json** instance and initialises the escape/unescape lookup tables, the set of value-terminating control characters, and the platform's end-of-line sequence. The lookup tables are also stored as package-local variables so that the **string2json** routine can access them.

Takes no arguments.

3.1.5. *NEW* jsonToXml (Class Method)



Serialises an ooRexx object tree to XML conforming to either **json.xsd** (with namespace **urn:json:xml:1.0**) or **json.dtd** (with DOCTYPE, no namespace).

Internally creates a **JsonXmlEmitter** instance and calls its `emit` method. The emitter sorts object keys alphabetically for deterministic output, matching the behaviour of `toJson`.

Arguments:

rexxObject

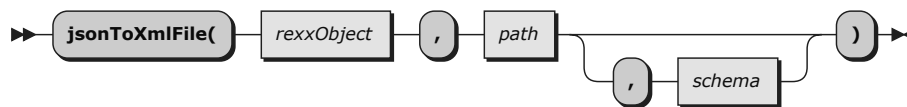
The ooRexx object tree to serialise. Any JSON-compatible value is accepted: **Directory**, **Array**, **String**, **JsonString**, **JsonBoolean**, or **.nil**.

schema

Optional schema type: "**xsd**" (default) or "**dtd**".

Returns: a well-formed XML string.

3.1.6. *NEW* jsonToXmlFile (Class Method)



Serialises an ooRexx object tree to XML and writes it to a file. Calls `jsonToXml` internally and writes the result using **Stream** with **charOut** to ensure platform-independent output.

Arguments:

rexxObject

The ooRexx object tree to serialise.

path

The file system path to write.

schema

Optional schema type: "**xsd**" (default) or "**dtd**".

3.1.7. *NEW* parseXml



Parses an XML string conforming to **json.xsd** or **json.dtd** back into an ooRexx object tree. This is an instance method. Internally creates a **JsonXmlParser** and delegates to its `parse` method.

Type preservation is maintained: **<boolean>** elements produce **JsonBoolean** instances, **<string>** elements produce **JsonString** instances, **<null/>** produces **.nil**, and **<number>** elements produce plain numeric strings.

Arguments:

input

An XML string or an **Array** of lines.

Returns: the ooRexx object tree.

3.1.8. *NEW* parseXmlFile



Reads an XML file and parses it into an ooRexx object tree. Opens the file, reads all lines into an **Array**, and delegates to `parseXml`.

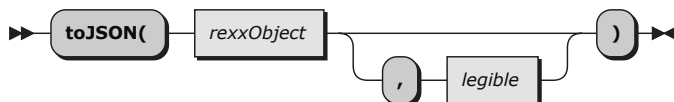
Arguments:

path

The file system path to read.

Returns: the ooRexx object tree.

3.1.9. toJSON



Encodes an ooRexx object as a JSON text string. The encoder dispatches based on object type: **String** objects are encoded via the `string2json` routine, **OrderedCollection** objects as JSON arrays, **MapCollection** objects as JSON objects (keys sorted alphabetically for reproducible output), **.nil** as **null**, and objects with a `makeJSON` method (including **JsonBoolean**) via that method. Objects with `makeArray` or `makeString` methods are handled as fallbacks.

This method is available both as an instance method and as a class method. When called as a class method, a temporary instance is created internally.

Arguments:

rexxObject

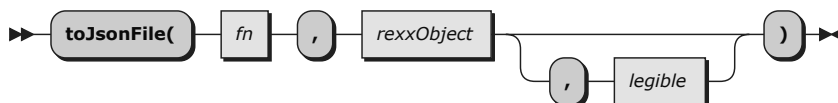
The ooRexx object to encode.

legible

Optional logical value (default **.false**). When **.true**, the output includes ignorable whitespace (newlines and 3-space indentation) for human readability. When **.false**, the output is minimised (no extra whitespace).

Returns: a JSON text string.

3.1.10. toJsonFile (Class Method)



Convenience class method that encodes an ooRexx object as JSON and writes it to a file. Creates a **Json** instance, delegates to `toJSON`, and writes the result to the specified file.

Arguments:

fn

The file name or file object to write the produced JSON text to.

rexxObject

The ooRexx object to encode.

legible

Optional logical value (default **.false**). When **.true**, the output includes ignorable whitespace for human readability.

3.1.11. true (Class Attribute)



Returns the **JsonBoolean** proxy for **.true**. It can be used interchangeably with ooRexx' **.true** or **1** values.

3.2. JsonBoolean Class

Sentinel class for JSON boolean values. Two singleton instances (**.JsonBoolean~true** and **.JsonBoolean~false**) are created at class activation time. They behave transparently as **1** and **0** via **makeString** and the **unknown** method, but can be detected with **isA(.JsonBoolean)** for type-aware serialisation.

It is advised to use the **Json** class attributes **.Json~true** and **.Json~false** as convenient accessors to the singleton instances.

The **JsonBoolean** class inherits from the **Comparable** mixin class.

Table 3.2. JsonBoolean Class

Object		
Methods inherited from the Object class		
Comparable (Mixin Class)		
Methods inherited from the Comparable class		
JsonBoolean		
<i>*NEW* activate (Class Method)</i>	<i>*NEW* init</i>	<i>true (Class Attribute)</i>
<i>*NEW* compareTo</i>	<i>*NEW* makeJSON</i>	<i>*NEW* unknown</i>
<i>*NEW* = (Equal)</i>	<i>*NEW* makeString</i>	<i>*NEW* value</i>
<i>false (Class Attribute)</i>	<i>*NEW* != (Not Equal)</i>	

3.2.1. *NEW* activate (Class Method)



Finalises the class initialisation by creating the two singleton class attribute values **true** and **false**. This method is called automatically when the class is loaded and does not normally need to be invoked directly.

3.2.2. *NEW* compareTo



Implements the abstract method inherited from the mixinclass **Comparable**. If *other* is a **JsonBoolean**, its logical value is obtained via the **value** method; otherwise, the string representation is requested. If the other object has no **makeString** method, a **Syntax** condition is raised.

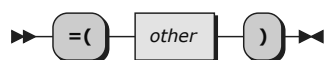
Arguments:

other

The other object representing a boolean/logical value.

Returns: **-1** if this value is less than *other*, **0** if equal, **1** if greater.

3.2.3. *NEW* = (Equal)



Equal comparison operator.

Arguments:

other

The other object representing a boolean/logical value.

Returns: `.true` if this object and *other* are equal, `.false` otherwise.

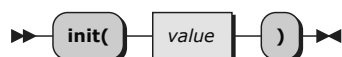
3.2.4. false (Class Attribute)



Class getter attribute that returns the singleton representing `.false`.

Returns: the `JsonBoolean` proxy for `.false`.

3.2.5. *NEW* init



Constructor that stores the logical value. Note that the new class method is private; only the class itself can create instances via the `activate` method.

Arguments:

value

A Rexx logical value (`0` or `1`).

3.2.6. *NEW* makeJSON



Renders this object as a JSON string representing its logical value.

Returns: `"true"` or `"false"`.

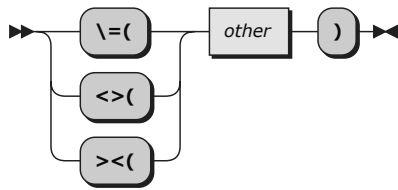
3.2.7. *NEW* makeString



Renders this object as a Rexx string representing its logical value, allowing instances of this class to be used as plug-in replacements for Rexx logical values.

Returns: `"1"` or `"0"`.

3.2.8. *NEW* \= (Not Equal)



Unequal comparison operator. The synonyms `<>` and `><` forward to this method.

Arguments:

other

The other object representing a boolean/logical value.

Returns: `.true` if this object and *other* are not equal, `.false` otherwise.

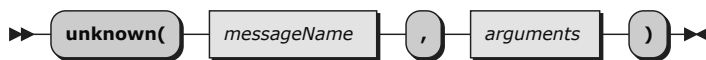
3.2.9. true (Class Attribute)



Class getter attribute that returns the singleton representing `.true`.

Returns: the **JsonBoolean** proxy for `.true`.

