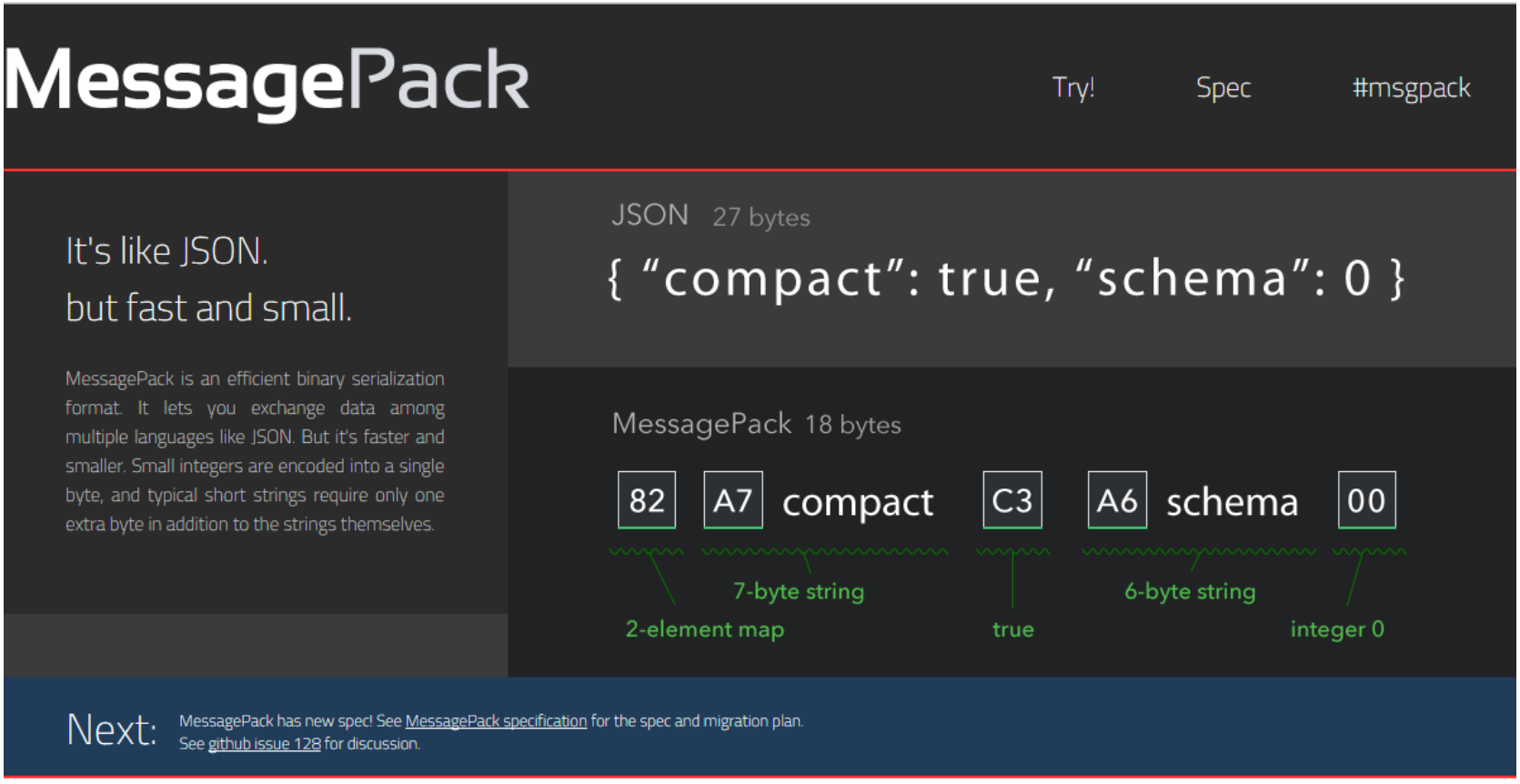
MessagePack(msgpack): A Compact and Fast Serialization Library

Takatoshi Kondo

About me

- Taka (Takatoshi Kondo)
  - from TOKYO, JAPAN
- OGIS-RI Co., Ltd.
  - Developing Pub/Sub IoT Platform using MessagePack
- A committer on the msgpack-c OSS project
- Other experience: Wrote the “Boost.MSM Guide”
  - http://redboltz.wikidot.com/boost-msm-guide

What is MessagePack?



redis

"Redis scripting has support for MessagePack because it is a fast and compact serialization format with a simple to implement specification. I liked it so much that I implemented a MessagePack C extension for Lua just to include it into Redis."

fluentd
Log everything in JSON

"Fluentd uses MessagePack for all internal data representation. It's crazy fast because of zero-copy optimization of msgpack-ruby. Now MessagePack is an essential component of Fluentd to achieve high performance and flexibility at the same time."

