

---

# **pandas-msgpack Documentation**

***Release 0.1.0***

**PyData Development Team**

**Apr 01, 2017**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Conda . . . . .	3
1.2	Pip . . . . .	3
1.3	Install from Source . . . . .	3
1.4	Dependencies . . . . .	3
<b>2</b>	<b>Tutorial</b>	<b>5</b>
<b>3</b>	<b>Compression</b>	<b>9</b>
<b>4</b>	<b>Read/Write API</b>	<b>11</b>
<b>5</b>	<b>API Reference</b>	<b>13</b>
<b>6</b>	<b>Changelog</b>	<b>15</b>
6.1	0.1.4 / 2017-03-30 . . . . .	15
<b>7</b>	<b>Indices and tables</b>	<b>17</b>



The `pandas_msgpack` module provides an interface from *pandas* <https://pandas.pydata.org> to the `msgpack` library. This is a lightweight portable binary format, similar to binary JSON, that is highly space efficient, and provides good performance both on the writing (serialization), and reading (deserialization).

Contents:



You can install `pandas-msgpack` with `conda`, `pip`, or by installing from source.

### Conda

```
$ conda install pandas-msgpack --channel conda-forge
```

This installs `pandas-msgpack` and all common dependencies, including `pandas`.

### Pip

To install the latest version of `pandas-msgpack`:

```
$ pip install pandas-msgpack -U
```

This installs `pandas-msgpack` and all common dependencies, including `pandas`.

### Install from Source

```
$ pip install git+https://github.com/pydata/pandas-msgpack.git
```

### Dependencies

- `pandas` `>=0.19.2`
- `blosc` library can be optionally installed as a compressor.





## CHAPTER 2

### Tutorial

```
In [1]: import pandas as pd
```

```
In [2]: from pandas_msgpack import to_msgpack, read_msgpack
```

```
In [3]: df = pd.DataFrame(np.random.rand(5,2), columns=list('AB'))
```

```
In [4]: to_msgpack('foo.msg', df)
```

```
In [5]: read_msgpack('foo.msg')
```

```
Out[5]:
```

	A	B
0	0.713005	0.041765
1	0.533135	0.355991
2	0.454220	0.109903
3	0.214311	0.335754
4	0.753745	0.226781

```
In [6]: s = pd.Series(np.random.rand(5), index=pd.date_range('20130101', periods=5))
```

You can pass a list of objects and you will receive them back on deserialization.

```
In [7]: to_msgpack('foo.msg', df, 'foo', np.array([1,2,3]), s)
```

```
In [8]: read_msgpack('foo.msg')
```

```
Out[8]:
```

	A	B		
0	0.713005	0.041765		
1	0.533135	0.355991		
2	0.454220	0.109903		
3	0.214311	0.335754		
4	0.753745	0.226781	'foo', array([1, 2, 3]),	2013-01-01 0.073538
2013-01-02		0.067729		
2013-01-03		0.408923		
2013-01-04		0.698742		

```
2013-01-05    0.557989
Freq: D, dtype: float64]
```

You can pass `iterator=True` to iterate over the unpacked results

```
In [9]: for o in read_msgpack('foo.msg', iterator=True):
...:     print(o)
...:
           A           B
0  0.713005  0.041765
1  0.533135  0.355991
2  0.454220  0.109903
3  0.214311  0.335754
4  0.753745  0.226781
foo
[1 2 3]
2013-01-01    0.073538
2013-01-02    0.067729
2013-01-03    0.408923
2013-01-04    0.698742
2013-01-05    0.557989
Freq: D, dtype: float64
```

You can pass `append=True` to the writer to append to an existing pack

```
In [10]: to_msgpack('foo.msg', df, append=True)

In [11]: read_msgpack('foo.msg')
Out[11]:
[           A           B
0  0.713005  0.041765
1  0.533135  0.355991
2  0.454220  0.109903
3  0.214311  0.335754
4  0.753745  0.226781, 'foo', array([1, 2, 3]), 2013-01-01    0.073538
2013-01-02    0.067729
2013-01-03    0.408923
2013-01-04    0.698742
2013-01-05    0.557989
Freq: D, dtype: float64,           A           B
0  0.713005  0.041765
1  0.533135  0.355991
2  0.454220  0.109903
3  0.214311  0.335754
4  0.753745  0.226781]
```

Furthermore you can pass in arbitrary python objects.

```
In [12]: to_msgpack('foo2.msg', { 'dict' : [ { 'df' : df }, { 'string' : 'foo' }, {
↪ 'scalar' : 1. }, { 's' : s } ] })

In [13]: read_msgpack('foo2.msg')
Out[13]:
{'dict': ({'df':           A           B
0  0.713005  0.041765
1  0.533135  0.355991
2  0.454220  0.109903
3  0.214311  0.335754
```

```
4 0.753745 0.226781},
{'string': 'foo'},
{'scalar': 1.0},
{'s': 2013-01-01 0.073538
2013-01-02 0.067729
2013-01-03 0.408923
2013-01-04 0.698742
2013-01-05 0.557989
Freq: D, dtype: float64}}}
```



## CHAPTER 3

---

### Compression

---

Optionally, a `compression` argument will compress the resulting bytes. These can take a bit more time to write. The available compressors are `zlib` and `blosc`.