3.2.10. *NEW* unknown



Forwards any unrecognised message to the underlying string value, so that **JsonBoolean** can be used transparently wherever a Rexx logical value is expected.

Arguments:

messageName

The name of the unrecognised message.

arguments

The arguments array.

3.2.11. *NEW* value



Returns the underlying Rexx logical value.

Returns: `1` for true, `0` for false.

3.3. *NEW* JsonString Class

A subclass of **String** that forces numeric values to be encoded as JSON strings rather than JSON numbers. When the encoder encounters a **JsonString**, it always quotes the value, even if it is numeric. For example, `.JsonString~new("123")` encodes as `"123"` rather than `123`.

Strings decoded from JSON are returned as **JsonString** instances, preserving their identity as JSON string values through round-trips.

Since **JsonString** is a subclass of **String**, it inherits all string methods and can be used transparently wherever a Rexx string is expected.

Table 3.3. JsonString Class

String
Methods inherited from the String class
JsonString
<i>*NEW* makeJSON</i>

3.3.1. *NEW* makeJSON



Encodes this string as a quoted JSON value using the `string2json` routine, regardless of whether the content is numeric.

Returns: a JSON-encoded quoted string.

3.4. *NEW* JsonError Class

Represents an error encountered during JSON parsing. Instances are raised via **Raise Syntax** and carry the error message, source location, and the offending input line for diagnostic purposes.

Table 3.4. JsonError Class

Object		
Methods inherited from the Object class		
JsonError		
<i>*NEW* column (Attribute)</i>	<i>*NEW* init</i>	<i>*NEW* makeString</i>
<i>*NEW* contextLine (Attribute)</i>	<i>*NEW* lineNumber (Attribute)</i>	<i>*NEW* message (Attribute)</i>

3.4.1. *NEW* column (Attribute)



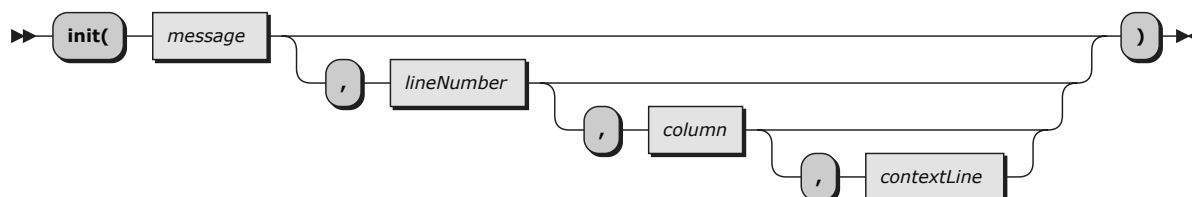
The 1-based column where the error was detected (0 if unknown). This is a read-write attribute.

3.4.2. *NEW* contextLine (Attribute)



The source line that triggered the error (empty string if unavailable). This is a read-write attribute.

3.4.3. *NEW* init



Creates a new **JsonError** instance.

Arguments:

message

The error description.

lineNumber

Optional 1-based line number (default 0).

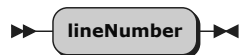
column

Optional 1-based column (default 0).

contextLine

Optional source line text (default "").

3.4.4. *NEW* lineNumber (Attribute)



The 1-based line number where the error was detected (0 if unknown). This is a read-write attribute.

3.4.5. *NEW* makeString



Returns a human-readable representation of the error, including the line number, column, and context line when available. The format is:

```
JsonError: message (line N, column M)  
    > contextLine
```

Returns: a formatted error string.

3.4.6. *NEW* message (Attribute)



The human-readable error description. This is a read-write attribute.

3.5. *NEW* JsonXmlEmitter Class

Serialises an in-memory ooRexx object tree to XML conforming to either **json.xsd** (with namespace **urn:json:xml:1.0**) or **json.dtd** (with DOCTYPE, no namespace).

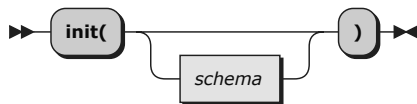
This class is used internally by `Json~jsonToXml` and `Json~jsonToXmlFile`. It can also be used directly if more control over the emission process is needed.

The XML vocabulary maps JSON types to elements as follows:

JSON type	XML element	Content
root	<json>	one value element
object	<object>	<entry> children
(entry)	<entry>	<n> + <value>
array	<array>	<item> children

JSON type	XML element	Content
string	<string>	text (XML-escaped)
number	<number>	text
boolean	<boolean>	true or false
null	<null/>	self-closing

3.5.1. *NEW* init



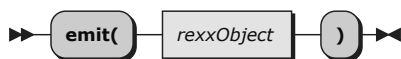
Creates a new **JsonXmlEmitter** instance.

Arguments:

schema

Optional schema type: "**xsd**" (default) or "**dtd**". Determines whether the output includes an XML namespace declaration (XSD) or a DOCTYPE declaration (DTD).

3.5.2. *NEW* emit



Emits the given ooRexx object tree as a well-formed XML string. Object keys are sorted alphabetically for deterministic output.

Arguments:

rexxObject

The root object to serialise.

Returns: a well-formed XML string.

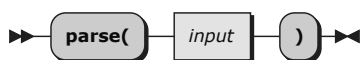
3.6. *NEW* JsonXmlParser Class

Parses XML conforming to **json.xsd** or **json.dtd** back into ooRexx objects. This is a purpose-built parser for the JSON XML vocabulary — it is not a general-purpose XML parser.

This class is used internally by **Json~parseXml** and **Json~parseXmlFile**. It can also be used directly.

Type preservation is maintained during parsing: **<boolean>** elements produce **JsonBoolean** instances, **<string>** elements produce **JsonString** instances, **<number>** elements produce plain numeric strings (normalised via **+ 0**), **<object>** elements produce **Directory** instances, and **<array>** elements produce **Array** instances.

3.6.1. *NEW* parse



Parses an XML string into an ooRexx object tree.

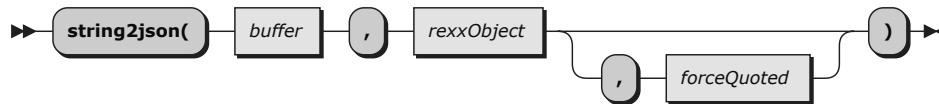
Arguments:

input

A well-formed XML string conforming to the JSON XML vocabulary.

Returns: the ooRexx object tree.

3.7. *NEW* string2json Public Routine



Encodes a Rexx string as a JSON value and appends it to a **MutableBuffer**. If the string is numeric and is a plain **String** (not a **JsonString**), it is appended unquoted as a JSON number, normalised via + 0 (e.g. `.5` becomes `0.5`, `042` becomes `42`). If it is a **JsonString**, the numeric value is quoted. Non-numeric strings are always quoted with proper JSON escaping of control characters, backslashes, and double quotes.

Any `\uXXXX` pattern already present in the source string is passed through unchanged (see the `json.cls` package documentation for the rationale).

Arguments:

buffer

The **MutableBuffer** to append to.

rexxObject

The string to encode.

forceQuoted

Optional; if `.true`, always quote the value regardless of whether it is numeric. This is used when encoding JSON object keys, which must always be quoted strings.

YAML Support (Package "yaml.cls")

The **yaml.cls** package provides a comprehensive YAML 1.2 (core schema) parser and emitter for ooRexx. It covers the vast majority of real-world YAML: block and flow mappings, block and flow sequences, block scalars (literal and folded with chomping), quoted and plain scalars, anchors and aliases, merge keys (<<), multi-document streams, front-matter extraction, and XML round-tripping via **yaml.xsd** and **yaml.dtd**. Merge keys are preserved in the XML representation as dedicated <merge> elements when the mergeSourceMap is provided.

The package defines the following public classes and routines:

**NEW* Yaml Class*

The main parser/emitter class. Parsing is performed through instance methods; serialisation through class methods.

**NEW* YamlBoolean Class*

Sentinel class for YAML boolean values. Two singletons represent **true** and **false**.

**NEW* YamlTagged Class*

Wraps a YAML value together with its explicit tag string. Produced by the parser when the **preserveTags** option is enabled.