Facebook Messenger
By Facebook, Inc.
Open iTunes to buy and download

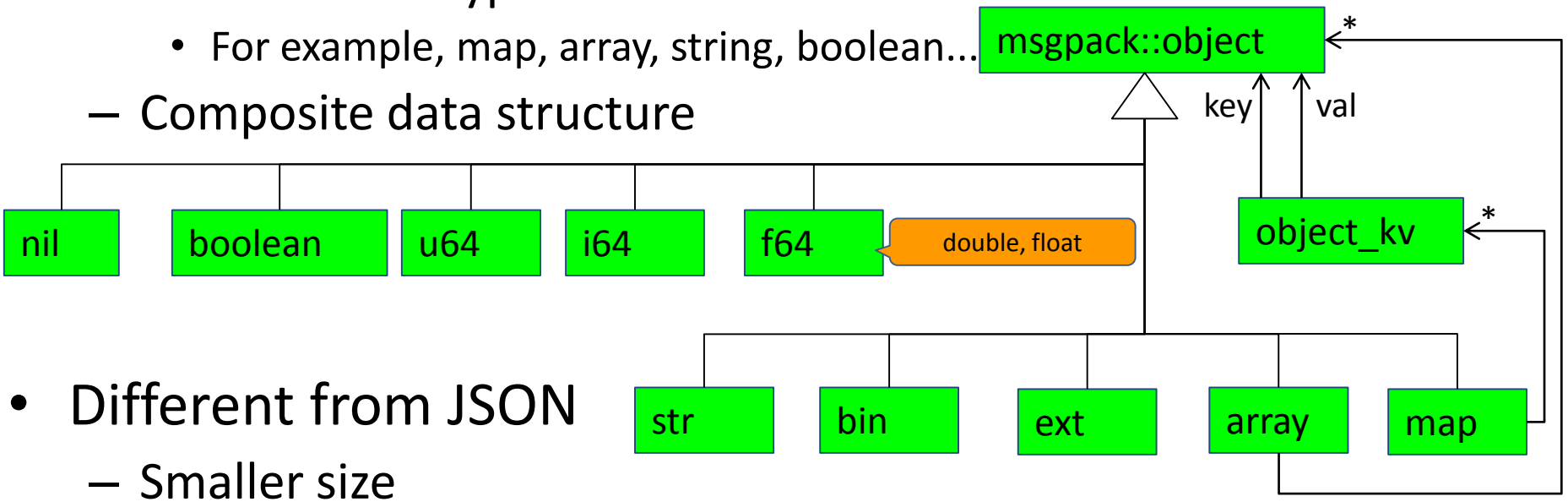
MessagePack vs. JSON

- Same as JSON
  - Portable
  - Contain basic type information
    - For example, map, array, string, boolean...
  - Composite data structure

- Different from JSON
  - Smaller size
  - Binary coded
  - Can handle binary data without text encoding such as Base64
  - Easier to parse, requires less computing power

MessagePack vs. JSON

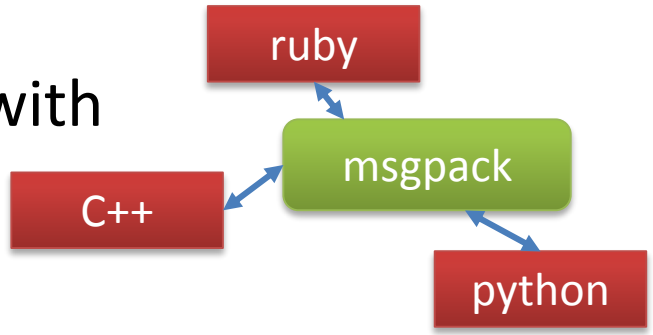
- Same as JSON
  - Portable
  - Contain basic type information
    - For example, map, array, string, boolean...
  - Composite data structure



