

# Package ‘qs’

March 10, 2020

**Type** Package

**Title** Quick Serialization of R Objects

**Version** 0.21.2

**Date** 2020-3-10

**Maintainer** Travers Ching <traversc@gmail.com>

**Description** Provides functions for quickly writing and reading any R object to and from disk.

**License** GPL-3 | file LICENSE

**LazyData** true

**Biarch** true

**SystemRequirements** C++11

**Imports** Rcpp, RApiSerialize

**LinkingTo** Rcpp, RApiSerialize

**RoxygenNote** 6.1.1

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**Copyright** This package includes code from the 'zstd' library owned by Facebook, Inc. and created by Yann Collet; the 'lz4' library created by Yann Collet; and code derived from the 'Blosc' library created by Francesc Alted.

**URL** <https://github.com/traversc/qs>

**BugReports** <https://github.com/traversc/qs/issues>

**NeedsCompilation** yes

**Author** Travers Ching [aut, cre, cph],  
Yann Collet [ctb, cph] (Yann Collet is the author of the bundled zstd and lz4 code),  
Facebook, Inc. [cph] (Facebook is the copyright holder of the bundled zstd code),  
Reichardt Tino [ctb, cph] (Contributor/copyright holder of zstd bundled code),

Skibinski Przemyslaw [ctb, cph] (Contributor/copyright holder of zstd bundled code),  
 Mori Yuta [ctb, cph] (Contributor/copyright holder of zstd bundled code),  
 Romain Francois [ctb, cph] (Derived example/tutorials for Alt-Rep structures),  
 Francisc Alted [ctb, cph] (Shuffling routines derived from Blosc library)

**Repository** CRAN

**Date/Publication** 2020-03-10 17:40:02 UTC

## R topics documented:

base85_decode	3
base85_encode	3
base91_decode	4
base91_encode	4
blosc_shuffle_raw	5
blosc_unshuffle_raw	5
catquo	6
convertToAlt	6
is_big_endian	7
lz4_compress_bound	7
lz4_compress_raw	8
lz4_decompress_raw	8
qdeserialize	9
qdump	9
qread	10
qread_fd	11
qread_handle	12
qread_ptr	12
qsave	13
qsave_fd	15
qsave_handle	16
qserialize	17
randomStrings	17
starnames	18
zstd_compress_bound	19
zstd_compress_raw	19
zstd_decompress_raw	20

**Index**

**21**

---

base85_decode	<i>Z85 Decoding</i>
---------------	---------------------

---

**Description**

Decodes a Z85 encoded string back to binary

**Usage**

```
base85_decode(encoded_string)
```

**Arguments**

encoded\_string A string

---

base85_encode	<i>Z85 Encoding</i>
---------------	---------------------

---

**Description**

Encodes binary data (a raw vector) as ascii text using Z85 encoding format

**Usage**

```
base85_encode(rawdata)
```

**Arguments**

rawdata A raw vector

**Details**

Z85 is a binary to ascii encoding format created by Pieter Hintjens in 2010 and is part of the ZeroMQ RFC. The encoding has a dictionary using 85 out of 94 printable ASCII characters. There are other base 85 encoding schemes, including Ascii85, which is popularized and used by Adobe. Z85 is distinguished by its choice of dictionary, which is suitable for easier inclusion into source code for many programming languages. The dictionary excludes all quote marks and other control characters, and requires no special treatment in R and most other languages. Note: although the official specification restricts input length to multiples of four bytes, the implementation here works with any input length. The overhead (extra bytes used relative to binary) is 25%. In comparison, base 64 encoding has an overhead of 33.33%.

**References**

<https://rfc.zeromq.org/spec/32/>

base91\_decode      *basE91 Decoding*

---

**Description**

Decodes a basE91 encoded string back to binary

**Usage**

```
base91_decode(encoded_string)
```

**Arguments**

encoded\_string    A string

---

base91\_encode      *basE91 Encoding*

---

**Description**

Encodes binary data (a raw vector) as ascii text using basE91 encoding format

**Usage**

```
base91_encode(rawdata)
```

**Arguments**

rawdata            A raw vector

**Details**

basE91 (capital E for stylization) is a binary to ascii encoding format created by Joachim Henke in 2005. The encoding has a dictionary using 91 out of 94 printable ASCII characters; excludes - (dash), \ (backslash) and ' (single quote). The overhead (extra bytes used relative to binary) is 22.97% on average. In comparison, base 64 encoding has an overhead of 33.33%. Because the dictionary includes double quotes, basE91 encoded data must be single quoted when stored as a string in R.