**NEW* YamlError Class*

Represents a YAML parsing error, carrying the error message, source location, and offending input line.

**NEW* YamlEmitter Class*

Low-level YAML emitter. Used internally by the **Yaml** class methods, but available for direct use when finer control is needed.

**NEW* YamlXmlEmitter Class*

Serialises an ooRexx object tree to XML conforming to **yaml.xsd** or **yaml.dtd**.

**NEW* YamlXmlParser Class*

Parses XML conforming to the YAML XML vocabulary back into ooRexx objects.

**NEW* yaml.deepEqual Public Routine*

Public routine that recursively compares two ooRexx objects for semantic equality.

Representation model: YAML mappings are represented as **Table** objects, sequences as **Array** objects, null as **.nil**, booleans as **YamlBoolean** singletons, and integers, floats, and strings as plain ooRexx strings. When the **preserveTags** option is enabled, any node that carries an explicit tag in the source YAML is wrapped in a **YamlTagged** instance whose **tag** attribute holds the original tag string and whose **value** attribute holds the resolved object.

Quick start – Parse a YAML file and emit it back:

```
parser = .Yaml~new
doc    = parser~parseFile("config.yml")
say doc["title"]
yaml   = .Yaml~toYaml(doc)

::requires "yaml.cls"
```

Example 1 – Build an ooRexx tree and emit YAML:

```
doc = .table~new
```

```

doc["title"] = "My Application"
doc["version"] = 2
authors = .array-of("Alice", "Bob")
doc["authors"] = authors
say .Yaml~toYaml(doc)
-- Output:
--   title: My Application
--   version: 2
--   authors:
--     - Alice
--     - Bob

::requires "yaml.cls"

```

Example 2 – Convert YAML to XML:

```

parser = .Yaml~new
doc = parser~parseString("name: Alice" || "0a"x || "age: 30")
xml = .Yaml~yamlToXml(doc)
say xml
-- Output (XSD-valid YAML-in-XML representation):
--   <?xml version="1.0" ...?>
--   <yaml xmlns="...">
--     <map>
--       <mapping>
--         <scalar>name</scalar>
--         <scalar>Alice</scalar>
--       </mapping>
--     ...
--   </map>
-- </yaml>

::requires "yaml.cls"

```

Example 3 – Convert XML-encoded YAML back to YAML:

```

parser = .Yaml~new
doc = parser~parseXml(xmlString)  -- xmlString is the XML from Example 2
yaml = .Yaml~toYaml(doc)
say yaml
-- Output:
--   name: Alice
--   age: 30

::requires "yaml.cls"

```

4.1. *NEW* Yaml Class

The **Yaml** class is the main entry point of the **yaml.cls** package. Parsing is performed through instance methods on a **Yaml** object; serialisation is provided as class methods. Each parser instance maintains its own anchor map and merge-source map from the most recent parse, which can be passed to the serialisation methods for lossless round-tripping.

Table 4.1. Yaml Class

Object		
Methods inherited from the Object class		
Yaml		
<i>*NEW* anchorMap (Attribute)</i>	<i>*NEW* parseFile</i>	<i>*NEW* toYamlAll (Class Method)</i>

<i>*NEW* directivesMap (Attribute)</i>	<i>*NEW* parseFrontMatter</i>	<i>*NEW* toYamlArray (Class Method)</i>
<i>*NEW* false (Class Attribute)</i>	<i>*NEW* parseFrontMatterFile</i>	<i>*NEW* toYamlFMFile (Class Method)</i>
<i>*NEW* init</i>	<i>*NEW* parseString</i>	<i>*NEW* toYamlFile (Class Method)</i>
<i>*NEW* init (Class Method)</i>	<i>*NEW* parseXml</i>	<i>*NEW* true (Class Attribute)</i>
<i>*NEW* mergeSourceMap (Attribute)</i>	<i>*NEW* parseXmlFile</i>	<i>*NEW* yamlToXml (Class Method)</i>
<i>*NEW* parseAll</i>	<i>*NEW* resolveTagHandle</i>	<i>*NEW* yamlToXmlFile (Class Method)</i>
<i>*NEW* parseAllFile</i>	<i>*NEW* readFileLines</i>	
<i>*NEW* parseArray</i>	<i>*NEW* toYaml (Class Method)</i>	

4.1.1. *NEW* anchorMap (Attribute)



Returns the anchor map from the most recent parse. The map is an **IdentityTable** that associates each anchored ooRexx object with its anchor name (a string). Pass this to the `toYaml*` and `yamlToXml*` methods to preserve anchors and aliases during round-tripping.

Returns: an **IdentityTable** mapping objects to anchor name strings.

4.1.2. *NEW* false (Class Attribute)



Returns the **YamlBoolean** proxy for `.false`. It can be used interchangeably with ooRexx' `.false` or `0` values.

4.1.3. *NEW* init (Class Method)



Initialises the class-level lookup tables for escape sequences, boolean words, and null words according to the YAML 1.2 core schema. This method is called automatically when the class is loaded and does not normally need to be invoked directly.

4.1.4. *NEW* mergeSourceMap (Attribute)



Returns the merge-source map from the most recent parse. The map is an **IdentityTable** that associates each target mapping (**Table**) that contained merge keys (`<<`) with an **Array** of the source mapping objects that were merged into it. Pass this to the `toYaml*` methods to reconstruct merge keys during round-tripping.

Returns: an **IdentityTable** mapping target **Tables** to **Arrays** of source **Tables**.

4.1.5. *NEW* directivesMap (Attribute)



Returns the directives map from the most recent parse. The map is an **IdentityTable** that associates each parsed document's root object with a **Table** containing its YAML directives. Documents with no directives have no entry in the map.

The directives **Table** can contain two entries:

"yamlVersion"

A string with the version number from a %YAML directive (e.g. "1.2").

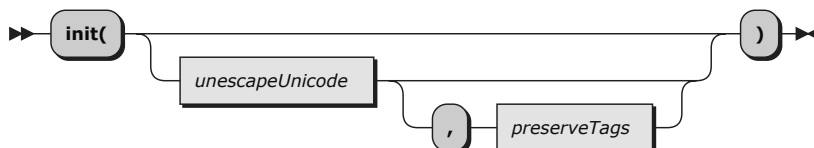
"tagHandles"

A **Table** mapping tag handle strings (e.g. "!!", "!e!") to their URI prefixes, from %TAG directives.

Pass this map to the `toYaml*` and `yamlToXml*` methods to preserve directives during round-tripping. Tags are stored as shorthand during parsing (e.g. `!e!person`); use `resolveTagHandle` to resolve them on demand against the **tagHandles** table.

Returns: an **IdentityTable** mapping root objects to directives **Tables**.

4.1.6. *NEW* init



Creates a new **Yaml** parser instance. Each instance maintains its own parsing state, including anchor, merge-source, and directive metadata from the most recent parse operation.

Arguments:

unescapeUnicode

Optional. When **.true** (the default), `\uXXXX` and `\UXXXXXXXX` escape sequences in double-quoted strings are converted to their UTF-8 byte equivalents during parsing. When **.false**, they are stored literally (e.g. as the six characters `\u00E9`). Use **.false** when exact byte-level round-tripping of escape sequences is required.

preserveTags

Optional. When **.true**, any YAML node that carries an explicit tag (e.g. `!!str`, `!custom`, `!<tag:yaml.org,2002:str>`) is wrapped in a **YamlTagged** object that pairs the original tag string with the resolved value. When **.false** (the default), tags are consumed during type resolution and are not preserved in the object tree.

Returns: a new **Yaml** instance.

4.1.7. *NEW* parseAll



Parses all YAML documents from a multi-document string (separated by --- document-start markers).

Arguments:

input

A YAML string containing one or more documents.

Returns: an **Array** of ooRexx object trees, one per document.

4.1.8. *NEW* parseAllFile



Parses all YAML documents from a multi-document file.

Arguments:

path

The file system path to read.

Returns: an **Array** of ooRexx object trees, one per document.

4.1.9. *NEW* parseArray



Parses a single YAML document from an **Array** of lines.

