LOTRO Investigation v0.2 (Riders)

Januar 22

2013

This document should carry together what kind of information is delivered through the LOTRO UDP packets. Some sample packets are analyzed and explained.

Gathering information for building a private LOTRO server.

Table of Contents

List of Authors

tA..... tAmMo

[Please feel free to add your name if you added, changed something or corrected spelling and grammar]

List of Figures

22.01.2013 Update packets to Riders Version

1. Tutorial for packet capturing with Wireshark

1.1 Workflow

Do the following preparations in Wireshark:

(1) Choose interface options

Please select the options for the relevant network interface.

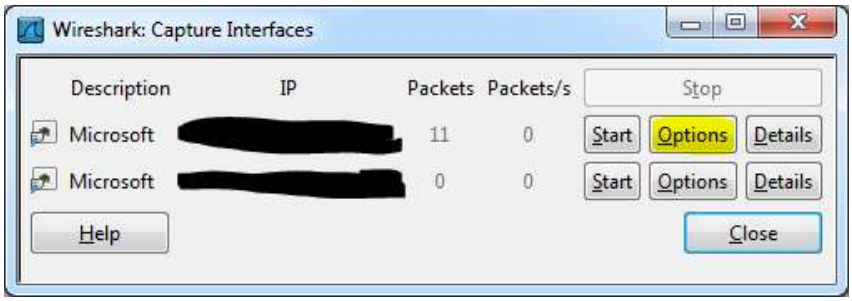

Figure 1: Choose interface options

(2) Apply capture filters

Use the following capture filters:

Not broadcast and not multicast and udp and not port 53 and not port 9200

Click start.

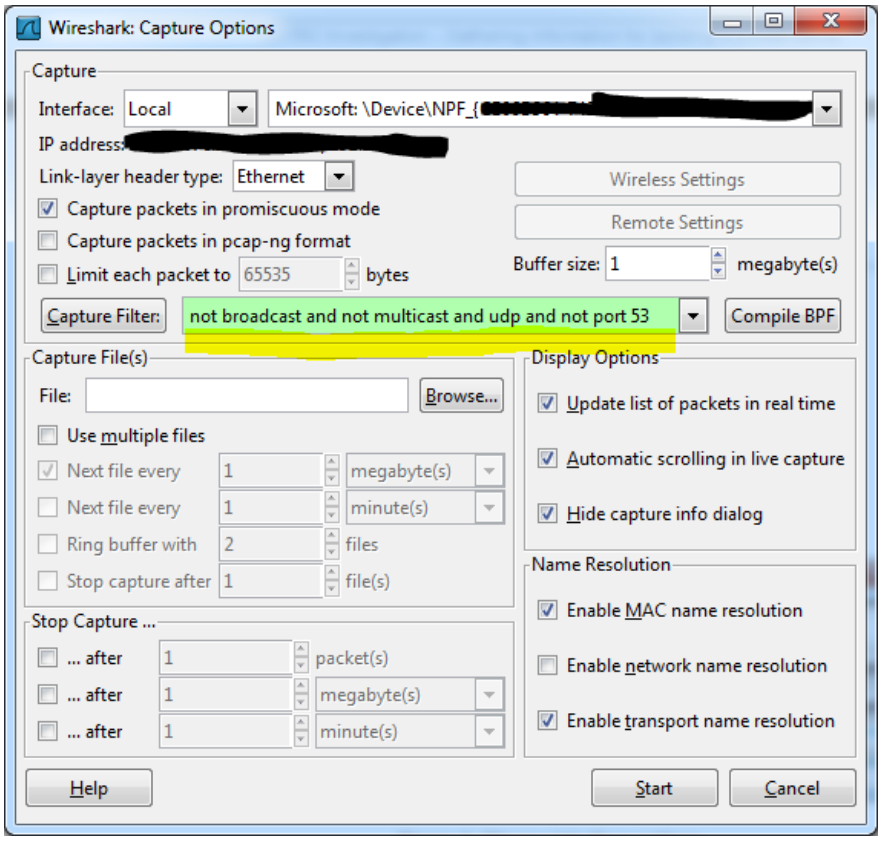

Figure 2: Capture filter

(3) Stop the capture after a while

After some time stop capturing. This should somehow look like this.