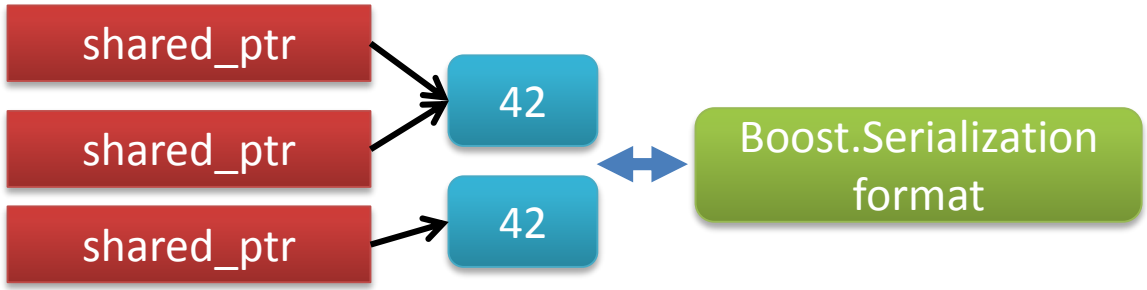
- Different from JSON
  - Smaller size
  - Binary coded
  - Can handle binary data without text encoding such as Base64
  - Easier to parse, requires less computing power

MessagePack/Boost.Serialization

- The goals of the msgpack and the Boost.Serialization are different.
  - msgpack: Interexchange the data with different programming languages.

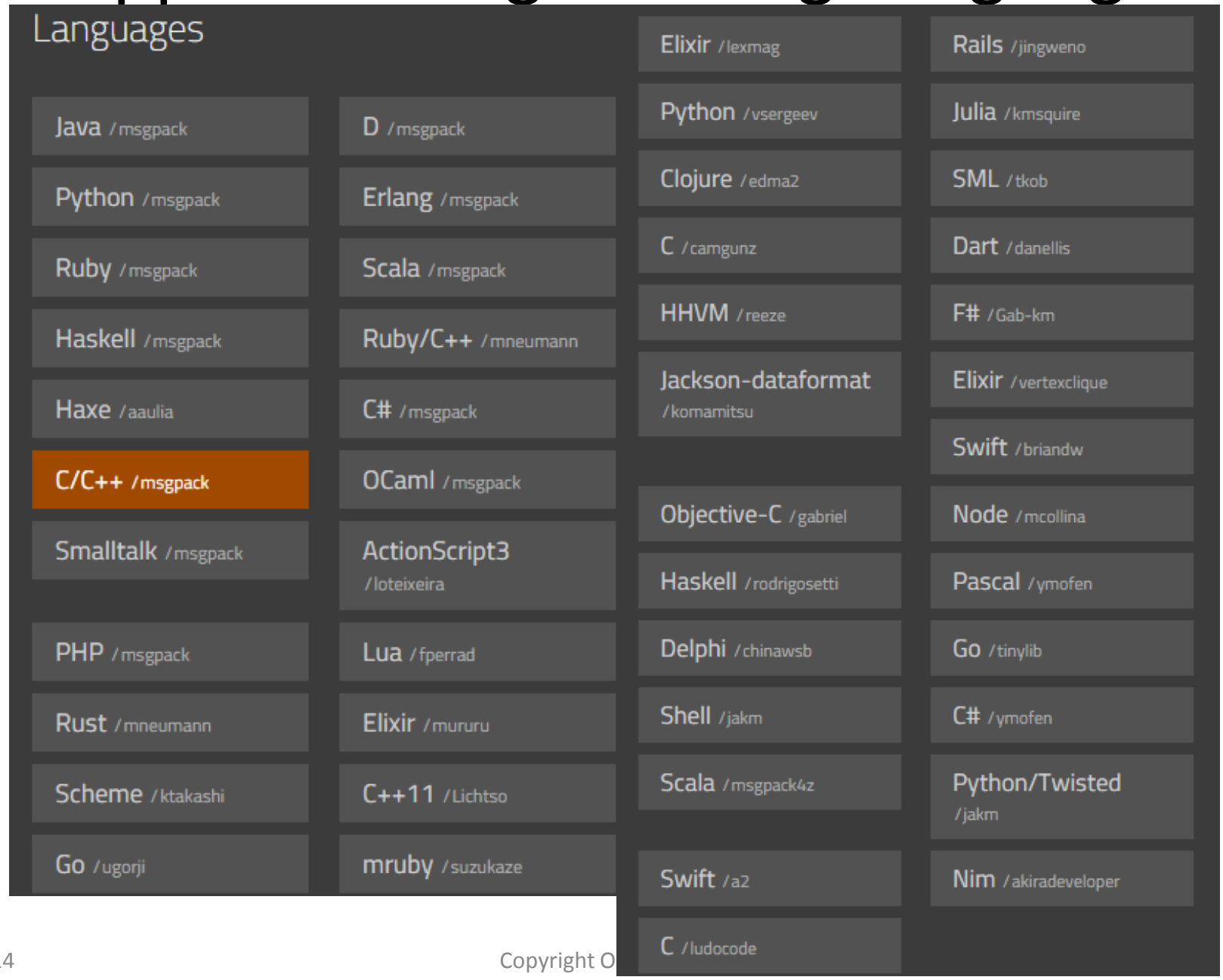


  - Boost.Serialization: Serialize every data and relations.
    e.g.) shared_ptr



- msgpack can be used as a portable binary archive of the Boost.Serialization.
  - https://github.com/redboltz/rui/tree/support_boost_1_57_0
    - **Forked from Norihisa Fujita's implementation**