Arguments:

anArray

An **Array** where each item is one line of YAML text.

Returns: the ooRexx object tree representing the parsed document.

4.1.10. *NEW* parseFile



Parses a single YAML document from a file.

Arguments:

path

The file system path to read.

Returns: the ooRexx object tree representing the parsed document.

4.1.11. *NEW* parseFrontMatter



Extracts and parses the YAML front-matter block from a text that uses the --- / --- convention (for example, Jekyll, Hugo, or Markdown documents).

Arguments:

input

The full text containing front-matter.

Returns: the ooRexx object tree of the front-matter, or **.nil** if no front-matter block is found.

4.1.12. *NEW* parseFrontMatterFile



Extracts and parses YAML front-matter from a file.

Arguments:

path

The file system path to read.

Returns: the ooRexx object tree of the front-matter, or **.nil** if no front-matter block is found.

4.1.13. *NEW* parseString



Parses a single YAML document from a string. If the input contains multiple documents, only the first one is parsed.

Arguments:

input

A YAML string (may be a multi-line string or an **Array** of lines).

Returns: the ooRexx object tree representing the parsed document.

4.1.14. *NEW* parseXml



Parses an XML string conforming to **yaml.xsd** or **yaml.dtd** back into an ooRexx object tree.

Arguments:

input

An XML string or an **Array** of lines.

Returns: the ooRexx object tree.

4.1.15. *NEW* parseXmlFile



Parses an XML file conforming to **yaml.xsd** or **yaml.dtd** back into an ooRexx object tree.

Arguments:

path

The file system path to read.

Returns: the ooRexx object tree.

4.1.16. *NEW* resolveTagHandle



Resolves a YAML tag handle against a tag-handles table obtained from **%TAG** directives. Tags are stored as shorthand during parsing (e.g. **!e!person**, **!!int**); this method performs on-demand (lazy) resolution to the full verbatim form.

For example, with **%TAG !e! tag:example.com,2000:**, calling **parser~resolveTagHandle("!e!person", tagHandles)** returns **!**

`<tag:example.com,2000:person>`. If no matching handle is found, the tag is returned unchanged.

Arguments:

tag

The tag string to resolve (e.g. `"!e!person"`, `"!!int"`).

tagHandles

A **Table** mapping handle strings to URI prefixes, typically obtained from `directivesMap[doc]["tagHandles"]`.

Returns: the resolved tag as a verbatim tag string, or the original tag if no matching handle is found.

4.1.17. *NEW* readFileLines



Reads the contents of a file into an **.Array** of lines. This is a utility method that handles stream opening, reading, and closing, and raises a Syntax error if the file cannot be opened.

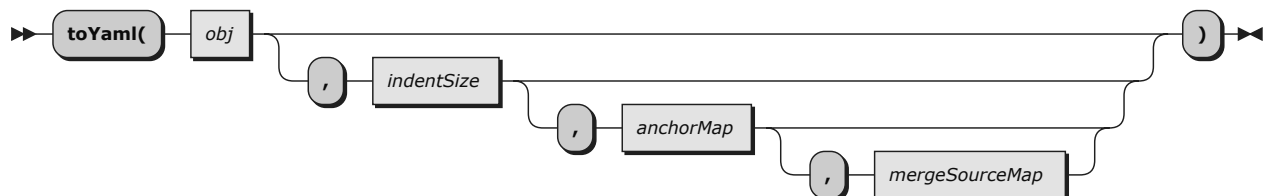
Arguments:

path

The file system path to read.

Returns: an **.Array** containing one element per line of the file.

4.1.18. *NEW* toYaml (Class Method)



Serialises an ooRexx object tree to a YAML string.

Arguments:

obj

The ooRexx object tree to serialise.

indentSize

Optional number of spaces per indentation level (default 2).

anchorMap

Optional **IdentityTable** from a previous parse to preserve anchors/aliases.

mergeSourceMap

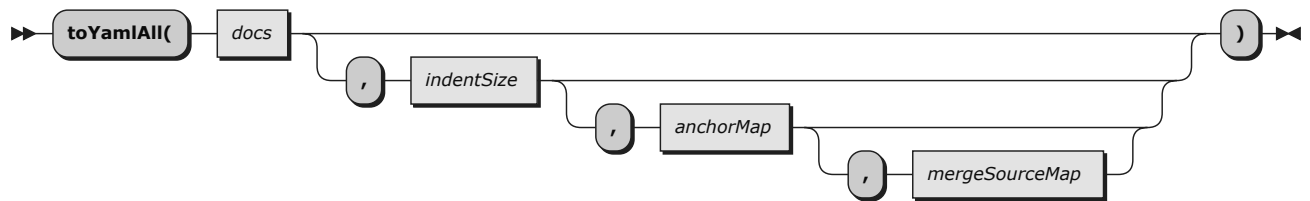
Optional **IdentityTable** from a previous parse to reconstruct merge keys.

directivesMap

Optional **IdentityTable** from a previous parse to preserve **%YAML** and **%TAG** directives. When present, the emitted YAML includes the directives followed by a `---` document-start marker.

Returns: a YAML-formatted string.

4.1.19. *NEW* toYamlAll (Class Method)



Serialises an array of ooRexx object trees to a multi-document YAML string, with each document preceded by a `---` marker.

Arguments:

docs

An **Array** of ooRexx object trees.

indentSize

Optional indentation size (default 2).

anchorMap

Optional anchor map.

mergeSourceMap

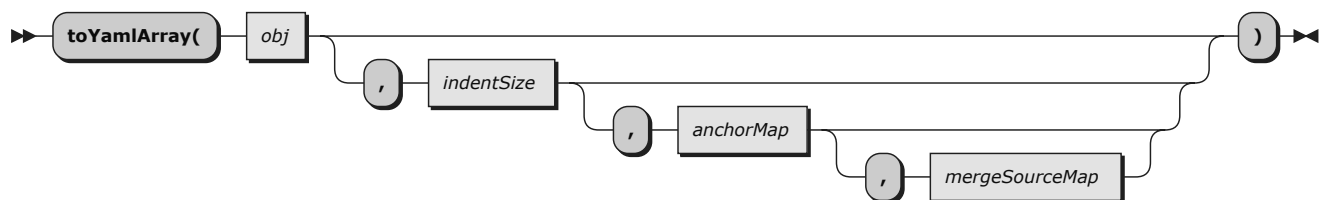
Optional merge-source map.

directivesMap

Optional directives map.

Returns: a multi-document YAML string.

4.1.20. *NEW* toYamlArray (Class Method)



Serialises an ooRexx object tree to an **Array** of YAML lines (one element per line, without trailing newlines).

Arguments:

obj

The ooRexx object tree to serialise.

indentSize

Optional indentation size (default 2).

anchorMap

Optional anchor map.

mergeSourceMap

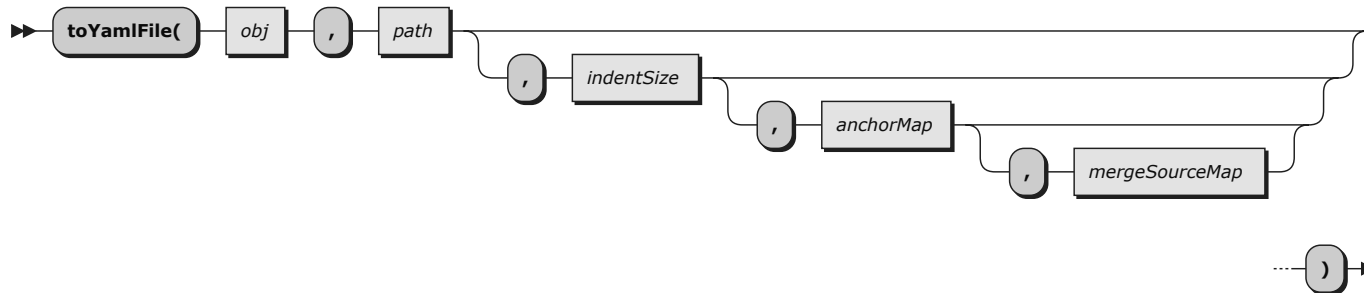
Optional merge-source map.

directivesMap

Optional directives map.

