# rsv Group

John D. Baker

https://github.com/bakerjd99/jackshacks/blob/main/rsv.ijs

SHA-256: 53d5c49f419f158761193d74e274ae2573b8bd9444cc8b01806270b17f410860

January 13, 2024

## Contents

# rsv Overview

rsv is a J script that decodes and encodes RSV files.

RSV is distributed as an auxiliary J addon. Auxillary addons are hosted in private GitHub repositories. rsv can be installed in the local J folder ~addons/jacks with that standard J pacman utility:

```
load 'pacman'

NB. files from https://github.com/bakerjd99/jackshacks
install 'github:bakerjd99/jackshacks'

NB. installed files
dir '~addons/jacks'
```

After installing the rsv addon download the test files at:

https://github.com/Stenway/RSV-Challenge/tree/main/TestFiles

and save them in a local directory. Cloning the repository

https://github.com/Stenway/RSV-Challenge

is a handy way to get these files.

After saving the test files in a local directory, define a J configured folder ~RSVTEST that points to the test files. J configured folders are defined in the file jpath '~user/config/folders.cfg'. Add a line to this file like:

```
NB. windows
RSVTEST c:/mp/zighacks/RSV-Challenge/TestFiles
```

```
NB. macOS linux
RSVTEST /users/mystuff/RSV-Challenge/TestFiles
```

Then save the edited config file and restart J. The expression `jpath '~RSVTEST'` should expand to the location of the test files. Running `rsv` is now a simple matter of:

```
load '~addons/jacks/rsv.ijs'

NB. files from https://github.com/Stenway/RSV-Challenge
NB. are stored in a J configured directory RSVTEST
jpath '~RSVTEST'

NB. decode rsv file
rsvdec read jpath '~RSVTEST/Valid_001.rsv'

NB. direct definition version
NB. one line of J code decodes rsv
rsvdecdd=: {{ ]`('null'"_)@.((,RSVNULL)&-:)L:0 <;._2&.> <;._2 y }}

NB. list of valid rsv test files
validrsv=: 1 dir '~RSVTEST/Valid*.rsv'

NB. encode decode test - 1 when OK 0 otherwise
rsvdent=: {{rsv -: rsvenc rsvdec rsv=. read y}}

NB. decode all valid files
```

```
(] ,. rsvdec@read&.>) validrsv

NB. list any valid files that fail decode/encode test - should be none
bool=: rsvdent&> valid_rsv
smoutput >valid_rsv #~ -.bool

NB. all tacit and dd decodings should match - result is 1
*./ (rsvdec -: rsvdecdd)&> read&.> validrsv
```

## rsv Interface

```
read   [7] reads a file as a list of bytes
rsvdec [7] decode rsv bytes - marks nulls with (NULLMARK)
rsvenc [7] encode rsv bytes - marks nulls with (NULLMARK)
rsvok  [8] 1 if blblcl rsv nouns have no bad bytes - 0 otherwise
write  [8] writes a list of bytes to file
```

## rsv Algorithm Notes

For an excellent description of RSV files see the YouTube video:

https://www.youtube.com/watch?v=tb_70o6ohMA

## rsv Source Code

```
NB.*rsv s-- j script for encoding and decoding rsv files.
NB.
NB. verbatim: see:
NB.
NB.   https://github.com/Stenway/RSV-Specification
NB.   https://github.com/Stenway/RSV-Challenge
NB.   https://www.youtube.com/watch?v=tb_7Oo6ohMA
NB.
NB. interface word(s):
NB. ------------------------------------------------------------------------
NB.  read   - reads a file as a list of bytes
NB.  rsvdec - decode rsv bytes - marks nulls with (NULLMARK)
NB.  rsvenc - encode rsv bytes - marks nulls with (NULLMARK)
NB.  rsvok  - 1 if blblcl rsv nouns have no bad bytes - 0 otherwise
NB.  write  - writes a list of bytes to file
NB.
NB. created: 2024jan08
NB. changes: ---------------------------------------------------------------

coclass 'rsv'

NB.*end-header

NB. interface words (IFACEWORDSrsv) group
IFACEWORDSrsv=: <;._1 ' read rsvdec rsvenc rsvok write'
```

```
NB. string used to mark RSV nulls
NULLMARK=: 'null'

NB. row terminator byte - hex: FD
REOR=: 253{a.

NB. value terminator byte - hex: FF
REOV=: 255{a.

NB. null value byte - hex: FE
RNULL=: 254{a.

NB. root words (ROOTWORDSrsv) group
ROOTWORDSrsv=: <;._1 ' IFACEWORDSrsv ROOTWORDSrsv VMDrsv read rsvdec rsvenc rsvok write'

NB. bytes that should never be emitted by UTF8 encoders
UTF8BADBYTES=: _8{.a.

NB. version, make count, and date
VMDrsv=: '0.9.0';01;'13 Jan 2024 11:09:01'

NB. signal with optional message
assert=: 0 0"_ $ 13!:8^:((0: e. ])`(12"_))

NB. tests for character data
ischar=: 2&=@(3!:0)
```

```
NB. reads a file as a list of bytes
read=: 1!:1&(]`<@.(32&>@(3!:0)))

rsvdec=: 3 : 0

NB.*rsvdec v-- decode rsv bytes - marks nulls with (NULLMARK).
NB.
NB. monad: blblcl =. rsvdec clRsv
NB.
NB.    rsv=. read jpath '~RSVTEST/Valid_001.rsv'
NB.    rsvdec rsv

]`(NULLMARK"_)@.((,RNULL)&-:)L:0 <;._2&.> <;._2 y
)

rsvenc=: 3 : 0

NB.*rsvenc v-- encode rsv bytes - marks nulls with (NULLMARK).
NB.
NB. monad: clRsv =. rsvenc blblclRsv
NB.
NB.    rsv=. rsvdec read jpath '~RSVTEST/Valid_001.rsv'
NB.    rsvenc rsv

(0=#y) }. ; ,&REOR&.> ;&.> REOV -.&.>~ ,&REOV L: 0 (]`(RNULL"_))@.(NULLMARK&-:) L: 0 y
)
```

```
rsvok=: 3 : 0

NB.*rsvok v-- 1 if blblcl rsv nouns have no bad bytes - 0 otherwise.
NB.
NB. monad: pa =. rsvok blblclRsv
NB.
NB.   NB. check blblcl for bad bytes before encoding
NB.   lol=: (<"1 <"0 ?2 2$1000), (<'') , <"0 ;:'some words'
NB.   lol=: utf8@": L: 0 ,&.> lol
NB.
NB.   NB. no bad bytes in utf8 formatted cells - result 1
NB.   rsvok lol
NB.
NB.   NB. add an RSV delimiter byte - result 0
NB.   rsvok lol,<,<RSVEOR

NB. sublists must be list of lists of char
msg=. 'not a list of lists of characters'
msg assert 1 = #@$y
msg assert *./ ischar&> ;y
msg assert -.0 e. #@$ &> y

NB. without bad bytes
-. +./ ;; L: 1 ] 1&e.@(UTF8BADBYTES&e.) L: 0 y
)

NB. writes a list of bytes to file
write=: 1!:2 ]`<@.(32&>@(3!:0))
```

```
NB.POST_rsv post processor.

smoutput IFACE=:  (0 : 0)
NB. (rsv) interface word(s): 20240113j110901
NB. ------------------------
NB. read    NB. reads a file as a list of bytes
NB. rsvdec  NB. decode rsv bytes - marks nulls with (NULLMARK)
NB. rsvenc  NB. encode rsv bytes - marks nulls with (NULLMARK)
NB. rsvok   NB. 1 if blblcl rsv nouns have no bad bytes - 0 otherwise
NB. write   NB. writes a list of bytes to file
)

cocurrent 'base'
coinsert  'rsv'
```

# Index