IConverse: Yet Another Instant Messenger?

olibenu@sigma-works.com, about me
13th August, 2012

2go sucks.

If you say ‘really?’, that means you don’t know or haven’t heard about it. If you say ‘of course!’, then you know what I’m saying.

- the app is poor on most platforms. Like Android, Blackberry and even PC.
- it cannot be accessed using a web browser alone
- and, it can be more than it is now.

Due to these reasons, I had proposed an API called 2go4x (on SourceForge) which would have enabled people to write better clients and improve on 2go. I decompiled an older 2go java client and started writing the resulting classes in VB.NET.

The problems with reverse engineering are several. One of them is that people frown at it. It is legally a gray area and I don’t want any one suing me as I don’t have an idle lawyer on retainer. Not yet at least. The second is that the path of reverse engineering always plays ‘catch up’ to the innovations of the original product. Why spend time on a service that can be modified the next day? It’s better to get ahead!

I then decided on something different but related. An integration of 2go, Facebook Chat and other IMs like GTalk, WhatsApp and Jabber in one fluid, easy-to-use and increasingly efficient, open source platform. And that was when IConverse was born.

What is IConverse? It all begins with a conversation. It is a model which describes the basic operation of most IM services:

1. Any and every chat falls under a Conversation. A user’s profile is a Conversation between him and his friends (or the whole network for a public profile). Messages between users also implement the Conversation model between the two users and any other person they choose to invite.
2. Conversation topics change. A topic is the summary of the ongoing conversation at points in time. A user’s status changes are different topics of the conversation between that user and the rest of the network.
3. People enter and leave a conversation (or are kicked out ;-). The timestamp defines a user’s access to conversation history.
4. Chat messages are limited to 140 characters. This makes network journeys easier and also enables such messages to optionally flow out of the network via SMS. Users are not restricted to this character length at the front end. Therefore, if source code (or some other lengthy material) is pasted into a chat, it is broken into messages of 140 characters at the back end.
5. Files can be sent. The hash (sha1) is first calculated then the file is broken into blobs of fixed length all having the same file id (and identified by an

incrementing index) and *sent*. The conversation the file belongs to is notified of the completion of the upload (with filename and hash to be used for saving and verification respectively). I also imagine that user profile images can use this. Also a slight deviation of sending the file url (if not local) might be possible.

I plan to implement the demo using a MySql server as a backend without any special server daemons. This is a quick hack and is bound to change. The database tables are as follows:

- User: id (int), username (string, 20), password (string, 40)
- Conversation: id (int, pk), stamp (timestamp, pk), topic (string, 50), user
- ConversationParty: conversation (pk), user, stamp (timestamp), flag (int, 0-arrival, 1-departure)
- Chat: id (int, pk), stamp (timestamp), conversation, message (string, 140)
- File: id (int, pk), index (int, pk), conversation id, content (blob)
- Alias: id (int, pk), service (string, 10), user, userid (string, 45)

The Alias table is for the integration of other IM services. A user can possess several aliases though limited to one for a particular service (for now). This means that supplying a 2go alias enables a 2go extension to the IConverse model (say, 2go4x) to pull in data (messages, status and so on) from the 2go network into IConverse. This enables one to bring other communities such as WhatsApp, Facebook Chat, GTalk and Jabber into IConverse and the creation of private communities too.

In the nearest future, when a better backend than using a MySql server is developed, people interested in owning a community can get the necessary software and host their own server.

IConverse can be more. You might have something in mind that will be a killer feature. There is currently a 2go API (2go4x on SourceForge) and the IConverse files on GitHub. Why not check them out and get cracking?

After all, it all starts with a conversation.