Returns: an **Array** of strings, one per YAML output line.

4.1.21. *NEW* toYamlFile (Class Method)



Serialises an ooRexx object tree and writes it to a YAML file using platform-independent LF line endings.

Arguments:

obj

The ooRexx object tree to serialise.

path

The file system path to write.

indentSize

Optional indentation size (default 2).

anchorMap

Optional anchor map.

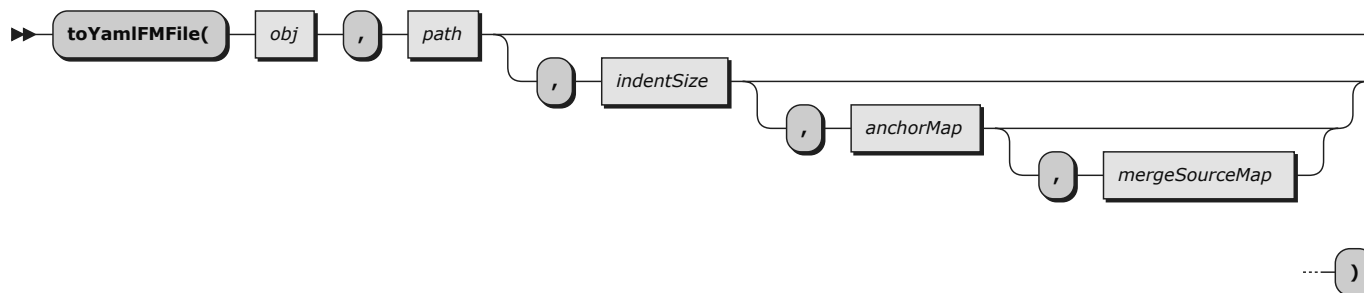
mergeSourceMap

Optional merge-source map.

directivesMap

Optional directives map.

4.1.22. *NEW* toYamlFMFile (Class Method)



Serialises an ooRexx object tree and writes it to a file wrapped in front-matter delimiters (--- before and after), suitable for Markdown, Jekyll, or Hugo documents.

Arguments:

obj

The ooRexx object tree to serialise.

path

The file system path to write.

indentSize

Optional indentation size (default 2).

anchorMap

Optional anchor map.

mergeSourceMap

Optional merge-source map.

directivesMap

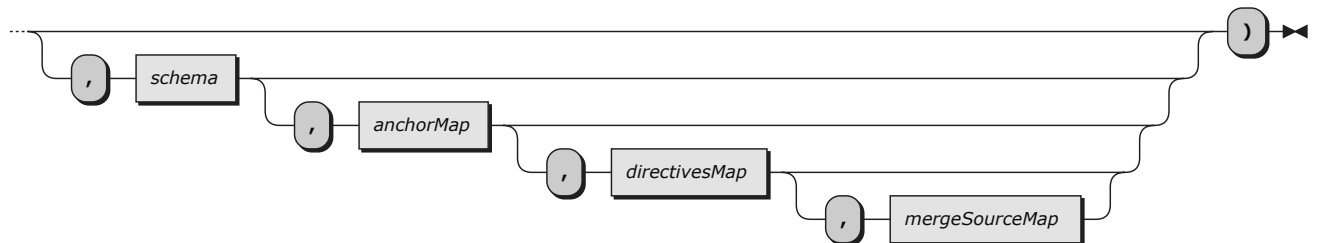
Optional directives map.

4.1.23. *NEW* true (Class Attribute)



Returns the **YamlBoolean** proxy for **.true**. It can be used interchangeably with ooRexx' **.true** or **1** values.

4.1.24. *NEW* yamlToXml (Class Method)



Serialises an ooRexx object tree to an XML string conforming to either **yaml.xsd** (with namespace) or **yaml.dtd** (with DOCTYPE).

Arguments:*obj*

The ooRexx object tree to serialise.

schema

Optional schema type: **"xsd"** (default) or **"dtd"**.

anchorMap

Optional anchor map from a previous parse.

directivesMap

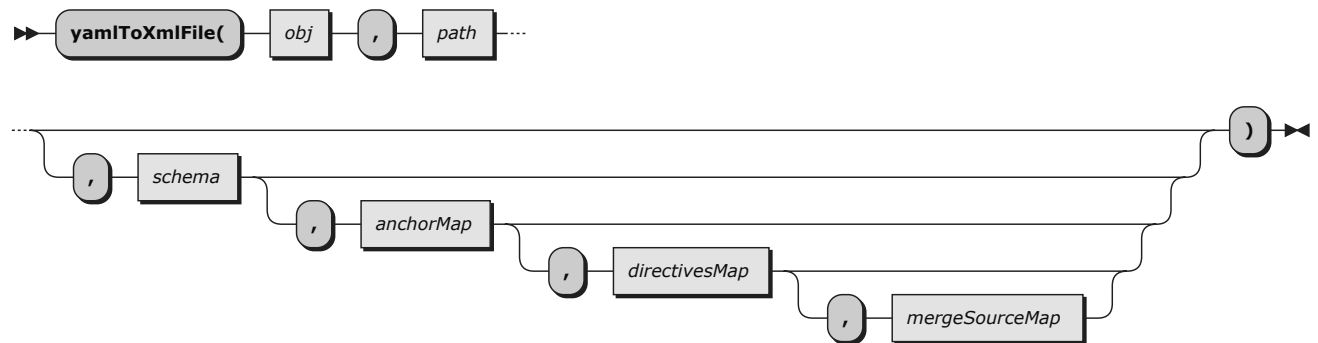
Optional directives map. When present, the XML output includes a **yaml-version** attribute on the **<document>** element and **<tag-directive>** child elements for each **%TAG** handle.

mergeSourceMap

Optional merge-source map. When present, mappings that had merge keys (**<<**) emit **<merge anchor="...">** elements instead of expanding the merged keys as regular entries.

Returns: an XML string.

4.1.25. *NEW* yamlToXmlFile (Class Method)



Serialises an ooRexx object tree to XML and writes it to a file.

Arguments:

obj

The ooRexx object tree to serialise.

path

The file system path to write.

schema

Optional schema type: "xsd" (default) or "dtd".

anchorMap

Optional anchor map from a previous parse.

directivesMap

Optional directives map.

mergeSourceMap

Optional merge-source map.

4.2. *NEW* YamlBoolean Class

Sentinel class for YAML boolean values, following the same pattern as **JsonBoolean** in `json.cls`. Two singleton instances (`.YamlBoolean~true` and `.YamlBoolean~false`) are created at class activation time. They behave transparently as **1** and **0** via `makeString` and the `unknown` method, but can be detected with `isa( .YamlBoolean )` for type-aware serialisation.

The **YamlBoolean** class inherits from the **Comparable** mixin class.

Table 4.2. YamlBoolean Class

Object		
Methods inherited from the Object class		
Comparable (Mixin Class)		
Methods inherited from the Comparable class		
YamlBoolean		
<i>*NEW* compareTo</i>	<i>*NEW* makeString</i>	<i>*NEW* unknown</i>
<i>*NEW* = (Equal)</i>	<i>*NEW* makeYAML</i>	<i>*NEW* value</i>

NEW false (Class Attribute)	*NEW* != (Not Equal)
NEW init	*NEW* true (Class Attribute)

4.2.1. *NEW* compareTo



Implements the abstract method inherited from the mixinclass **Comparable**.

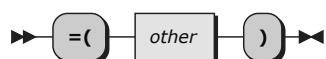
Arguments:

other

The other object representing a boolean/logical value.

Returns: **-1** if this value is less than *other*, **0** if equal, **1** if greater.

4.2.2. *NEW* = (Equal)



Equal comparison operator.

Arguments:

other

The other object representing a boolean/logical value.

Returns: **.true** if this object and *other* are equal, **.false** otherwise.

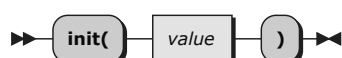
4.2.3. *NEW* false (Class Attribute)



Class getter attribute that returns the singleton representing **.false**.

Returns: the **YamlBoolean** proxy for **.false**.