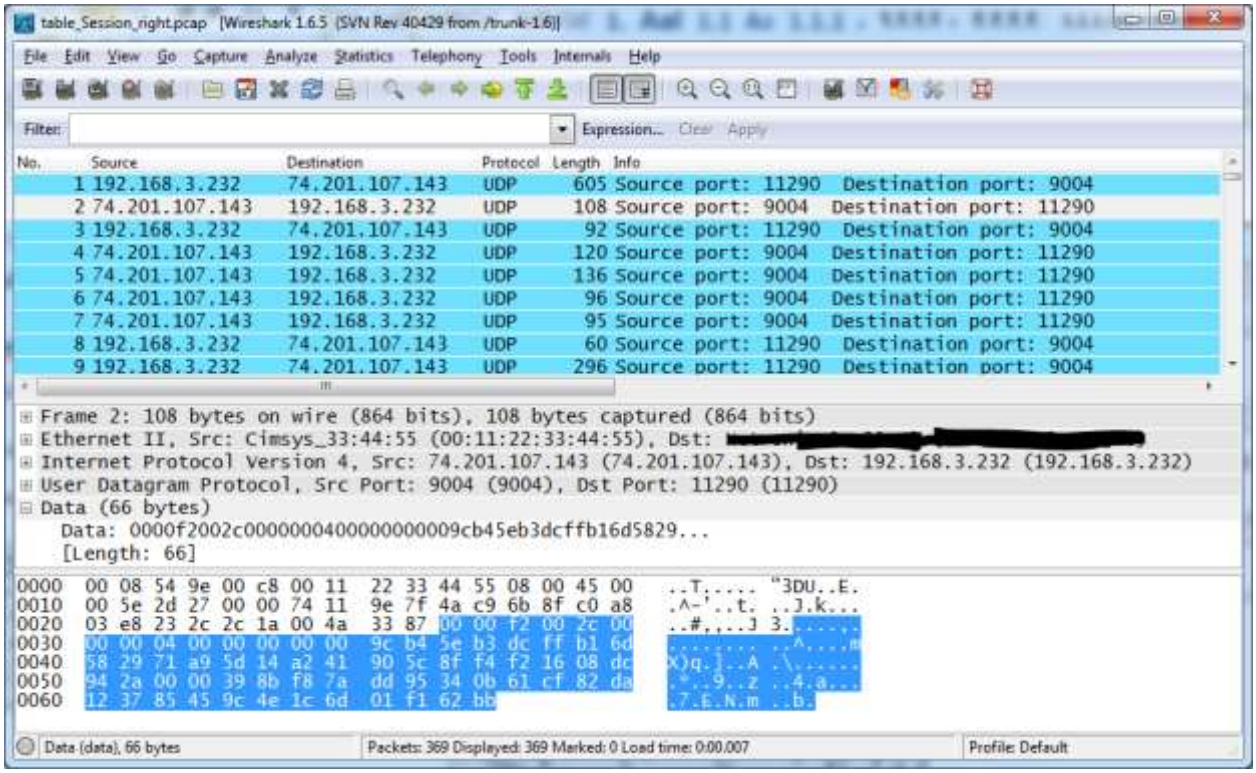

Figure 3: Captured traffic

(4) Rightclick “Follow UDP Stream”

Choose the right direction and click “Save As”.