**References**

<http://base91.sourceforge.net/>

---

blosc\_shuffle\_raw      *Shuffle a raw vector*

---

**Description**

A function for shuffling a raw vector using BLOSC shuffle routines

**Usage**

```
blosc_shuffle_raw(x, bytesofsize)
```

**Arguments**

x	The raw vector
bytesofsize	Either 4 or 8

**Value**

The shuffled vector

**Examples**

```
x <- serialize(1L:1000L, NULL)
xshuf <- blosc_shuffle_raw(x, 4)
xunshuf <- blosc_unshuffle_raw(xshuf, 4)
```

---

blosc\_unshuffle\_raw      *Un-shuffle a raw vector*

---

**Description**

A function for un-shuffling a raw vector using BLOSC un-shuffle routines

**Usage**

```
blosc_unshuffle_raw(x, bytesofsize)
```

**Arguments**

x	The raw vector
bytesofsize	Either 4 or 8

**Value**

The unshuffled vector

**Examples**

```
x <- serialize(1L:1000L, NULL)
xshuf <- blosc_shuffle_raw(x, 4)
xunshuf <- blosc_unshuffle_raw(xshuf, 4)
```

---

catquo	<i>catquo</i>
--------	---------------

---

**Description**

Prints a string with single quotes on a new line

**Usage**

```
catquo(...)
```

**Arguments**

... Arguments passed to 'cat' function

---

convertToAlt	<i>Convert character vector to alt-rep</i>
--------------	--

---

**Description**

A function for generating a alt-rep object from a character vector, for users to experiment with the alt-rep system. This function is not available in R versions earlier than 3.5.0.

**Usage**

```
convertToAlt(x)
```

**Arguments**

x The character vector

**Value**

The character vector in alt-rep form

**Examples**

```
xalt <- convertToAlt(randomStrings(N=10, string_size=20))
xalt2 <- convertToAlt(c("a", "b", "c"))
```

---

is_big_endian	<i>System Endianness</i>
---------------	--------------------------

---

**Description**

Tests system endianness. Intel and AMD based systems are little endian, and so this function will likely return 'FALSE'. The 'qs' package is not capable of transferring data between systems of different endianness. This should not matter for the large majority of use cases.

**Usage**

```
is_big_endian()
```

**Value**

'TRUE' if big endian, 'FALSE' if little endian.

**Examples**

```
is_big_endian() # returns FALSE on Intel/AMD systems
```

---

lz4_compress_bound	<i>lz4 compress bound</i>
--------------------	---------------------------

---

**Description**

Exports the compress bound function from the lz4 library. Returns the maximum compressed size of an object of length 'size'.

**Usage**

```
lz4_compress_bound(size)
```

**Arguments**

size	An integer size
------	-----------------

**Value**

maximum compressed size

**Examples**

```
lz4_compress_bound(100000)  
#' lz4_compress_bound(1e9)
```

---

lz4\_compress\_raw      *lz4 compression*

---

**Description**

Compression of raw vector. Exports the main lz4 compression function.

**Usage**

```
lz4_compress_raw(x, compress_level)
```

**Arguments**

x                    A Raw Vector  
compress\_level      The compression level (> 1).

**Value**

The compressed data

**Examples**

```
x <- 1:1e6  
xserialized <- serialize(x, connection=NULL)  
xcompressed <- lz4_compress_raw(xserialized, compress_level = 1)  
xrecovered <- unserialize(lz4_decompress_raw(xcompressed))
```

---

lz4\_decompress\_raw      *lz4 decompression*

---

**Description**

Decompresses of raw vector

**Usage**

```
lz4_decompress_raw(x)
```

**Arguments**

x                    A Raw Vector

**Value**

The uncompressed data



**Examples**

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- lz4_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(lz4_decompress_raw(xcompressed))
```

---

qdeserialize

*qdeserialize*


---

**Description**

Reads an object from a fd

**Usage**

```
qdeserialize(x, use_alt_rep=FALSE, strict=FALSE)
```

**Arguments**

x	a raw vector
use_alt_rep	Use alt rep when reading in string data. Default: FALSE. (Note: on R versions earlier than 3.5.0, this parameter does nothing.)
strict	Whether to throw an error or just report a warning (Default: FALSE, report warning)

**Details**

See ‘?qserialize’ for additional details and examples.

**Value**

The de-serialized object

---

qdump

*qdump*


---

**Description**

Exports the uncompressed binary serialization to a list of Raw Vectors. For testing purposes and exploratory purposes mainly.

**Usage**

