IM protocol-draft V 0.1

Date: 2010-06-29 version: 0.1 Author: William Lv

Description: Message protocol.

Msg : Message

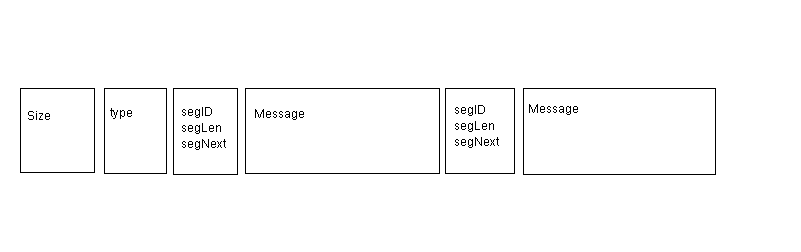
Seg: Segment

Len: Length

1. Msg protocol

Head content:

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Data type | Max Len | Description |
| Size | quint32 | 2^32 | Whole message length |
| Type | quint8 | 2^8 | Message Type |
| SegID | quint8 | 2^8 | Segment ID |
| SegLen | quint16 | 2^16 | Segment length |
| SegNext | quint8 | 2^8 | If has next segment |
| Message |  |  | Message input by user/server |



1. Size (total size in byte)
2. Type :

Default value: 0x00

|  |  |  |
| --- | --- | --- |
| 0x00 | Default value |  |
| 0x01 | Syn message, sent by server and client automaitcly to check connection status.More details inform written in first Message segment. |  |
| 0x02 | Login message, sent by client and the server sent 0x03 as response. |  |
| 0x04 | Sent by server to tell client that connection accpected and server is ready for next message. | Mutex, can’t appear with 0x08 in one message. |
| 0x08 | Sent by server to tell client that connection build failed. Failed reason should be written in Message segment. | Mutex, can’t appear with 0x04 in one message. |
| 0x10 | User message. |  |
| 0x20 | Reserve for next version. |  |
| 0x30 | Reserve for next version. |  |
| 0x40 | Reserve for next version. |  |

除了标成黄色的不能同时出现，其他的0Xxx可以同时出现，多种类型直接相与。提高传输效率。

例如：

一条消息包含：同步+用户发送的信息 格式如下：

0X01 & 0X10 = 0X11

0X11 --十六进制 00010001 -- 二进制

C． SegID

标识每一段Message segment 。

minimum：0x00

maximum：0x0F

1. SegLen

Message segment length.

E. SegNext

0XFF means no message segments behind.

Otherwise more message segment follow behind.

Example:

void MainWindow::sendData()

{

QByteArray block;

QDataStream out(&block, QIODevice::WriteOnly);

out.setVersion(QDataStream::Qt\_4\_6);

quint32 blockSize = 0;

quint8 blockType = 0;

quint8 segID = 0;

quint32 segLen = 0;

quint8 segNext = 0xFF;

QString content = msgEdit->toPlainText();

segLen = content.size() + sizeof(segLen) + sizeof(segID);

out<<blockSize<<blockType<<segID<<segLen<<segNext<<content;

out.device()->seek(0);

out<<quint32(block.size() - sizeof(quint32));

tcpSocket->write(block);

}

1. Message

Length: Message segment length (size in byte)

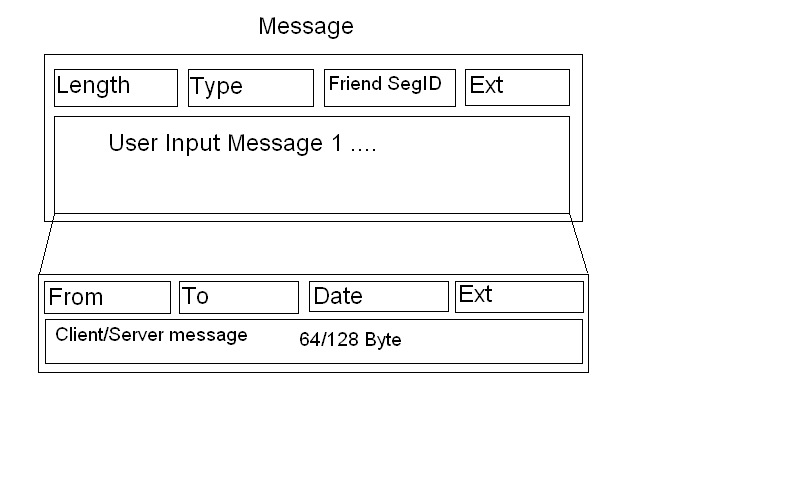
Type:

|  |  |  |
| --- | --- | --- |
| 0x00 | Default value |  |
| 0x01 | P2P Message | Person to person message. Include simple text message and multi-media message. |
| 0x02 | P2M | One to multi message. |
| 0x04 | SCM | Server control message. |
| Other for reservation |  |  |
|  |  |  |
|  |  |  |

Friend SegID:

One message segment may be not enough to carry all data. Friend SegID tell client that this message should be operated with following message segments identified with Friend SegID.

Extension: Reserve for next version.

1. May used as link priority.
2. May used as connection method. ( A. One time connection; B. Temporary/short connection; C. Long time connection )
3. May used as contain more control/other informati

Message maximum size (in bytes) =

4(size) + 1(type) + 1(Segment ID) + 2(Segment length) +

[2(Length in message segment) + 1(type) + 1(Friend SegID) + 4(Ext) +

4(From) + 4(To) + 4(Date) + 4(Ext) + 128(User/Server input message)] \* 0x0F = 8 + 2432 = 2440Byte

16 Client (Server) messages / Maximum Size with 16 message fragments

= 2048 / 2440 = 0.84

Message minimum size (in bytes) =

4(size) + 1(type) + 1(Segment ID) + 2(Segment length) +

2(Length in message fragment) + 1(type) + 1(Friend SegID) + 4(Ext) +

4(From) + 4(To) + 4(Date) + 4(Ext) + 128(User/Server input message)

= 8 + 8 + 16 + 128 = 160

One Client (Server) message / Minimum Size with one message fragment

= 128 / 160 = 0.8