4.2.4. *NEW* init



Constructor that stores the logical value. Note that the new class method is private; only the class itself can create instances via the `activate` method.

Arguments:

value

A Rexx logical value (**0** or **1**).

4.2.5. *NEW* makeString



Renders this object as a Rexx string representing its logical value.

Returns: **"1"** or **"0"**.

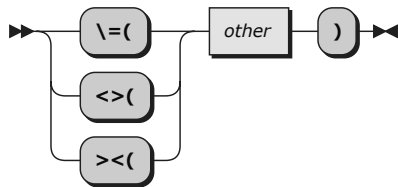
4.2.6. *NEW* makeYAML



Renders this object as a YAML string representing its logical value.

Returns: `"true"` or `"false"`.

4.2.7. *NEW* \= (Not Equal)



Unequal comparison operator. The synonyms `<>` and `><` forward to this method.

Arguments:

other

The other object representing a boolean/logical value.

Returns: `.true` if this object and *other* are not equal, `.false` otherwise.

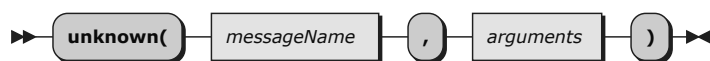
4.2.8. *NEW* true (Class Attribute)



Class getter attribute that returns the singleton representing `.true`.

Returns: the `YamlBoolean` proxy for `.true`.

4.2.9. *NEW* unknown



Forwards any unrecognised message to the underlying string value, so that `YamlBoolean` can be used transparently wherever a Rexx logical value is expected.

Arguments:

messageName

The name of the unrecognised message.

arguments

The arguments array.

4.2.10. *NEW* value



Returns the underlying Rexx logical value.

Returns: `1` for true, `0` for false.

4.3. *NEW* YamlTagged Class

Wraps a YAML value together with its explicit tag string. When the parser's **preserveTags** option is enabled (`.Yaml~new(, .true)`), any node that carries an explicit tag in the source YAML (e.g. `!!str, !custom, !<tag:yaml.org,2002:str>`) is represented as a **YamlTagged** instance instead of a bare value. Nodes without explicit tags are unaffected and appear as plain ooRexx objects.

The **value** attribute holds the resolved ooRexx object (string, integer, **Table**, **Array**, **YamlBoolean**, or **.nil**) and the **tag** attribute holds the original tag text exactly as it appeared in the YAML source.

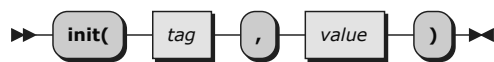
YamlTagged provides a `makeString` method so that tagged scalars can be used transparently in string contexts (e.g. **Say**), but code that performs explicit type checks with **isA** should check for **.YamlTagged** first and then inspect the **value** attribute.

The YAML emitter, XML emitter, and `yaml.deepEqual` routine all handle **YamlTagged** transparently. When a **YamlTagged** node is serialised to XML, the tag is preserved as a **tag** attribute on the corresponding element (`<scalar>`, `<mapping>`, or `<sequence>`), enabling lossless XML round-tripping of tagged YAML documents.

Table 4.3. YamlTagged Class

Object	
Methods inherited from the Object class	
YamlTagged	
<i>*NEW* init</i>	<i>*NEW* tag (Attribute)</i>
<i>*NEW* makeString</i>	<i>*NEW* value (Attribute)</i>

4.3.1. *NEW* init



Creates a new **YamlTagged** wrapper.

Arguments:

tag

The tag string (e.g. `!!str`, `!custom`, `!<tag:yaml.org,2002:str>`).

value

The resolved ooRexx value.

Returns: a new **YamlTagged** instance.

4.3.2. *NEW* makeString



Returns the string representation of the wrapped value, so that **YamlTagged** can be used transparently in string contexts. If the wrapped value is **.nil**, returns the empty string.

Returns: the string representation of the wrapped value.

4.3.3. *NEW* tag (Attribute)



Returns the tag string exactly as it appeared in the YAML source (e.g. "!!str", "!custom", "!<tag:yaml.org,2002:str>").

Returns: a string.

4.3.4. *NEW* value (Attribute)



Returns the resolved ooRexx value (string, integer, **Table**, **Array**, **YamlBoolean**, or **.nil**).

Returns: the wrapped ooRexx object.

4.4. *NEW* YamlError Class

Represents an error encountered during YAML parsing. Instances are raised via **Raise Syntax** and carry the error message, source location, and the offending input line for diagnostic purposes.

Table 4.4. YamlError Class

Object		
Methods inherited from the Object class		
YamlError		
<i>*NEW* column (Attribute)</i>	<i>*NEW* init</i>	<i>*NEW* makeString</i>
<i>*NEW* contextLine (Attribute)</i>	<i>*NEW* lineNumber (Attribute)</i>	<i>*NEW* message (Attribute)</i>

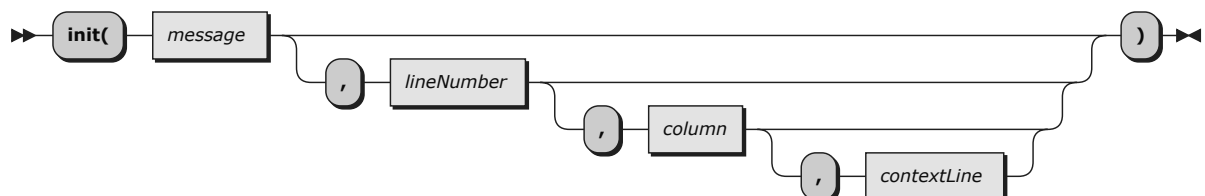
4.4.1. *NEW* column (Attribute)

The 1-based column where the error was detected (0 if unknown). This is a read-write attribute.

4.4.2. *NEW* contextLine (Attribute)

The source line that triggered the error (empty string if unavailable). This is a read-write attribute.

4.4.3. *NEW* init



Creates a new **YamlError** instance.

Arguments:

message

The error description.

lineNumber

Optional 1-based line number (default 0).

column

Optional 1-based column (default 0).

contextLine

Optional source line text (default "").

4.4.4. *NEW* lineNumber (Attribute)

The 1-based line number where the error was detected (0 if unknown). This is a read-write attribute.

4.4.5. *NEW* makeString



Returns a human-readable representation of the error, including the line number and context line when available.

Returns: a formatted error string.

4.4.6. *NEW* message (Attribute)

The human-readable error description. This is a read-write attribute.

4.5. *NEW* YamlEmitter Class

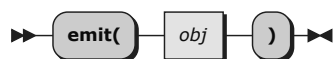
Serialises an ooRexx object tree to a YAML block-style string. This class is used internally by the **Yaml** class methods (`toYaml`, `toYamlAll`, etc.), but can also be used directly for finer control over emission.

Supports anchors/aliases and merge-key reconstruction when the corresponding metadata tables from a previous parse are supplied.

Table 4.5. YamlEmitter Class

Object	
Methods inherited from the Object class	
YamlEmitter	
<i>*NEW* emit</i>	<i>*NEW* init</i>

4.5.1. *NEW* emit



Emits the given ooRexx object tree as a YAML string.

The emitter chooses the most appropriate YAML representation for string scalars that contain newlines, following these rules:

Folded block scalar (>)

Used when the content (excluding trailing newlines) is a single line of text that does not start with a space. Long lines are wrapped at word boundaries to keep the output readable. The parser re-joins the folded lines automatically.

Literal block scalar (|)

Used when the content has multiple distinct lines (embedded newlines) or starts with a space, since those forms would be altered by folding.

Double-quoted scalar

Used when the string does not end with a newline. Embedded newlines are represented as `\n` escapes.

For block scalars (`|` and `>`), the emitter adds a chomp indicator when necessary: *clip* (no suffix) trims to a single trailing newline, while *keep* (+) preserves all trailing newlines. An explicit indent indicator digit (e.g. `|2`, `>2+`) is added whenever any content line starts with a space or tab, so that the parser can distinguish indentation from content.

When **YamlTagged** wrappers are present, the tag is placed on the same line as the block scalar header (e.g. `!custom >`), conforming to the YAML 1.2 specification requirement that a node's tag and its content indicator appear on the same line.