Generally compression will increase the writing time.

```
In [1]: import pandas as pd

In [2]: from pandas_msgpack import to_msgpack, read_msgpack

In [3]: df = pd.DataFrame({'A': np.arange(100000),
...:                      'B': np.random.randn(100000),
...:                      'C': 'foo'})
...:
```

```
In [4]: %timeit -n 1 -r 1 to_msgpack('uncompressed.msg', df)
1 loop, best of 1: 21.9 ms per loop
```

```
In [5]: %timeit -n 1 -r 1 to_msgpack('compressed_blosc.msg', df, compress='blosc')
1 loop, best of 1: 27.3 ms per loop
```

```
In [6]: %timeit -n 1 -r 1 to_msgpack('compressed_zlib.msg', df, compress='zlib')
1 loop, best of 1: 129 ms per loop
```

If compressed, it will be automatically inferred and de-compressed upon reading.

```
In [7]: %timeit -n 1 -r 1 read_msgpack('uncompressed.msg')
1 loop, best of 1: 21.9 ms per loop
```

```
In [8]: %timeit -n 1 -r 1 read_msgpack('compressed_blosc.msg')
1 loop, best of 1: 20.3 ms per loop
```

```
In [9]: %timeit -n 1 -r 1 read_msgpack('compressed_zlib.msg')
1 loop, best of 1: 33.8 ms per loop
```

These can provide storage space savings.

```
In [10]: !ls -ltr *.msg
-rw-r--r-- 1 docs docs 2000582 Apr  1 15:36 uncompressed.msg
-rw-r--r-- 1 docs docs 1188179 Apr  1 15:36 compressed_blosc.msg
-rw-r--r-- 1 docs docs 1320608 Apr  1 15:36 compressed_zlib.msg
```

## CHAPTER 4

### Read/Write API

Msgpacks can also be read from and written to strings.

```
In [1]: import pandas as pd

In [2]: from pandas_msgpack import to_msgpack, read_msgpack

In [3]: df = pd.DataFrame({'A': np.arange(10),
...:                      'B': np.random.randn(10),
...:                      'C': 'foo'})
...:

In [4]: to_msgpack(None, df)
Out[4]: b
'\x84\xa4axes\x92\x86\xa5dtype\xa6object\xa4name\xc0\xa4data\x93\xa1A\xa1B\xa1C\xa3typ\xa5index\xa5index\xa5klass\xaaRangeIndex\xa3typ\xadbblock_
manager\xa5klass\xa9DataFrame\xa6blocks\x93\x86\xa5dtype\xa7float64\xa5shape\x92\x01\n\xa6values\x92\x01\n\xa6values\x92\x01\n\xa6values\x92\x01\n
\xe1\x04\xf4H\x10\xe2\xd6\xbf\xfbw\xa2\x06\xbb\xeb?q\r\xc4\x90\xf2\x8c\xe7?
G2cm\xec\x97\xea\xbf\x9bN\xe9h`\xc7\xd8?
\xd8N\xec\xdf+\xca\xe1\xbf[\x92U\x8b\xbc\xbd\xba?
\x92q\x84\x11\xcd1\xfb\xbf\xc1\x84\xe2y#*\xe0?
\xa8compress\xc0\xa4locs\x86\xa4ndim\x01\xa5dtype\xa5int64\xa5shape\x91\x01\xa4data\xd7\x00\x01\x00
'
```

Furthermore you can concatenate the strings to produce a list of the original objects.

```
In [5]: read_msgpack(to_msgpack(None, df) + to_msgpack(None, df.A))
Out[5]:
[   A      B      C
0  0  0.386764  foo
1  1 -0.357548  foo
2  2  0.866580  foo
3  3  0.735956  foo
4  4 -0.831045  foo
5  5  0.387169  foo
6  6 -0.555929  foo]
```

```
7 7 0.104458 foo
8 8 -1.699658 foo
9 9 0.505144 foo, 0 0
1 1
2 2
3 3
4 4
5 5
6 6
7 7
8 8
9 9
Name: A, dtype: int64]
```



<code>read_msgpack(path_or_buf[, encoding, iterator])</code>	Load msgpack pandas object from the specified
<code>to_msgpack(path_or_buf, *args, **kwargs)</code>	msgpack (serialize) object to input file path

`pandas_msgpack.read_msgpack(path_or_buf, encoding='utf-8', iterator=False, **kwargs)`  
Load msgpack pandas object from the specified file path

**Parameters** `path_or_buf` : string File path, BytesIO like or string

**encoding**: Encoding for decoding msgpack str type

**iterator** : boolean, if True, return an iterator to the unpacker  
(default is False)

**Returns** `obj` : type of object stored in file

`pandas_msgpack.to_msgpack(path_or_buf, *args, **kwargs)`  
msgpack (serialize) object to input file path

**Parameters** `path_or_buf` : string File path, buffer-like, or None  
if None, return generated string

**args** : an object or objects to serialize

**encoding**: encoding for unicode objects

**append** : boolean whether to append to an existing msgpack  
(default is False)

**compress** : type of compressor (zlib or blosc), default to None (no  
compression)



#### 0.1.4 / 2017-03-30

Initial release of transfered code from [pandas](#)

Includes patches since the 0.19.2 release on pandas with the following:

- Bug in `read_msgpack()` in which `Series` categoricals were being improperly processed, see [pandas-GH#14901](#)
- Bug in `read_msgpack()` which did not allow loading of a dataframe with an index of type `CategoricalIndex`, see [pandas-GH#15487](#)
- Bug in `read_msgpack()` when deserializing a `CategoricalIndex`, see [pandas-GH#15487](#)



## CHAPTER 7

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## R

`read_msgpack()` (in module `pandas_msgpack`), [13](#)

## T

`to_msgpack()` (in module `pandas_msgpack`), [13](#)