Supported Programming Languages



Pack/Unpack

```
#include <tuple>
#include <string>
#include <sstream>
#include <msgpack.hpp>

int main() {
    auto t1 = std::make_tuple("hello", true, 42, 12.3);
    std::stringstream ss;
    msgpack::pack(ss, t1);

    msgpack::unpacked unpacked
        = msgpack::unpack(ss.str().data(), ss.str().size());
    msgpack::object obj = unpacked.get();

    auto t2 = obj.as<std::tuple<std::string, bool, int, double>>();
    assert(t1 == t2);
}
```

packing

You can use any types that have the member function write(const char*, std::size_t);

unpacking

convering

Stream Deserialization

```
class unpacker {
public:
    // Constructor is omitted in this presentation

    void reserve_buffer(std::size_t size);
    char* buffer();
    void buffer_consumed(std::size_t size);
    bool next(unpacked& result);
};
```

Stream Deserialization

```
std::size_t const try_read_size = 100;
msgpack::unpacker unp;


while (/* block until input becomes readable */) {
    unp.reserve_buffer(try_read_size);


    std::size_t actual_read_size = input.readsome(
        unp.buffer(), try_read_size);



    unp.buffer_consumed(actual_read_size);

    msgpack::unpacked result;
    // MessagePack data loop
    while(unp.next(result)) {
        msgpack::object obj(result.get());

    }
}
```

Stream Deserialization

```
std::size_t const try_read_size = 100;
msgpack::unpacker unp;


while (/* block until input becomes readable */) {
    unp.reserve_buffer(try_read_size);


    std::size_t actual_read_size = input.readsome(
        unp.buffer(), try_read_size);



    unp.buffer_consumed(actual_read_size);

    msgpack::unpacked result;
    // MessagePack data loop
    while(unp.next(result)) {
        msgpack::object obj(result.get());

    }
}
```

The size may decided by receive performance, transmit layer's protocol and so on.

Stream Deserialization

```
std::size_t const try_read_size = 100;
msgpack::unpacker unp;


while (/* block until input becomes readable */) {
    unp.reserve_buffer(try_read_size);

    std::size_t actual_read_size = input.readsome(
        unp.buffer(), try_read_size);



    unp.buffer_consumed(actual_read_size);

    msgpack::unpacked result;
    // MessagePack data loop
    while(unp.next(result)) {
        msgpack::object obj(result.get());

    }
}
```

The size may decided by receive performance, transmit layer's protocol and so on.

unp has at least try_read_size buffer on this point.

Stream Deserialization

```
std::size_t const try_read_size = 100;
msgpack::unpacker unp;


while (/* block until input becomes readable */) {
    unp.reserve_buffer(try_read_size);

    std::size_t actual_read_size = input.readsome(
        unp.buffer(), try_read_size);

    unp.buffer_consumed(actual_read_size);

    msgpack::unpacked result;
    // MessagePack data loop
    while(unp.next(result)) {
        msgpack::object obj(result.get());

    }
}
```

The size may decided by receive performance, transmit layer's protocol and so on.

unp has at least try_read_size buffer on this point.

input is a kind of I/O library object.
read message to msgpack::unpacker's internal buffer directly.

Stream Deserialization

```
std::size_t const try_read_size = 100;
msgpack::unpacker unp;


while (/* block until input becomes readable */) {
    unp.reserve_buffer(try_read_size);

    std::size_t actual_read_size = input.readsome(
        unp.buffer(), try_read_size);

    unp.buffer_consumed(actual_read_size);

    msgpack::unpacked result;
    // MessagePack data loop
    while(unp.next(result)) {
        msgpack::object obj(result.get());

    }
}
```

The size may decided by receive performance, transmit layer's protocol and so on.

unp has at least try_read_size buffer on this point.

input is a kind of I/O library object.
read message to msgpack::unpacker's internal buffer directly.

notify msgpack::unpacker actual consumed size.