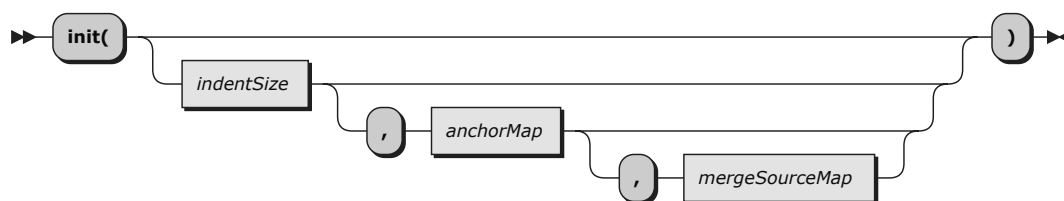
Arguments:

obj

The root object to serialise (**Table**, **Array**, scalar, or **.nil**).

Returns: a YAML-formatted string terminated by a single newline.

4.5.2. *NEW* init



Creates a new **YamlEmitter** instance.

Arguments:

indentSize

Optional number of spaces per indentation level (default 2).

anchorMap

Optional **IdentityTable** mapping objects to anchor names; if **.nil**, an empty table is used.

mergeSourceMap

Optional **IdentityTable** mapping target mappings to arrays of merge sources; if **.nil**, an empty table is used.

directivesMap

Optional **IdentityTable** mapping root objects to directives tables; if **.nil**, an empty table is used. When a root object has an entry, **%YAML** and **%TAG** directives are emitted before the document content, followed by a `---` document-start marker.

4.6. *NEW* YamlXmlEmitter Class

Serialises an in-memory ooRexx object tree to XML conforming to either **yaml.xsd** (with namespace) or **yaml.dtd** (with DOCTYPE). Used internally by `Yaml~yamlToXml` and `Yaml~yamlToXmlFile`.

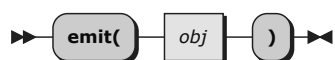
When the object tree contains **YamlTagged** wrappers (produced by the parser with **preserveTags** = **.true**), the emitter preserves each tag as a **tag** attribute on the corresponding XML element (`<scalar>`, `<mapping>`, or `<sequence>`). Verbatim tags containing `<` and `>` are XML-escaped in the attribute value.

When a `mergeSourceMap` is provided, mappings that had merge keys (`<<`) emit `<merge anchor=" . . . "/>` child elements before the regular `<entry>` elements. Keys that came from the merged source (and were not overridden by the target) are excluded from the `<entry>` list. Without a `mergeSourceMap`, all keys are emitted as regular entries (the pre-v8y backward-compatible behaviour).

Table 4.6. `YamlXmlEmitter` Class

Object	
Methods inherited from the Object class	
YamlXmlEmitter	
<i>*NEW* emit</i>	<i>*NEW* init</i>

4.6.1. *NEW* emit



Emits the given ooRexx object tree as an XML string.

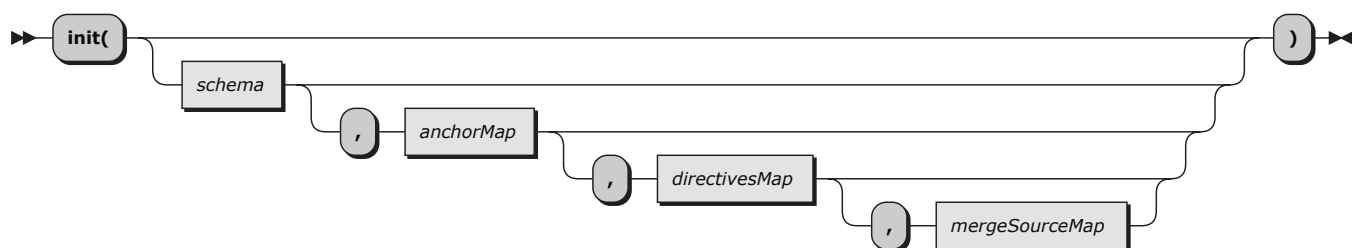
Arguments:

obj

The root object to serialise.

Returns: a well-formed XML string.

4.6.2. *NEW* init



Creates a new `YamlXmlEmitter` instance.

Arguments:

schema

Optional schema type: "xsd" (default) or "dtd".

anchorMap

Optional **IdentityTable** mapping objects to anchor names.

directivesMap

Optional **IdentityTable** mapping root objects to directives tables. When present, the `<document>` element includes a `yaml-version` attribute and `<tag-directive>` child elements as appropriate.

mergeSourceMap

Optional **IdentityTable** mapping target mappings to arrays of merge sources. When present, mappings with merge keys emit `<merge anchor=" . . . "/>` elements and exclude merged keys from the regular `<entry>` list.

4.7. *NEW* YamlXmlParser Class

Parses XML conforming to **yaml.xsd** or **yaml.dtd** back into ooRexx objects. This is a purpose-built parser for the YAML XML vocabulary; it is not a general-purpose XML parser.

When a **<mapping>**, **<sequence>**, or **<scalar>** element carries a **tag** attribute, the parser wraps the resulting node in a **YamlTagged** object that pairs the tag string with the resolved value. Verbatim tags are XML-unescape to restore the original tag text.

When a **<mapping>** element contains **<merge anchor="...">** child elements, the parser resolves the referenced anchors, copies their keys into the target mapping (without overwriting existing keys), and records the merge relationship in `mergeSourceMap`. Forward references (merge anchors that appear later in the document) are resolved automatically after the full document has been parsed.

Used internally by `Yaml~parseXml` and `Yaml~parseXmlFile`.

Table 4.7. YamlXmlParser Class

Object	
Methods inherited from the Object class	
YamlXmlParser	
<i>*NEW* directivesMap (Attribute)</i>	<i>*NEW* parse</i>
<i>*NEW* mergeSourceMap (Attribute)</i>	<i>*NEW* xmlAnchors (Attribute)</i>

4.7.1. *NEW* directivesMap (Attribute)

Returns the directives map from the most recent parse. Has the same structure as `Yaml~directivesMap`: an **IdentityTable** mapping root objects to directives **Tables**. The `Yaml~parseXml` method merges this map into its own `directivesMap` automatically.

Returns: an **IdentityTable**.

4.7.2. *NEW* mergeSourceMap (Attribute)

Returns the merge-source map from the most recent parse. Has the same structure as `Yaml~mergeSourceMap`: an **IdentityTable** mapping target mappings to **Arrays** of source mappings that were merged into them. The `Yaml~parseXml` method merges this map into its own `mergeSourceMap` automatically.

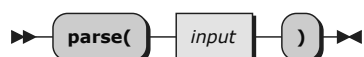
Returns: an **IdentityTable**.

4.7.3. *NEW* xmlAnchors (Attribute)

Returns the anchor table from the most recent parse, mapping anchor name strings to the resolved ooRexx objects. The `Yaml~parseXml` method uses this to build the instance `anchorMap` (which maps objects to names, i.e. the reverse direction).

Returns: a **Table**.

4.7.4. *NEW* parse



Parses an XML string into an ooRexx object tree.

Arguments:*input*

A well-formed XML string conforming to the YAML XML vocabulary.

Returns: the ooRexx object tree.

4.8. *NEW* yaml.deepEqual Public Routine



Recursively compares two ooRexx objects for semantic equality. **YamlTagged** objects are compared by tag (strict comparison) and by recursively comparing the wrapped values. **Table** objects are compared by keys and values (order-independent), **Array** objects by length and element-wise equality (order-dependent), and scalars by strict comparison (`==`). If only one operand is a **YamlTagged** wrapper, the objects are not equal.

Arguments:*expected*

The expected object (may be `.nil`, **YamlTagged**, **Table**, **Array**, or a scalar).

actual

The actual object.

Returns: `.true` if the two objects are semantically equal, `.false` otherwise.