```
qdump(file)
```

**Arguments**

file            the file name/path.

**Value**

The uncompressed serialization

**Examples**

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
x2 <- qdump(myfile)
```

---

qread

*qread*


---

**Description**

Reads an object in a file serialized to disk

**Usage**

```
qread(file, use_alt_rep=FALSE, strict=FALSE, nthreads=1)
```

**Arguments**

file            the file name/path

use\_alt\_rep    Use alt rep when reading in string data. Default: FALSE. (Note: on R versions earlier than 3.5.0, this parameter does nothing.)

strict         Whether to throw an error or just report a warning (Default: FALSE, report warning)

nthreads       Number of threads to use. Default 1.

**Value**

The de-serialized object

**Examples**

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
x2 <- qread(myfile)
identical(x, x2) # returns true

# qs support multithreading
qsave(x, myfile, nthreads=2)
x2 <- qread(myfile, nthreads=2)
identical(x, x2) # returns true

# Other examples
z <- 1:1e7
myfile <- tempfile()
qsave(z, myfile)
z2 <- qread(myfile)
identical(z, z2) # returns true

w <- as.list(rnorm(1e6))
myfile <- tempfile()
qsave(w, myfile)
w2 <- qread(myfile)
identical(w, w2) # returns true
```

---

qread\_fd

*qread\_fd*

---

**Description**

Reads an object from a file descriptor

**Usage**

```
qread_fd(fd, use_alt_rep=FALSE, strict=FALSE)
```

**Arguments**

fd	A file descriptor
use_alt_rep	Use alt rep when reading in string data. Default: FALSE. (Note: on R versions earlier than 3.5.0, this parameter does nothing.)
strict	Whether to throw an error or just report a warning (Default: FALSE, report warning)

**Details**

See ‘?qsave\_fd’ for additional details and examples.

**Value**

The de-serialized object

---

qread_handle	<i>qread_handle</i>
--------------	---------------------

---

**Description**

Reads an object from a windows handle

**Usage**

```
qread_handle(handle, use_alt_rep=FALSE, strict=FALSE)
```

**Arguments**

handle	A windows handle external pointer
use_alt_rep	Use alt rep when reading in string data. Default: FALSE. (Note: on R versions earlier than 3.5.0, this parameter does nothing.)
strict	Whether to throw an error or just report a warning (Default: FALSE, report warning)

**Details**

See ‘?qsave\_handle’ for additional details and examples.

**Value**

The de-serialized object

---

qread_ptr	<i>qread_ptr</i>
-----------	------------------

---

**Description**

Reads an object from a external pointer

**Usage**

```
qread_ptr(pointer, length, use_alt_rep=FALSE, strict=FALSE)
```

**Arguments**

pointer	An external pointer to memory
length	the length of the object in memory
use_alt_rep	Use alt rep when reading in string data. Default: FALSE. (Note: on R versions earlier than 3.5.0, this parameter does nothing.)
strict	Whether to throw an error or just report a warning (Default: FALSE, report warning)

**Value**

The de-serialized object

---

qsave	<i>qsave</i>
-------	--------------

---

**Description**

Saves (serializes) an object to disk.

**Usage**

