QuickStart for C++

Implementation Status

The serialization library is production-ready.

Currently, RPC implementation is testing phase. Requires newer kernel, not running on RHEL5/CentOS5.

Install

Same as C language.

Serialization QuickStart for C++

First program

Include `msgpack.hpp` header and link `msgpack` library to use MessagePack on your program.

```
#include <msgpack.hpp>
#include <vector>
#include <string>
#include <iostream>

int main(void) {
        // serializes this object.
        std::vector<std::string> vec;
        vec.push_back("Hello");
        vec.push_back("MessagePack");

        // serialize it into simple buffer.
        msgpack::sbuffer sbuf;
        msgpack::pack(sbuf, vec);

        // deserialize it.
        msgpack::unpacked msg;
        msgpack::unpack(&msg, sbuf.data(), sbuf.size());

        // print the deserialized object.
        msgpack::object obj = msg.get();
        std::cout << obj << std::endl;  //=> ["Hello", "MessagePack"]

        // convert it into statically typed object.
        std::vector<std::string> rvec;
        obj.convert(&rvec);
}
```

Compile it as follows:

```
$ g++ hello.cc -lmsgpack -o hello
$ ./hello
["Hello", "MessagePack"]
```

Streaming feature

```
#include <msgpack.hpp>
#include <iostream>
#include <string>

int main(void) {
        // serializes multiple objects using msgpack::packer.
        msgpack::sbuffer buffer;

        msgpack::packer<msgpack::sbuffer> pk(&buffer);
        pk.pack(std::string("Log message ... 1"));
        pk.pack(std::string("Log message ... 2"));
        pk.pack(std::string("Log message ... 3"));

        // deserializes these objects using msgpack::unpacker.
        msgpack::unpacker pac;

        // feeds the buffer.
        pac.reserve_buffer(buffer.size());
        memcpy(pac.buffer(), buffer.data(), buffer.size());
        pac.buffer_consumed(buffer.size());

        // now starts streaming deserialization.
        msgpack::unpacked result;
        while(pac.next(&result)) {
            std::cout << result.get() << std::endl;
        }

        // results:
        // $ g++ stream.cc -lmsgpack -o stream
        // $ ./stream
        // "Log message ... 1"
        // "Log message ... 2"
        // "Log message ... 3"
}
```

Streaming into an array or map

```
#include <msgpack.hpp>
#include <iostream>
#include <string>

int main(void) {
        // serializes multiple objects into one message containing an array using msgpack::packer.
        msgpack::sbuffer buffer;

        msgpack::packer<msgpack::sbuffer> pk(&buffer);
        pk.pack_array(3)
        pk.pack(std::string("Log message ... 1"));
        pk.pack(std::string("Log message ... 2"));
        pk.pack(std::string("Log message ... 3"));

        // serializes multiple objects into one message containing a map using msgpack::packer.
        msgpack::sbuffer buffer2;

        msgpack::packer<msgpack::sbuffer> pk2(&buffer2);
        pk2.pack_map(2)
        pk2.pack(std::string("x"));
        pk2.pack(3);
        pk2.pack(std::string("y));
        pk2.pack(3.4321);

}
```

User-defined classes

You can use serialize/deserializes user-defined classes using MSGPACK_DEFINE macro.

```
#include <msgpack.hpp>
#include <vector>
#include <string>

class myclass {
private:
    std::string m_str;
    std::vector<int> m_vec;
public:
    MSGPACK_DEFINE(m_str, m_vec);
};

int main(void) {
        std::vector<myclass> vec;
        // add some elements into vec...

        // you can serialize myclass directly
        msgpack::sbuffer sbuf;
        msgpack::pack(sbuf, vec);

        msgpack::unpacked msg;
        msgpack::unpack(&msg, sbuf.data(), sbuf.size());

        msgpack::object obj = msg.get();

        // you can convert object to myclass directly
        std::vector<myclass> rvec;
        obj.convert(&rvec);
}
```

2 Comments

Jao Rabary Jun 29, 2012

Hi all, I'm trying to pack and unpack array object from ruby to cpp. In the ruby side is use (1.001,2,3.99).to_msgpack. The result is converted into byte string and sent to a cpp code. When I unpack the string in cpp side, unpacked.get() give the correct litteral object. But when I do unpacked.get().as<vector<double> > I got nothing. What can be the problem ?

iha jaa tt Feb 20, 2013

Absolute garbage! Your examples are utterly ridiculous and stupid and pointless. Your taking an sbufffer that you used to pack data and unpacking it. Pointless! When you convert it into something meaningful, it doesn't work!

Can someone who is actually using this trash please give a proper c++ example; eg saving it to a file, and then reading from a file, etc

regards