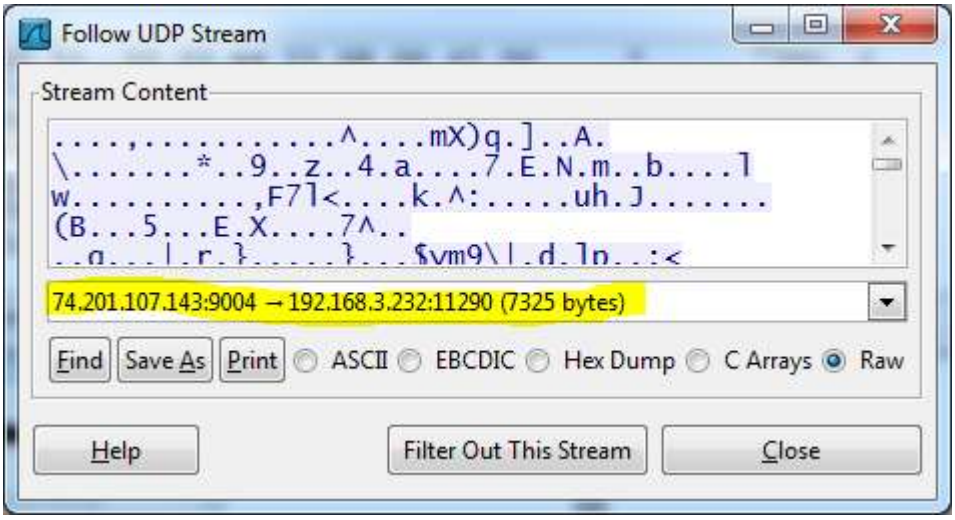

Figure 4: Follow UDP Stream

(5) Use the “CreatePacketsFromWiresharkHexDump”-Tool to batch extract the packets

2. Client -> Server communication flow

Client starts a session request.Server looks if the account was authenticated through web service.
Server responds with temp session key and two 32 bit checksum(bases).
Client sends back the reversed session key and generates the checksum table. Session is now established.

Later:
Server sends authorization and character data, which had been stored during webservice auth.
When user joins world, server sends ip and port of world server.
Client makes new connection to world server where and is inside middle earth.

3. Packet analyze

3.1 General structure

The datagram payload is segmented in two parts:

- **The (pseudo) Header part** – During session setup (the header starts with 0x00 0x00) its length is 22 bytes, in all other cases its 20 bytes long
- **The Data part** – can contain ***zero*** *to* ***n*** data objects (requests)

