The Instant Message eXtension

This subsystem is intended to provide an asynchronous instant communication between one-to-one or one-to-many users. The notion of ‘asynchronous’ here means that the data flows at the same time from the client to the server and the other way around. The notion of ‘instant’ is relative because it depends on how the module is set up and how big the interval will be at the server side. This interval, that the user can tune, is the time in which the server will sleep before searching for new messages to send them to the client.

XSIME_START

XSIME_START indicates to the server that the Instant Message eXtension module -now on IME- should be initialized. The return status for this command could be:

OK
FAILED 1 : Thread allocation failed.

Until here the rules of the TF PROTOCOL main module apply.

If the command has succeed, the dynamics of sending and receiving data, as the buffers sizes, will change completely.
First of all, this module use two threads/processes at the same time. One thread should be used for reading the data from the server and the other one to write the data to the server.
The buffer has a maximum size, which is 16 * 1024 bytes, and must be two of them -one for receiving and one for sending-. However, each transmission does not consume that much, so, somehow this

variable size should be announced to the peer (the server or the client). This is where the header enters the stage.

The IME header as it's defined in C.

```
struct xsime_hdr {
   char opcode;
   int32_t sz;
};
```

The IME header has as first member a byte that indicates the opcode -operation code- which is the way the parties specify the type of the operation. The second member is a signed 32bit integer. The total amount of bytes for this header should be 5 bytes. Some programming language should be careful with this due some optimizations that could lead to a greater size. This it's known as padding. This header is send it prior the buffer.

The second member, as it is the memory representation of an integer, it has some endianness. As it is known some machines are big-endian and others are little-endian. To avoid this problem of memory representation of data across machines, this number must be 'normalized' to a specific endianness before it can be transmitted. The IME uses big-endian -or network byte order- to transmit this data. In consequence, the computer at the client side must convert the number -if needed- to the actual memory model before sending it or after receiving.

Once the **sz** member of the header is converted to the machine specific endianness, it contains the size -in bytes- of the data that the server will send after. The size never will exceed the maximum possible size of the buffer.

The value of **sz** is used to know how many bytes should be read back from the network stack.

Sometimes the server will send only a header without sending data after. This type of transmission is used to indicate a return status of a previous opcode sent by the client. In the same way some times the data sent to the server could be only the header, it depends of the opcode.

The OPCODES

The code is the value of the **opcode** member of the IME header structure.

Opcode ‘4’: Setup the IME module.
Opcode ‘1’: Set organizing unit.
Opcode ‘2’: Set user.
Opcode ‘5’: Send message.
Opcode ‘7’: Change sleep interval.
Opcode ‘0’: Exit IME.

In the future others opcodes can be defined.

There is a particular order in which these opcodes must be executed.

4: To set up the module configuration.
1: To set the organizing unit; this is: the folder of all users.
2: To set the specific user’s folder in the organizing unit.

After this, one can call 5 to send messages; 7 to change sleep interval; 0 to exit IME. Even it’s possible to call 4 again to change the configuration of the module.

One particularity of IME is that you can exit the module with the opcode 0 and reenter it with the command **XSIME_START**. This is different; for example, from the Notification mechanism of the TF PROTOCOL, that once you enter it, there is no way back.
Now, if you exit the module and reenter it, it’s not necessary to re-setup. The IME module remembers the last configuration, except in the case that you exit the entire communication of TF PROTOCOL.

Opcode 4

This opcode can encode in the **sz** member of the header two values with the | operator.

1: Unique user.
2: Auto delete.

The 1 option -if set- indicates to the server that only one instance of a particular user can be logged at the same time. This is a cooperative protocol, it means that all involved instances of a particular user must set it on, otherwise it won’t work.
The 2 option -if set- indicates to the server that every messages after receiving in the client side, must be delete in the server user’s folder.

After header has been sent, an unsigned 64bit number -8 bytes-, equally ‘normalized’ to big-endian must be send it. This number represents a timestamp, it tells to the server the point in the time continuum from which the server must start to look for messages. 0 means the beginning of time.

This opcode can be invoked every time the users wants to change the configuration and/or the timestamp.

Opcode 1

This opcode is used to indicate to the server the path where all the users are.

After header been sent, it goes the path early mentioned. The **sz** member should have the size of the payload; this is: the size of the path.
This opcode has a return value from the server indicating the success or failure of the operation.
The server only returns a header with the opcode 1 and value of 0 -for success- or -1 -for failure- in the **sz** member.

Opcode 2

This opcode is used to indicate to the server the path of the user’s folder inside the organizing unit. The maximum allowed size for the user’s name it’s 128 bytes.

After header has been sent, it goes the path early mentioned. The **sz** member should have the size of the payload; this is: the size of the path.
This opcode has a return value from the server indicating the success or failure of the operation.
The server only returns a header with the opcode 2 and value of 0 -for success- or -1 -for failure- in the **sz** member.

The specified user’s folder must exist. Inside of it, the IME module creates automatically -if does not exist- a folder for messages labeled as ‘messages’ and inside of this one, another named ‘blacklist’ -if does not exist too-.
Inside this ‘blacklist’ folder the client can use other commands of the TF PROTOCOL to create files with the name of the desired blocked users. By means of this, a client can block incoming messages from a particular user. Theses messages are silently discarded.

Opcode 5

This opcode is used to send a message to one or many user/s.

The **sz** member of the header it contains the size of the payload; this is: the amount of bytes that will be send it after the header. The syntax for sending messages is:

user1:user2: “message to users”

Each user should be separated by colon ‘:’ including the last one of them. Then comes a whitespace and then the message.
Once again, the total size can’t exceed the maximum length allowed by the buffer, which in this version of IME is 16 * 1024.
If there is no indicated user, or no message is specified, the message is silently discarded.

At the same time, the saver can send messages to the client too. The opcode of the header will be 5. The **sz** member will contain the size of the payload that will be send it after. In this case, the payload will be the message that the server sent. It looks like this:

timestamp:user:message

The **timestamp** represents the moment in which the server received the message, not the moment in which the user send it. This timestamp must be used only for the time line of messages. Suppose that the module is configured to no delete messages after send them, in this case the client can uses the last timestamp receive it from the server to indicate to the server the point -in time- to start looking for new messages. In the case that the module is set up to auto delete messages after sending them to the client, it’s not necessary to take into account the timestamp because in the user’s folder will be only new messages.

The **user** indicates who sends the message.
The **message** it’s the text that the **user** send it.

A word of caution:
If you set the module to allow multiple instances of the same user you should turn off auto delete. This is necessary because some instances of the user could receive the message and some not. In this case, the timestamp allows you to specify the next message -in the time line- you want to receive. As the server does not delete it after sending it, other instances of the user can request the same message too.

If you only use one instance of the user, it’s safe to set auto delete on. In this case, you don’t need to take into account the timestamp at all. Just set it to 0 when execute the opcode 4.

Opcode 7

This opcode is used to set the sleep interval of the server. This interval is the amount of time in which the server will not look for new messages to send them to the client.

The interval goes encoded in an unsigned 64bit integer send it after the header. This number must be converted to network byte-order before it can be transmitted. The time resolution is expressed in nanoseconds. So, 25000000 (25 millions of nanoseconds) are 25 milliseconds. 2500000000 (Two thousand five hundred millions of nanoseconds) are 2 seconds and 500 milliseconds, and so on.

Opcode 0

This opcode is used to exit the IME module and get back to the command interface of the TF PROTOCOL. Once the client send it to server, it must continue receiving data -that potentially is coming- until receives Opcode 0 from the server.