```
qsave(x, file,
      preset = "high", algorithm = "zstd", compress_level = 4L,
      shuffle_control = 15L, check_hash=TRUE, nthreads = 1)
```

**Arguments**

x	the object to serialize.
file	the file name/path.
preset	One of "fast", "high" (default), "high", "archive", "uncompressed" or "custom". See details.
algorithm	Compression algorithm used: "lz4", "zstd", "lz4hc", "zstd_stream" or "uncompressed".
compress_level	The compression level used (Default 1). For lz4, this number must be > 1 (higher is less compressed). For zstd, a number between -50 to 22 (higher is more compressed).
shuffle_control	An integer setting the use of byte shuffle compression. A value between 0 and 15 (Default 15). See details.
check_hash	Default TRUE, compute a hash which can be used to verify file integrity during serialization
nthreads	Number of threads to use. Default 1.

## Details

This function serializes and compresses R objects using block compression with the option of byte shuffling. There are lots of possible parameters. This function exposes three parameters related to compression level and byte shuffling.

‘compress\_level’ - Higher values tend to have a better compression ratio, while lower values/negative values tend to be quicker. Due to the format of qs, there is very little benefit to compression levels > 5 or so.

‘shuffle\_control’ - This sets which numerical R object types are subject to byte shuffling. Generally speaking, the more ordered/sequential an object is (e.g., ‘1:1e7’), the larger the potential benefit of byte shuffling. It is not uncommon to have several orders magnitude benefit to compression ratio or compression speed. The more random an object is (e.g., ‘rnorm(1e7)’), the less potential benefit there is, even negative benefit is possible. Integer vectors almost always benefit from byte shuffling whereas the results for numeric vectors are mixed. To control block shuffling, add +1 to the parameter for logical vectors, +2 for integer vectors, +4 for numeric vectors and/or +8 for complex vectors.

The ‘preset’ parameter has several different combination of parameter sets that are performant over a large variety of data. The ‘algorithm’ parameter, ‘compress\_level’ and ‘shuffle\_control’ parameters are ignored unless ‘preset’ is "custom". "fast" preset: algorithm lz4, compress\_level 100, shuffle\_control 0. "balanced" preset: algorithm lz4, compress\_level 1, shuffle\_control 15. "high" preset: algorithm zstd, compress\_level 4, shuffle\_control 15. "archive" preset: algorithm zstd\_stream, compress\_level 14, shuffle\_control 15. (zstd\_stream is currently single threaded only)

## Value

The total number of bytes written to the file (returned invisibly)

## Examples

```
x <- data.frame(int = sample(1e3, replace=TRUE),
                num = rnorm(1e3),
                char = randomStrings(1e3), stringsAsFactors = FALSE)
myfile <- tempfile()
qsave(x, myfile)
x2 <- qread(myfile)
identical(x, x2) # returns true

# qs support multithreading
qsave(x, myfile, nthreads=2)
x2 <- qread(myfile, nthreads=2)
identical(x, x2) # returns true

# Other examples
z <- 1:1e7
myfile <- tempfile()
qsave(z, myfile)
z2 <- qread(myfile)
identical(z, z2) # returns true

w <- as.list(rnorm(1e6))
```

```
myfile <- tempfile()
qsave(w, myfile)
w2 <- qread(myfile)
identical(w, w2) # returns true
```

---

qsave\_fd

*qsave\_fd*


---

## Description

Saves an object to a file descriptor

## Usage

```
qsave_fd(x, fd,
  preset = "high", algorithm = "zstd", compress_level = 4L,
  shuffle_control = 15L, check_hash=TRUE)
```

## Arguments

x	the object to serialize.
fd	A file descriptor
preset	One of "fast", "balanced" , "high" (default), "archive", "uncompressed" or "custom". See details.
algorithm	Compression algorithm used: "lz4", "zstd", "lz4hc", "zstd_stream" or "uncompressed".
compress_level	The compression level used (Default 1). For lz4, this number must be > 1 (higher is less compressed). For zstd, a number between -50 to 22 (higher is more compressed).
shuffle_control	An integer setting the use of byte shuffle compression. A value between 0 and 15 (Default 15). See details.
check_hash	Default TRUE, compute a hash which can be used to verify file integrity during serialization

## Details

This function serializes and compresses an R object to a stream using a file descriptor. If your data is important, make sure you know what happens on the other side of the pipe. See examples for usage.

## Value

the number of bytes serialized (returned invisibly)

---

qsave_handle	<i>qsave_handle</i>
--------------	---------------------

---

### Description

Saves an object to a windows handle

### Usage

```
qsave_handle(x, handle,
  preset = "high", algorithm = "zstd", compress_level = 4L,
  shuffle_control = 15L, check_hash=TRUE)
```

### Arguments

x	the object to serialize.
handle	A windows handle external pointer
preset	One of "fast", "balanced" , "high" (default), "archive", "uncompressed" or "custom". See details.
algorithm	Compression algorithm used: "lz4", "zstd", "lz4hc", "zstd_stream" or "uncompressed".
compress_level	The compression level used (Default 1). For lz4, this number must be > 1 (higher is less compressed). For zstd, a number between -50 to 22 (higher is more compressed).
shuffle_control	An integer setting the use of byte shuffle compression. A value between 0 and 15 (Default 15). See details.
check_hash	Default TRUE, compute a hash which can be used to verify file integrity during serialization

### Details

This function serializes and compresses an R object to a stream using a file descriptor. If your data is important, make sure you know what happens on the other side of the pipe. See examples for usage.

### Value

the number of bytes serialized (returned invisibly)



---

qserialize	<i>qserialize</i>
------------	-------------------

---

### Description

Saves an object to a raw vector

### Usage