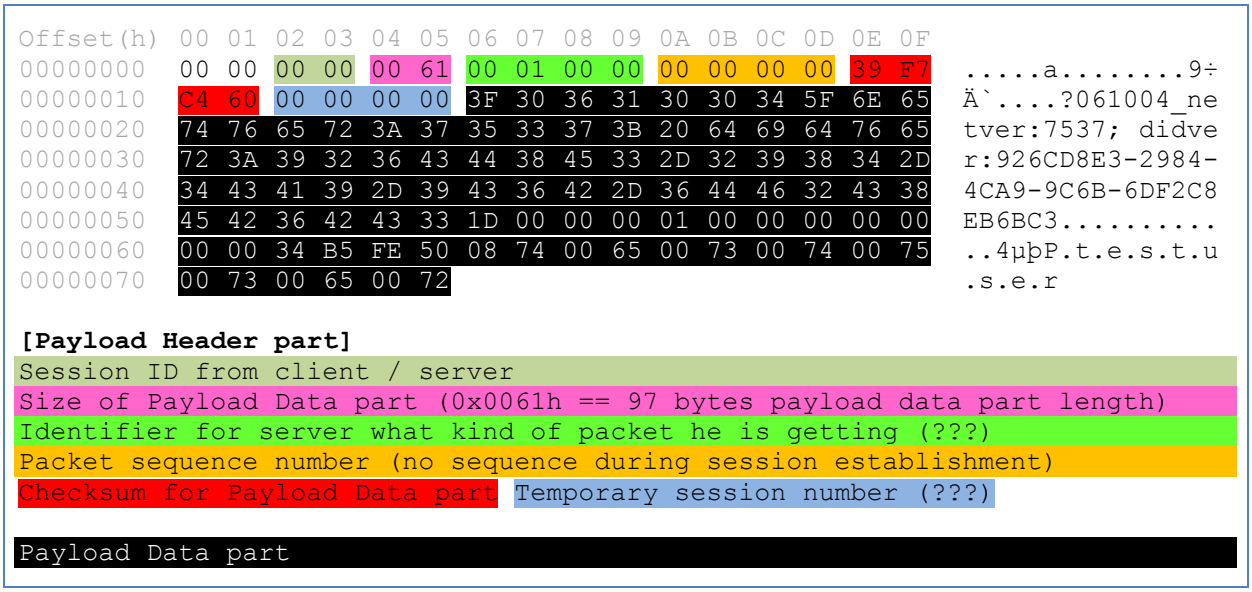

Figure 5: Header from "session setup" packet

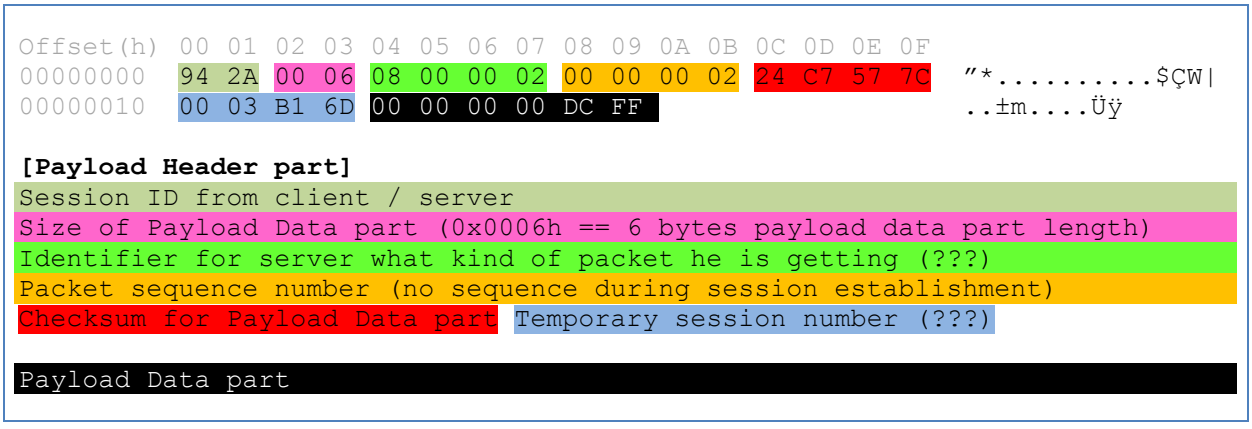

Figure 6: Header from "in session" packet

The (payload) Header part has an Session-ID.

The (payload) Header part has a sequence number.

The (payload) Header part has a checksum.

The (payload) Header part has an temporary session number / ack number.

The length of the Data part is given in the (payload) Header part.

3.2 First Client payload packets

3.2.1 1st client payload packet (this packet is not encrypted)

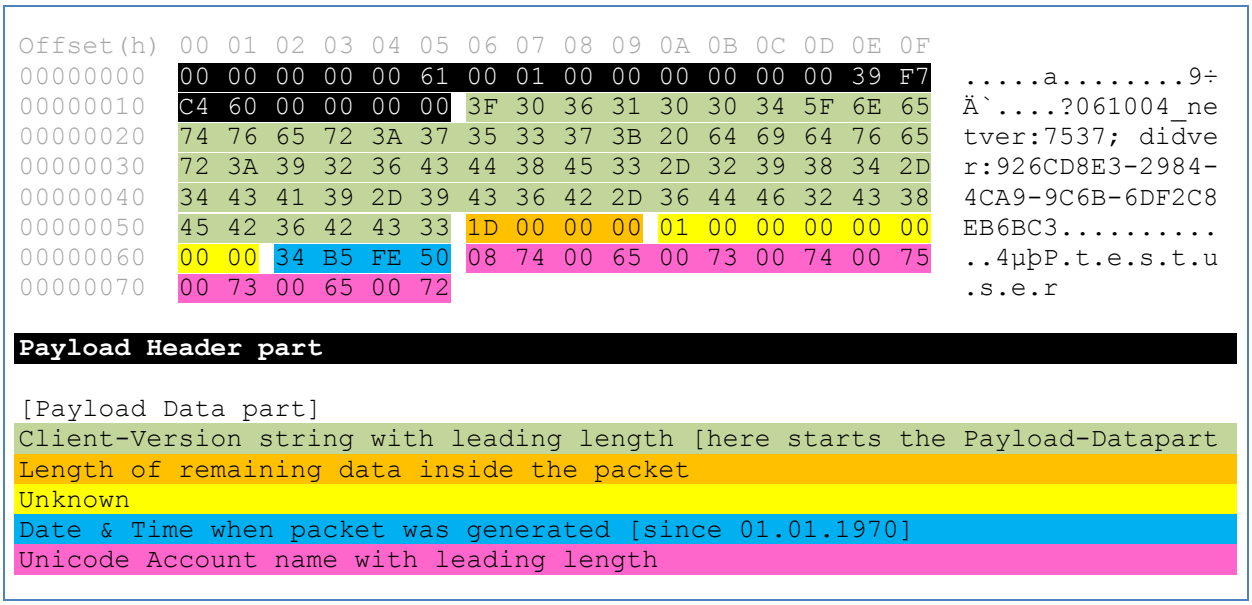

Figure 7: 1st client packet

Investigations:

- The first part of the Client-ID String may change after software update.
- Date & Time as 4 Byte unsigned Integer value is the time in seconds since 01.01.1970 ~~(reverse it to 0x4F480235 = 1330119221 seconds. Add the seconds to 01.01.1970 and you get 24.02.2012 21:33:41)~~.
- ~~The Session Key Block contains the 128 Bit **Session Key**. The Session Key Block is encrypted with the Clients Public Key. The Public Key size is 1024 Bit and he is known. The Session Key Block can be decrypted with the Servers **Private Key** to gain the **Session Key**. The **Private Key** is only known at the Game Server and can't be brute forced. It would take ages. The Account name + GLS Ticket Block is **RC4** encrypted with the **Session Key**. In this Account name + GLS Ticket Block only the Unicode string of fictional account name "tammo" is encrypted (Unicode: 2 bytes per character plus the leading length specification of the string = 11 Bytes or 0x0B in Hex).~~

More?

3.2.2 2nd client payload packet (this packet is not encrypted)

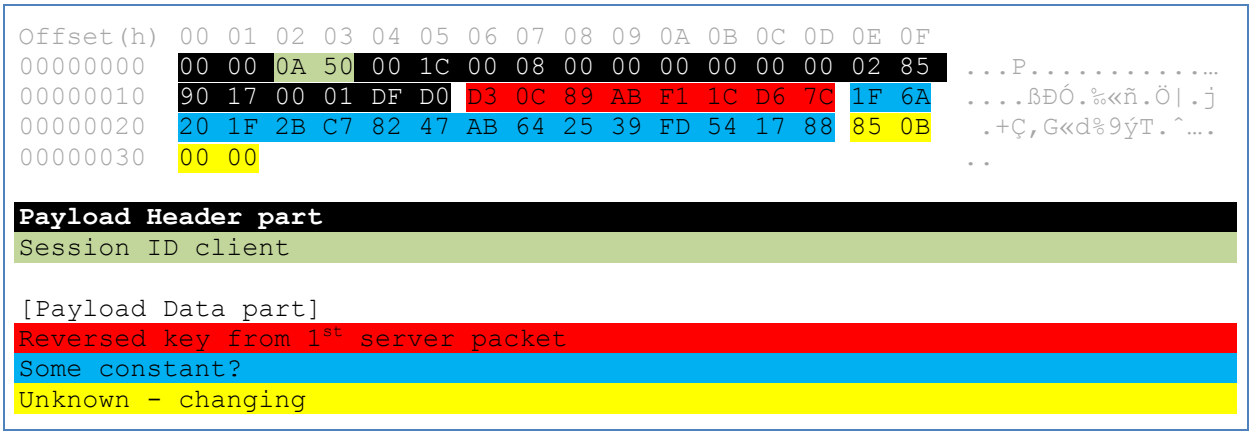

Figure 8: 2nd client packet

Investigations:

- Client returns reversed session key from 1st server packet.

3.3 First Server payload packets

3.3.1 1st server payload packet (this packet is not encrypted)

