

---

# **msgpack Documentation**

*Release 0.4*

**Author**

**2017-11-04**



---

## Contents

---

<b>1 API reference</b>	<b>3</b>
<b>Python Module Index</b>	<b>9</b>



MessagePack is a efficient format for inter language data exchange.



`msgpack.pack(o, stream, **kwargs)`

Pack object *o* and write it to *stream*

See *Packer* for options.

`dump()` is alias for `pack()`

`msgpack.packb(o, **kwargs)`

Pack object *o* and return packed bytes

See *Packer* for options.

`dumps()` is alias for `packb()`

`msgpack.unpack(stream, object_hook=None, list_hook=None, bool use_list=1, encoding=None, unicode_errors='strict', object_pairs_hook=None, ext_hook=ExtType, Py_ssize_t max_str_len=2147483647, Py_ssize_t max_bin_len=2147483647, Py_ssize_t max_array_len=2147483647, Py_ssize_t max_map_len=2147483647, Py_ssize_t max_ext_len=2147483647)`

Unpack an object from *stream*.

Raises *ValueError* when *stream* has extra bytes.

See *Unpacker* for options.

`load()` is alias for `unpack()`

`msgpack.unpackb(packed, object_hook=None, list_hook=None, bool use_list=1, encoding=None, unicode_errors='strict', object_pairs_hook=None, ext_hook=ExtType, Py_ssize_t max_str_len=2147483647, Py_ssize_t max_bin_len=2147483647, Py_ssize_t max_array_len=2147483647, Py_ssize_t max_map_len=2147483647, Py_ssize_t max_ext_len=2147483647)`

Unpack packed\_bytes to object. Returns an unpacked object.

Raises *ValueError* when *packed* contains extra bytes.

See *Unpacker* for options.

`loads()` is alias for `unpackb()`

**class** msgpack.**Packer** (*default=None, encoding='utf-8', unicode\_errors='strict', use\_single\_float=False, bool autoreset=1, bool use\_bin\_type=0, bool strict\_types=0*)

MessagePack Packer

usage:

```
packer = Packer()
astream.write(packer.pack(a))
astream.write(packer.pack(b))
```

Packer's constructor has some keyword arguments:

#### Parameters

- **default** (*callable*) – Convert user type to builtin type that Packer supports. See also simplejson's document.
- **encoding** (*str*) – Convert unicode to bytes with this encoding. (default: 'utf-8')
- **unicode\_errors** (*str*) – Error handler for encoding unicode. (default: 'strict')
- **use\_single\_float** (*bool*) – Use single precision float type for float. (default: False)
- **autoreset** (*bool*) – Reset buffer after each pack and return its content as *bytes*. (default: True). If set this to false, use *bytes()* to get content and *.reset()* to clear buffer.
- **use\_bin\_type** (*bool*) – Use bin type introduced in msgpack spec 2.0 for bytes. It also enables str8 type for unicode.
- **strict\_types** (*bool*) – If set to true, types will be checked to be exact. Derived classes from serializable types will not be serialized and will be treated as unsupported type and forwarded to default. Additionally tuples will not be serialized as lists. This is useful when trying to implement accurate serialization for python types.

**bytes** (*self*)

Return buffer content.

**pack** (*self, obj*)

**pack\_array\_header** (*self, long long size*)

**pack\_ext\_type** (*self, typecode, data*)

**pack\_map\_header** (*self, long long size*)

**pack\_map\_pairs** (*self, pairs*)

Pack *pairs* as msgpack map type.

*pairs* should be a sequence of pairs. (*len(pairs)* and *for k, v in pairs:* should be supported.)

**reset** (*self*)

Clear internal buffer.

**class** msgpack.**Unpacker** (*file\_like=None, Py\_ssize\_t read\_size=0, bool use\_list=1, object\_hook=None, object\_pairs\_hook=None, list\_hook=None, encoding=None, unicode\_errors='strict', int max\_buffer\_size=0, ext\_hook=ExtType, Py\_ssize\_t max\_str\_len=2147483647, Py\_ssize\_t max\_bin\_len=2147483647, Py\_ssize\_t max\_array\_len=2147483647, Py\_ssize\_t max\_map\_len=2147483647, Py\_ssize\_t max\_ext\_len=2147483647*)

Streaming unpacker.

arguments:

#### Parameters

- **file\_like** – File-like object having `.read(n)` method. If specified, unpacker reads serialized data from it and `feed()` is not usable.
- **read\_size** (*int*) – Used as `file_like.read(read_size)`. (default: `min(1024**2, max_buffer_size)`)
- **use\_list** (*bool*) – If true, unpack msgpack array to Python list. Otherwise, unpack to Python tuple. (default: True)
- **object\_hook** (*callable*) – When specified, it should be callable. Unpacker calls it with a dict argument after unpacking msgpack map. (See also `simplejson`)
- **object\_pairs\_hook** (*callable*) – When specified, it should be callable. Unpacker calls it with a list of key-value pairs after unpacking msgpack map. (See also `simplejson`)
- **encoding** (*str*) – Encoding used for decoding msgpack raw. If it is None (default), msgpack raw is deserialized to Python bytes.
- **unicode\_errors** (*str*) – Used for decoding msgpack raw with `encoding`. (default: `'strict'`)
- **max\_buffer\_size** (*int*) – Limits size of data waiting unpacked. 0 means system's INT\_MAX (default). Raises `BufferFull` exception when it is insufficient. You should set this parameter when unpacking data from untrusted source.
- **max\_str\_len** (*int*) – Limits max length of str. (default: `2**31-1`)
- **max\_bin\_len** (*int*) – Limits max length of bin. (default: `2**31-1`)
- **max\_array\_len** (*int*) – Limits max length of array. (default: `2**31-1`)
- **max\_map\_len** (*int*) – Limits max length of map. (default: `2**31-1`)