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 Advanced 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 United 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 <https://github.com/GuntherRademacher/rr>. 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

Symbols

%TAG directive, 23
%YAML directive, 23
= method
 of JsonBoolean class, 16
 of YamlBoolean class, 33
\= method
 of JsonBoolean class, 17
 of YamlBoolean class, 34

A

activate method
 of JsonBoolean class, 15
anchorMap attribute
 of Yaml class, 24
attribute
 anchorMap
 of Yaml class, 24
 tag
 of YamlTagged class, 35
 value
 of YamlTagged class, 36

B

block scalar
 style selection, 37

C

chomp indicator
 in block scalars, 37
CLOSE method
 scvStream, 2
column attribute
 of JsonError class, 18
 of YamlError class, 36
Common Public License, 44
compareTo method
 of JsonBoolean class, 15
 of YamlBoolean class, 33
contextLine attribute
 of JsonError class, 18
 of YamlError class, 36
CPL, 44
CSVLINEIN method
 scvStream, 2
CSVLINEOUT method
 scvStream, 3
csvStream
 Attributes, 1
 CLOSE method, 2
 CSVLINEIN method, 2

CSVLINEOUT method, 3
DELIMITER attribute, 5
Example code, 5
GETHEADERS method, 3
HEADERS attribute, 5
Inherited methods, 1, 2
INIT method, 3
Methods, 1, 2
OPEN method, 4
QUALIFIER attribute, 5
SETHEADERS method, 5
SKIPHEADERS attribute, 5

D

DELIMITER attribute
 scvStream, 5
directives
 %TAG, 23
 %YAML, 23
directivesMap attribute
 of Yaml class, 23, 24
 of YamlXmlParser class, 40

E

emit method
 of JsonXmlEmitter class, 20
 of YamlEmitter class, 37
 of YamlXmlEmitter class, 39
EXECIO subcommand
 hostemu, 8

F

false attribute
 of Json class, 11
 of JsonBoolean class, 16
 of Yaml class, 24
 of YamlBoolean class, 33
folded block scalar (see [emit method](#))
fromJSON method
 of Json class, 12
fromJsonFile method
 of Json class, 12

G

GETHEADERS method
 scvStream, 3

H

HEADERS attribute
 scvStream, 5
HI subcommand
 hostemu, 9
HostEmu subcommands

EXECIO, 8
HI, 9
TE, 9
TS, 9

I

indent indicator
 in block scalars, 37
INIT method
 scvStream, 3
init method
 of Json class, 12
 of JsonBoolean class, 16
 of JsonError class, 18
 of JsonXmlEmitter class, 20
 of Yaml class, 24, 25
 of YamlBoolean class, 33
 of YamlEmitter class, 38
 of YamlError class, 36
 of YamlTagged class, 35
 of YamlXmlEmitter class, 39

J

Json class, 11
 false attribute, 11
 fromJson method, 11
 fromJsonFile method, 11
 init method, 11
 toJson method, 11
 toJsonFile method, 11
 true attribute, 11
json.cls, 10
JsonBoolean class, 15
 = method, 15
 activate method, 15
 compareTo method, 15
 false attribute, 15
 init method, 15
 makeJSON method, 15
 makeString method, 15
 true attribute, 15
 unknown method, 15
 value method, 15
 \= method, 15
JsonError class, 18
 column attribute, 18
 contextLine attribute, 18
 init method, 18
 lineNumber attribute, 18
 makeString method, 18
 message attribute, 18
JsonString class, 17
 makeJSON method, 17

jsonToXml method
 of Json class, 12
jsonToXmlFile method
 of Json class, 13
JsonXmlEmitter class, 19
JsonXmlParser class, 20

L

License, Common Public, 44
License, Open Object Rexx, 44
lineNumber attribute
 of JsonError class, 19
 of YamlError class, 37
literal block scalar (see [emit method](#))

M

makeJSON method
 of JsonBoolean class, 16
 of JsonString class, 18
makeString method
 of JsonBoolean class, 16
 of JsonError class, 19
 of YamlBoolean class, 33
 of YamlError class, 37
 of YamlTagged class, 35
makeYAML method
 of YamlBoolean class, 34
merge element
 in XML vocabulary, 40
mergeSourceMap attribute
 of Yaml class, 24
 of YamlXmlParser class, 40
message attribute
 of JsonError class, 19
 of YamlError class, 37
Methods, csvStream, 1

N

Notices, 42

O

ooRexx License, 44
OPEN method
 scvStream, 4
Open Object Rexx License, 44

P

parse method
 of JsonXmlParser class, 20
 of YamlXmlParser class, 40
parseAll method
 of Yaml class, 25
parseAllFile method

- of Yaml class, 26
- parseArray method
 - of Yaml class, 26
- parseFile method
 - of Yaml class, 26
- parseFrontMatter method
 - of Yaml class, 26
- parseFrontMatterFile method
 - of Yaml class, 26
- parseString method
 - of Yaml class, 27
- parseXml method
 - of Json class, 13
 - of Yaml class, 27
- parseXmlFile method
 - of Json class, 13
 - of Yaml class, 27
- preserveTags (see [YamlTagged class](#))
- preserveTags parameter
 - of Yaml class, 23

Q

QUALIFIER attribute
scvStream, 5

R

readFileLines method

- of Yaml class, 28

resolveTagHandle method

- of Yaml class, 27

routine

- string2json, 21
- yaml.deepEqual, 41

S

SETHEADERS method
scvStream, 5

SKIPHEADERS attribute
scvStream, 5

string2json routine, 21

T

tag attribute

- in XML vocabulary, 38
- of YamlTagged class, 35

tag handle resolution

- %TAG directive, 23
- resolveTagHandle method, 27

tag preservation, 35

TE subcommand
hostemu, 9

toJSON method

- of Json class, 14

toJsonFile method

- of Json class, 14

toYaml method

- of Yaml class, 28

toYamlAll method

- of Yaml class, 29

toYamlArray method

- of Yaml class, 29

toYamlFile method

- of Yaml class, 30

toYamlFMFile method

- of Yaml class, 30

true attribute

- of Json class, 15
- of JsonBoolean class, 17
- of Yaml class, 31
- of YamlBoolean class, 34

TS subcommand
hostemu, 9

U

unknown method

- of JsonBoolean class, 17
- of YamlBoolean class, 34

V

value attribute

- of YamlTagged class, 36

value method

- of JsonBoolean class, 17
- of YamlBoolean class, 34

X

xmlAnchors attribute

- of YamlXmlParser class, 40

Y

Yaml class, 23

- anchorMap attribute, 23
- directivesMap attribute, 23
- false attribute, 23
- init method, 23
- mergeSourceMap attribute, 23
- parseAll method, 23
- parseAllFile method, 23
- parseArray method, 23
- parseFile method, 23
- parseFrontMatter method, 23
- parseFrontMatterFile method, 23
- parseString method, 23
- parseXml method, 23
- parseXmlFile method, 23
- resolveTagHandle method, 23

- toYaml method, 23
- toYamlAll method, 23
- toYamlArray method, 23
- toYamlFile method, 23
- toYamlFMFile method, 23
- true attribute, 23
- yamlToXml method, 23
- yamlToXmlFile method, 23
- yaml.cls, 22
- yaml.deepEqual routine, 41
- YamlBoolean class, 32
 - = method, 32
 - compareTo method, 32
 - false attribute, 32
 - init method, 32
 - makeString method, 32
 - makeYAML method, 32
 - true attribute, 32
 - unknown method, 32
 - value method, 32
 - \= method, 32
- YamlEmitter class, 37
 - emit method, 37
 - init method, 37
- YamlError class, 36
 - column attribute, 36
 - contextLine attribute, 36
 - init method, 36
 - lineNumber attribute, 36
 - makeString method, 36
 - message attribute, 36
- YamlTagged class, 35
 - init method, 35
 - makeString method, 35
 - tag attribute, 35
 - value attribute, 35
- yamlToXml method
 - of Yaml class, 31
- yamlToXmlFile method
 - of Yaml class, 32
- YamlXmlEmitter class, 38
 - emit method, 38
 - init method, 38
- YamlXmlParser class, 40
 - directivesMap attribute, 40
 - mergeSourceMap attribute, 40
 - parse method, 40
 - xmlAnchors attribute, 40