Stream Deserialization

```
std::size_t const try_read_size = 100;
msgpack::unpacker unp;


while (/* block until input becomes readable */) {
    unp.reserve_buffer(try_read_size);

    std::size_t actual_read_size = input.readsome(
        unp.buffer(), try_read_size);

    unp.buffer_consumed(actual_read_size);

    msgpack::unpacked result;
    // MessagePack data loop
    while(unp.next(result)) {
        msgpack::object obj(result.get());

    }
}
```

The size may decided by receive performance, transmit layer's protocol and so on.

unp has at least try_read_size buffer on this point.

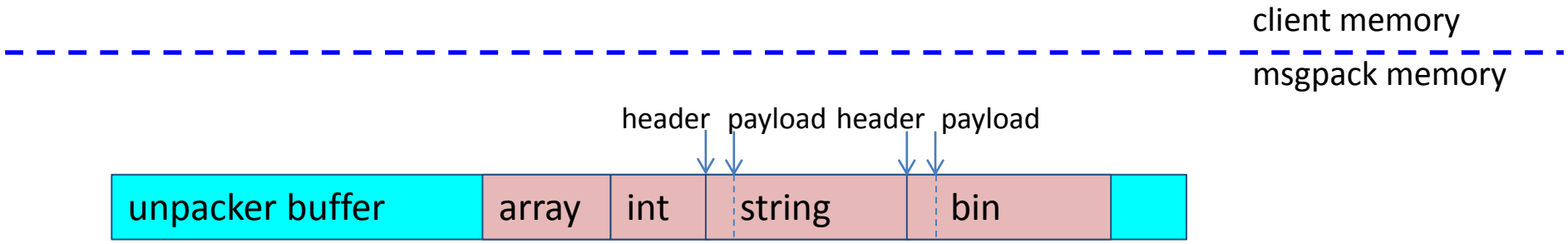
input is a kind of I/O library object.
read message to msgpack::unpacker's internal buffer directly.

notify msgpack::unpacker actual consumed size.