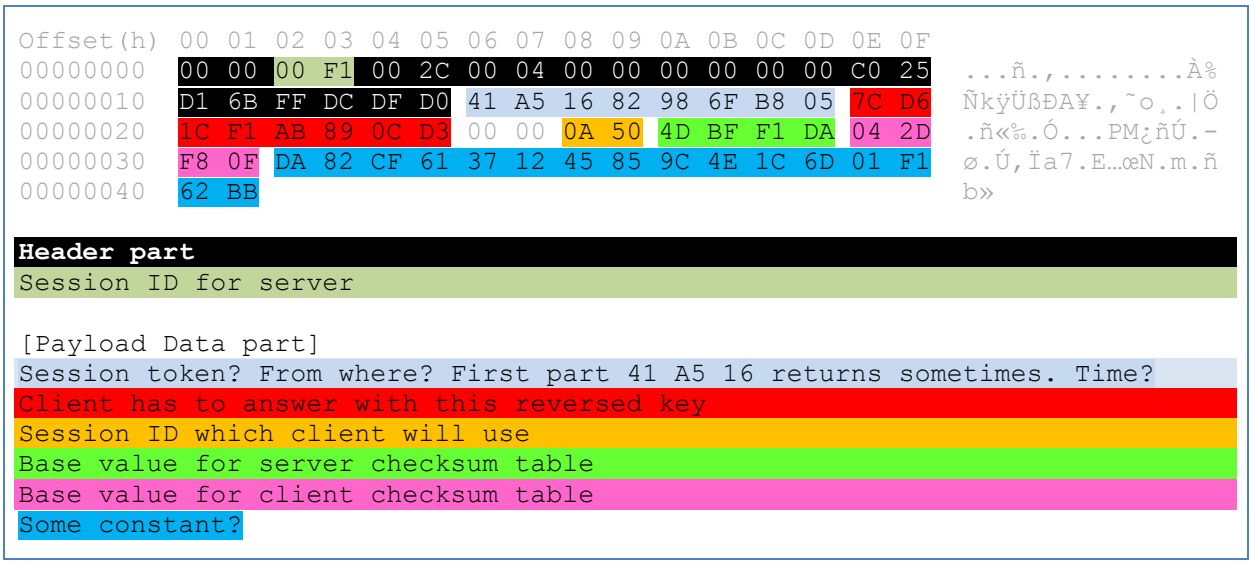

Figure 9: 1st server packet

Investigations:

- Server submits two values for checksum tables.
- Server submits client the session id, he should use.

3.3.2 2nd server payload packet (from now packets are encrypted)

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   00 F1 00 61 01 00 00 06 00 00 00 02 11 D3 5D DB   .ñ.a.........Ó]Û
00000010   FF DD DF D0 41 A5 16 82 98 B7 DB B5 01 02 02 04   ÿÝßÐA¥.‚˜·Ûµ....
00000020   00 00 00 00 00 06 81 49 26 00 00 08 2D 01 0E 33   .......I&...-..X
00000030   00 31 00 2E 00 31 00 35 00 30 00 2E 00 32 00 30   .X...X.X.X...X.X
00000040   00 35 00 2E 00 32 00 31 00 38 00 01 00 00 00 00   .X...X.X.X......
00000050   01 02 02 04 00 00 00 00 00 06 80 AA 17 00 00 08   ..........€ª....
00000060   00 00 00 00 00 00 00 00 9E 81 E5 0A FE 01 00 25   ........ž.å.þ..%
00000070   01 00 00 00 00                                    .....
```

Header part

Session ID for server

[Payload Data part]

Session key? From where? First part 41 A5 16 returns [01 in Header]

Indicator for Data part object (???)

Length for Data part object

[Changed my IP-Address inside the packet to XX.XXX.XX.XXX]

Figure 10: 2nd server packet

Investigations:

- Inside this packet you will find your IP-Address
- The other values are unknown to me

3.4 Other packets

3.4.1 Client confirms server session

```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000   0A 50 00 15 08 00 00 06 00 00 00 07 16 64 7E BA   .P..........d~°
00000010   00 03 DF D0 CD 01 00 00 DD FF 01 00 03 04 00 00   ..ßÐÍ...Ýÿ......
00000020   00 00 00 06 80 F5 01 00 00                       ....€õ...
```

Header part

[Payload Data part]

Data length from session packet

Temporary session ID

Figure 11: Client packet which confirms data length from last temp session

3.4.2 Client confirm/request last/next sequence packet

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000  0A 50 00 04 00 00 40 00 00 00 00 08 C5 4B 90 C9  .P....@.....ÅK.É
00000010  00 05 DF D0 00 00 00 10                          ..ßÐ....

Header part

[Payload Data part]
Sequence number to confirm/request next?
```

Figure 12: Server pong packet

Investigations:

- This seems to be a client ping packet, or a confirm packet

List of Links

Document Attachments

Source code for:

- Packet extraction from Wireshark stream
- Client packet de- and encryption
- Server packet de- and encryption
- Simple server-test program for your own logged packets(no multithread, not thread save, not performance optimized)

Please feel free to mod!