```
qserialize(x, preset = "high",
algorithm = "zstd", compress_level = 4L,
shuffle_control = 15L, check_hash=TRUE)
```

### Arguments

x	the object to serialize.
preset	One of "fast", "balanced", "high" (default), "archive", "uncompressed" or "custom". See details.
algorithm	Compression algorithm used: "lz4", "zstd", "lz4hc", "zstd_stream" or "uncompressed".
compress_level	The compression level used (Default 1). For lz4, this number must be > 1 (higher is less compressed). For zstd, a number between -50 to 22 (higher is more compressed).
shuffle_control	An integer setting the use of byte shuffle compression. A value between 0 and 15 (Default 15). See details.
check_hash	Default TRUE, compute a hash which can be used to verify file integrity during serialization

### Details

This function serializes and compresses an R object to a raw vector. If your data is important, make sure you know what happens on the other side of the pipe. See examples for usage.

---

randomStrings	<i>Generate random strings</i>
---------------	--------------------------------

---

### Description

A function for generating a character vector of random strings, for testing purposes.

### Usage

```
randomStrings(N, string_size)
```

**Arguments**

N                    The number of random strings to generate  
string\_size        The number of characters in each string (default 50).

**Value**

A character vector of random alpha-numeric strings.

**Examples**

```
randomStrings(N=10, string_size=20) # returns 10 alphanumeric strings of length 20  
randomStrings(N=100, string_size=200) # returns 100 alphanumeric strings of length 200
```

---

starnames	<i>Official list of IAU Star Names</i>
-----------	--

---

**Description**

Data from the International Astronomical Union. An official list of the 336 internationally recognized named stars, updated as of June 1, 2018.

**Usage**

```
data(starnames)
```

**Format**

A 'data.frame' with official IAU star names and several properties, such as coordinates.

**Source**

[Naming Stars | International Astronomical Union.](#)

**References**

E Mamajek et. al. (2018), *WG Triennial Report (2015-2018) - Star Names*, Reports on Astronomy, 22 Mar 2018.

**Examples**

```
data(starnames)
```

---

zstd_compress_bound	<i>Zstd compress bound</i>
---------------------	----------------------------

---

**Description**

Exports the compress bound function from the zstd library. Returns the maximum compressed size of an object of length 'size'.

**Usage**

```
zstd_compress_bound(size)
```

**Arguments**

size	An integer size
------	-----------------

**Value**

maximum compressed size

**Examples**

```
zstd_compress_bound(100000)  
zstd_compress_bound(1e9)
```

---

zstd_compress_raw	<i>Zstd compression</i>
-------------------	-------------------------

---

**Description**

Compression of raw vector. Exports the main zstd compression function.

**Usage**

```
zstd_compress_raw(x, compress_level)
```

**Arguments**

x	A Raw Vector
compress_level	The compression level (-50 to 22)

**Value**

The compressed data

**Examples**

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- zstd_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(zstd_decompress_raw(xcompressed))
```

---

zstd\_decompress\_raw    *Zstd decompression*

---

**Description**

Decompresses of raw vector

**Usage**

```
zstd_decompress_raw(x)
```

**Arguments**

x                    A Raw Vector

**Value**

The uncompressed data

**Examples**

```
x <- 1:1e6
xserialized <- serialize(x, connection=NULL)
xcompressed <- zstd_compress_raw(xserialized, compress_level = 1)
xrecovered <- unserialize(zstd_decompress_raw(xcompressed))
```

# Index

## \*Topic **datasets**

starnames, [18](#)

base85\_decode, [3](#)

base85\_encode, [3](#)

base91\_decode, [4](#)

base91\_encode, [4](#)

blosc\_shuffle\_raw, [5](#)

blosc\_unshuffle\_raw, [5](#)

catquo, [6](#)

convertToAlt, [6](#)

is\_big\_endian, [7](#)

lz4\_compress\_bound, [7](#)

lz4\_compress\_raw, [8](#)

lz4\_decompress\_raw, [8](#)

qdeserialize, [9](#)

qdump, [9](#)

qread, [10](#)

qread\_fd, [11](#)

qread\_handle, [12](#)

qread\_ptr, [12](#)

qsave, [13](#)

qsave\_fd, [15](#)

qsave\_handle, [16](#)

qserialize, [17](#)

randomStrings, [17](#)

starnames, [18](#)

zstd\_compress\_bound, [19](#)

zstd\_compress\_raw, [19](#)

zstd\_decompress\_raw, [20](#)