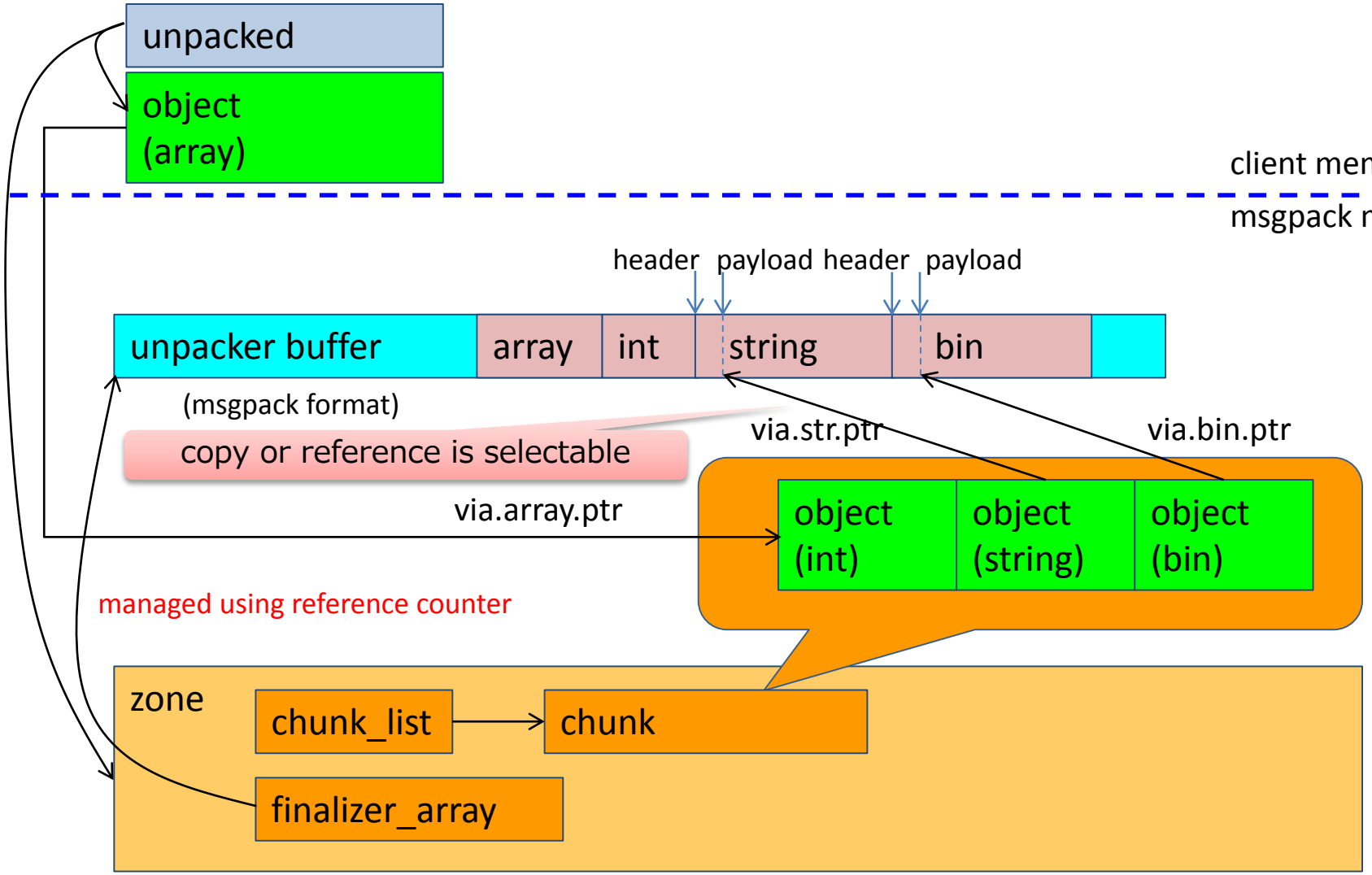
Use obj. convert to C++ types

All complete msgpack message is processed at this point, then continue to read addtional message.