example of streaming deserialize from file-like object:

```
unpacker = Unpacker(file_like)
for o in unpacker:
    process(o)
```

example of streaming deserialize from socket:

```
unpacker = Unpacker()
while True:
    buf = sock.recv(1024**2)
    if not buf:
        break
    unpacker.feed(buf)
    for o in unpacker:
        process(o)
```

**feed** (*self, next\_bytes*)

Append *next\_bytes* to internal buffer.

**read\_array\_header** (*self, write\_bytes=None*)

assuming the next object is an array, return its size *n*, such that the next *n* `unpack()` calls will iterate over its contents.

Raises `OutOfData` when there are no more bytes to unpack.

**read\_bytes** (*self, Py\_ssize\_t nbytes*)

Read a specified number of raw bytes from the stream

**read\_map\_header** (*self*, *write\_bytes=None*)

assuming the next object is a map, return its size *n*, such that the next  $n * 2$  `unpack()` calls will iterate over its key-value pairs.

Raises *OutOfData* when there are no more bytes to unpack.

**skip** (*self*, *write\_bytes=None*)

Read and ignore one object, returning `None`

If *write\_bytes* is not `None`, it will be called with parts of the raw message as it is unpacked.

Raises *OutOfData* when there are no more bytes to unpack.

**tell** (*self*)

**unpack** (*self*, *write\_bytes=None*)

Unpack one object

If *write\_bytes* is not `None`, it will be called with parts of the raw message as it is unpacked.

Raises *OutOfData* when there are no more bytes to unpack.

**class** `msgpack.ExtType`

`ExtType` represents ext type in msgpack.

## 1.1 exceptions

These exceptions are accessible via `msgpack` package. (For example, `msgpack.OutOfData` is shortcut for `msgpack.exceptions.OutOfData`)

**exception** `msgpack.exceptions.BufferFull`

Bases: `msgpack.exceptions.UnpackException`

**exception** `msgpack.exceptions.ExtraData` (*unpacked*, *extra*)

Bases: `msgpack.exceptions.UnpackValueError`

**exception** `msgpack.exceptions.OutOfData`

Bases: `msgpack.exceptions.UnpackException`

**exception** `msgpack.exceptions.PackException`

Bases: `Exception`

Deprecated. Use `Exception` instead to catch all exception during packing.

**exception** `msgpack.exceptions.PackOverflowError`

Bases: `msgpack.exceptions.PackValueError`, `OverflowError`

`PackOverflowError` is raised when integer value is out of range of msgpack support  $[-2^{31}, 2^{32})$ .

Deprecated. Use `ValueError` instead.

**exception** `msgpack.exceptions.PackValueError`

Bases: `msgpack.exceptions.PackException`, `ValueError`

`PackValueError` is raised when type of input data is supported but it's value is unsupported.

Deprecated. Use `ValueError` instead.

**exception** `msgpack.exceptions.UnpackException`

Bases: `Exception`

Deprecated. Use `Exception` instead to catch all exception during unpacking.

**exception** `msgpack.exceptions.UnpackValueError`

Bases: `msgpack.exceptions.UnpackException`, `ValueError`

Deprecated. Use `ValueError` instead.



**m**

msgpack, 3

msgpack.exceptions, 6



## B

BufferFull, 6  
bytes() (msgpack.Packer method), 4

## E

ExtraData, 6  
ExtType (class in msgpack), 6

## F

feed() (msgpack.Unpacker method), 5

## M

msgpack (module), 3  
msgpack.exceptions (module), 6

## O

OutOfData, 6

## P

pack() (in module msgpack), 3  
pack() (msgpack.Packer method), 4  
pack\_array\_header() (msgpack.Packer method), 4  
pack\_ext\_type() (msgpack.Packer method), 4  
pack\_map\_header() (msgpack.Packer method), 4  
pack\_map\_pairs() (msgpack.Packer method), 4  
packb() (in module msgpack), 3  
Packer (class in msgpack), 3  
PackException, 6  
PackOverflowError, 6  
PackValueError, 6

## R

read\_array\_header() (msgpack.Unpacker method), 5  
read\_bytes() (msgpack.Unpacker method), 5  
read\_map\_header() (msgpack.Unpacker method), 5  
reset() (msgpack.Packer method), 4

## S

skip() (msgpack.Unpacker method), 6

## T

tell() (msgpack.Unpacker method), 6

## U

unpack() (in module msgpack), 3  
unpack() (msgpack.Unpacker method), 6  
unpackb() (in module msgpack), 3  
Unpacker (class in msgpack), 4  
UnpackException, 6  
UnpackValueError, 6