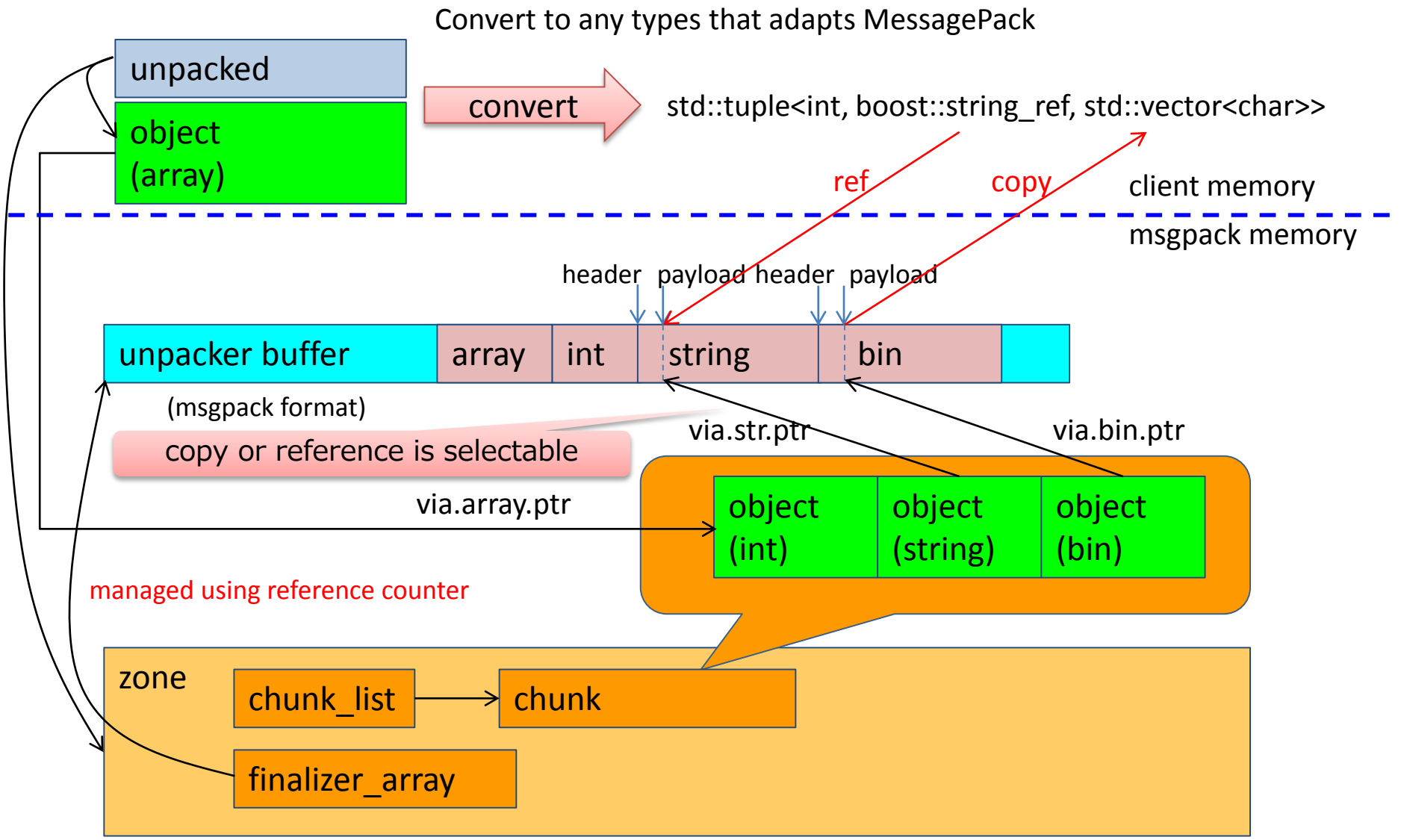
Zero-Copy Deserialization



Zero copy deserialization



Zero copy deserialization



MessagePack Adaptors

C++ type	msgpack::object type
bool	bool
char*	str
std::deque	array
char	positive/negative integer
signed ints *1	positive/negative integer
unsigned ints *2	positive integer
std::list	array
std::map	array
std::pair	array
std::set	array
std::string	str
std::vector	array
std::vector<char>	bin

C++11 type	msgpack:: object type
std::array	array
std::array<char>	bin
std::forward_list	array
std::tuple	array
std::array	array
std::unordered_map	array
std::unordered_set	array

boost type	msgpack:: object type
boost::optional<T>	T
boost::string_ref	str

*1 signed ints signed char, signed short, signed int, signed long, signed long long
*2 unsigned ints unsigned char, unsigned short, unsigned int, signed long, signed long long

https://github.com/msgpack/msgpack-c/wiki/v1_1_cpp_adaptor

MessagePack Adaptors

```
#include <msgpack.hpp>

struct your_class : base1, base2 {
    int a;
    std::string b;

    // You can choose any order.
    // It is represented to the msgpack array elements order.
    MSGPACK_DEFINE(a, b, MSGPACK_BASE(base1), MSGPACK_BASE(base2));
};
```

https://github.com/msgpack/msgpack-c/wiki/v1_1_cpp_adaptor

Thank you

- If you have questions, feel free to contact me :)
- Takatoshi Kondo
  - redboltz@gmail.com
  - twitter: redboltz
- Resources
  - MessagePack
    - http://msgpack.org/
  - msgpack-c
    - https://github.com/msgpack/msgpack-c
  - msgpack-c Documents
    - https://github.com/msgpack/msgpack-c/wiki
  - msgpack-c stream unpack algorithm
    - A little old but concept is the same
    - http://www.slideshare.net/taka111/msgpackc

Extra Slides

MessagePack Formats

Format name	first byte (in binary)	first byte (in hex)
positive fixint	0xxxxxxx	0x00 - 0x7f
fixmap	1000xxxx	0x80 - 0x8f
fixarray	1001xxxx	0x90 - 0x9f
fixstr	101xxxxx	0xa0 - 0xbf
nil	11000000	0xc0
(never used)	11000001	0xc1
false	11000010	0xc2
true	11000011	0xc3
bin 8	11000100	0xc4
bin 16	11000101	0xc5
bin 32	11000110	0xc6
ext 8	11000111	0xc7
ext 16	11001000	0xc8
ext 32	11001001	0xc9
float 32	11001010	0xca
float 64	11001011	0xcb
uint 8	11001100	0xcc
uint 16	11001101	0xcd
uint 32	11001110	0xce
uint 64	11001111	0xcf

Format name	first byte (in binary)	first byte (in hex)
int 8	11010000	0xd0
int 16	11010001	0xd1
int 32	11010010	0xd2
int 64	11010011	0xd3
fixext 1	11010100	0xd4
fixext 2	11010101	0xd5
fixext 4	11010110	0xd6
fixext 8	11010111	0xd7
fixext 16	11011000	0xd8
str 8	11011001	0xd9
str 16	11011010	0xda
str 32	11011011	0xdb
array 16	11011100	0xdc
array 32	11011101	0xdd
map 16	11011110	0xde
map 32	11011111	0xdf
negative fixint	111xxxxx	0xe0 - 0xff

https://github.com/msgpack/msgpack/blob/master/spec.md

MessagePack Formats

Format name	first byte (in binary)	first byte (in hex)
positive fixint	0xxxxxxx	0x00 - 0x7f
fixmap	1000xxxx	0x80 - 0x8f
fixarray	1001xxxx	0x90 - 0x9f
fixstr	101xxxxx	0xa0 - 0xbf
nil	11000000	0xc0
(never used)	11000001	0xc1
false	11000010	0xc2
true	11000011	0xc3
bin 8	11000100	0xc4
bin 16	11000101	0xc5
bin 32	11000110	0xc6

https://github.com/msgpack/msgpack/blob/master/spec.md

MessagePack Format

int format family

https://github.com/msgpack/msgpack/blob/master/spec.md#int-format-family

Int format family stores an integer in 1, 2, 3, 5, or 9 bytes.

```
positive fixnum stores 7-bit positive integer
+--------+
|0XXXXXXX|
+--------+

negative fixnum stores 5-bit negative integer
+--------+
|111YYYYY|
+--------+

* 0XXXXXXX is 8-bit unsigned integer
* 111YYYYY is 8-bit signed integer

uint 8 stores a 8-bit unsigned integer
+--------+--------+
|  0xcc  |ZZZZZZZZ|
+--------+--------+

uint 16 stores a 16-bit big-endian unsigned integer
+--------+--------+--------+
|  0xcd  |ZZZZZZZZ|ZZZZZZZZ|
+--------+--------+--------+
```

MessagepPack format

int format family

Int format family stores an integer in 1, 2, 3, 5, or 9 bytes.

```
uint 32 stores a 32-bit big-endian unsigned integer
+--------+--------+--------+--------+--------+
|  0xce  |ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|
+--------+--------+--------+--------+--------+

uint 64 stores a 64-bit big-endian unsigned integer
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
|  0xcf  |ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|ZZZZZZZZ|
+--------+--------+--------+--------+--------+--------+--------+--------+--------+
```

What is MessagePack?

Facebook Messenger

By Facebook, Inc.

Open iTunes to buy and download apps.

View Mo

View in iTunes

This app is designed for both iPhone and iPad

Free

Category: Social Networking
Updated: Apr 24, 2015
Version: 26.0
Size: 66.7 MB
Languages: English, Bokmål, Norwegian, Croatian, Czech, Danish, Dutch, Finnish, French, German, Greek, Hungarian, Indonesian, Italian, Japanese, Korean, Malay, Polish, Portuguese, Russian, Simplified Chinese, Slovak, Spanish, Swedish, Thai, Traditional Chinese, Turkish, Vietnamese, Zulu
Seller: Facebook, Inc.
© Facebook, Inc.
Rated 4+

Compatibility: Requires iOS 7.0 or later. Compatible with iPhone, iPad, and iPod touch. This app is optimized for iPhone 5, iPhone 6, and iPhone 6 Plus.

Description

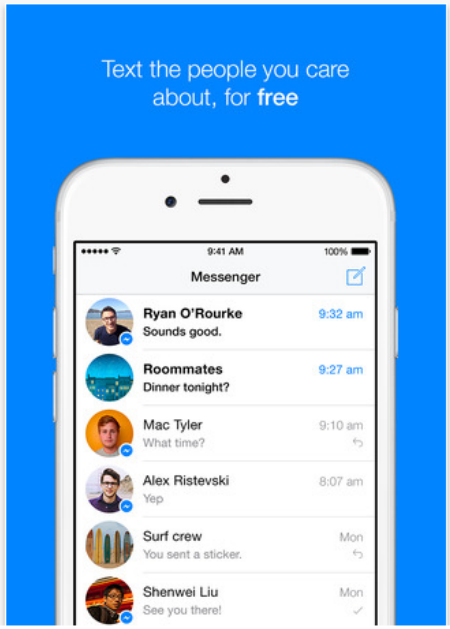
Instantly reach the people in your life—for free. Messenger is just like texting, but you don't hav message (it works with your data plan).

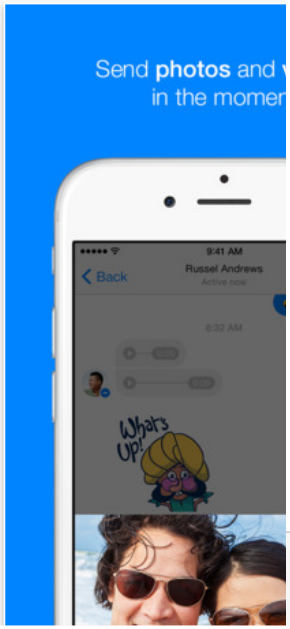
Facebook, Inc. Web Site › Facebook Messenger Support › Application License Agreement ›

What's New in Version 26.0

Fixed issues to make the app faster.

Screenshots





au 23:01 67%

設定 サードパーティ通知

MessagePack

The following software may be included in this product: MessagePack. This software contains the following license and notice below:

Copyright (C) 2008-2010 FURUHASHI Sadayuki

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Apache License Version 2.0, January 2004
http://www.apache.org/licenses/

最近 グループ 友達 設定