

# Making the iChat server work with MSN

by Filipp Lepalaan <[filipp@mac.com](mailto:filipp@mac.com)> 16.11.2006

The iChat server in 10.4 was a welcome addition to the already impressive list of services bundled with Mac OS X Server. To those who don't know - it is actually the Jabber server which uses the Open Source [XMPP protocol](#) that's also used by Google in their GoogleTalk product. But let's face it, how many IM users out there use the Jabber/XMPP protocol and how many, say something like MSN?

At least here in Europe, MSN is considered practically synonymous with instant messaging and people are usually quite disappointed when they hear they can't use their new IM account to talk to all of their friends on MSN. While using the iChat server "as is" for only internal communications is fine, to make it useful as a general-purpose IM tool, we'd need some broader protocol support, like MSN.

Luckily, when Apple chose to base it's instant messaging implementation on Jabber/XMPP it opened it to a world of possibilities. All we need to get our iChat servers talking to the MSN network are the right tools and a little elbow grease (isn't that the case with pretty much everything?).

We have to install a gateway that would translate between an iChat account an MSN account. This isn't a novel idea at all, a simple Google search will find you a number of servers which may have several of these gateways installed, but using something someone else has made is rarely as much fun as making it yourself! ;-)

These gateways are called *transports* and there's a wide selection of them [out there](#) for different protocols and actually more than one for MSN, but we're going to focus on the most actively developed one (and one that I know supports all the features that I need), namely [PyMSNt](#). As the name suggests, it's written in Python which makes it straightforward to install and run on Mac OS X.

This tutorial would have been much harder to put together without the help of James Bunton's PyMSNt documentation. While all the info here should be sufficient to get things working, you might want to check [his documentation](#) too which focuses only on the transport configuration. I've listed here all the most important commands, skipping the totally obvious ones (like changing directories or untarring).

## System Requirements and Dependencies

We'll need to have XCode installed for the following steps to work. I've tested this with XCode 2.3 (gcc version 4.0.1) running on a G4 PowerMac with 10.4.6 server. You'll also need an MSN passport for testing.

On the networking side of things, you'll need to make sure your server can make outgoing connections on port 443 (HTTPS) and 1863 (MSN) as well as incoming connections on port 8010 (Jabber file transfers).

Once we're done, our iChat server should be capable of doing text messaging with MSN clients, support avatars and, most importantly, file transfers (in my opinion maybe the best application for IM). PyMSNT depends on the [Twisted networking framework](#), [PyOpenSSL](#), the [Python Imaging Library](#) and [PyCrypto](#). To get JPEG avatar support (which most users would expect), we'll also need to install *libjpeg*.

## Installation

First, libjpeg and the Python Imaging Library:

```
$ curl -O http://www.ijg.org/files/jpegsrc.v6b.tar.gz
$ export MACOSX_DEPLOYMENT_TARGET=10.4
$ ln -s `which glibtool` ./libtool
$ cp /usr/share/libtool/config.sub .
$ cp /usr/share/libtool/config.guess .
$ ./configure --enable-shared
```

(just ignore what it says about shared library support not actually being enabled, it'll work, trust me)

```
$ sudo mkdir /usr/local/include
$ make && sudo make install
```

Then we install the PIL:

```
$ curl -O http://effbot.org/downloads/Imaging-1.1.5.tar.gz
$ python setup.py build
```

Make sure that JPEG support is available by checking the final lines of the build phase:

```
-----
--- TKINTER support ok
--- JPEG support ok
--- ZLIB (PNG/ZIP) support ok
*** FREETYPE2 support not available
-----
```

If it is, just go ahead and install (if not, go back and check that the libjpeg install was successful):

```
$ sudo python setup.py install
```

Then, using the exact same method, go ahead and install the [PyCrypto](#) and [PyOpenSSL](#).

As I mentioned before, the Twisted networking framework is the main dependency of PyMSNT, so let's install that next. If you're a Fink user, you can also find it from the "unstable" repository, but it's pretty straight forward to install yourself:

```
$ curl -O http://tmrc.mit.edu/mirror/twisted/Twisted/2.4/Twisted-2.4.0.tar.bz2
```

```
$ python setup.py build
```

Notice the last message:

```
twisted.python.dist module not found. Make sure you have installed the Twisted core package before attempting to install any other Twisted projects.
```

This means we actually need to run this again after we've installed the first pass:

```
$ sudo python setup.py install
```

And now do the same thing once more (this time there won't be any complaints):

```
$ python setup.py build && sudo python setup.py install
```

If you ever feel the need to remove the twisted package, you'll find it in:

```
/System/Library/Frameworks/Python.framework/Versions/2.3/lib/python2.3/site-packages/twisted/
```

And finally PyMSNt, the gateway software responsible for the actual protocol conversion:

```
$ curl -O http://delx.cjb.net/pymsnt/tarballs/pymsnt-0.11.2.tar.gz
```

## Configuring the transport

After unpacking, let's move on to configuring the transport by first copying *config-example.xml* to *config.xml* and editing the config file. Please take a look at all the config directives and adjust accordingly. I will only address the most critical ones here:

```
<jid>msn.host.com</jid>
<host>host.com</host>
<port>5223</port>
<secret>secret</secret>
<spooldir>data</spooldir>
<!-- <background /> -->
```

*JID* (Jabber ID, also called the Service ID) is just a name for the transport the server will identify it by, unlike *<host>*, it doesn't have to be a FQDN. *5223* is the port that the iChat server runs on. *<secret>* can be anything as long as the same thing is later set in *jabber.xml* (we'll look at that in a moment).

I commented *<background />* just for aiding in the initial testing, we'll change it back later. You can also set the log level (*<debugLevel>*) to something more useful if you run into trouble.

The spool directory is pretty important. PyMSNt will create a directory named after the JID that contains, the MSN credentials of every user who's registered with the transport, in an XML file (*username%host.com*) as *clear text*. While this may not necessarily jeopardize

our server, we should still take special care in protecting this directory (which we'll do in just a minute).

Oh, and if You're wondering what the `<reactor>` setting does, go find out more about Twisted's reactors from [this link](#).

As a side-note, when looking for information or add-on software for you iChat server, remember that it's based on the original [1.x Jabber spec](#).

## "We're commenting these out, of course :)"

Now that our gateway's configured, it's time to let the iChat server know about it. Let's start by defining the service in `jabber.xml`:

```
$ sudo -u jabber nano /etc/jabber/jabber.xml
```

As you can see, that file is nicely commented and reading it can tell quite a lot about the iChat server. You can just edit the following in right after the `<jabber>` tag:

```
<!-- MSN transport -->
<service id="msn.host.com">
  <host>msn.host.com</host>
  <accept>
    <ip>127.0.0.1</ip>
    <port>5223</port>
    <secret>secret</secret>
  </accept>
</service>
```

The service id and `<host>` should be the same as the "jid" you set earlier in the transport configuration and `<ip>` should equal `<mainServer>`. Now let's scroll down to the `<browse>` section and inside it, add the following. Again, "jid" is the service ID we defined earlier:

```
<!-- More MSN transport stuff -->
<service type="msn" jid="msn.host.com" name="MSN Transport">
  <ns>jabber:iq:register</ns>
  <ns>jabber:iq:gateway</ns>
</service>
```

The most important thing about all these settings is that they match. Oh, and if you're wondering about the title of this paragraph, search for it in `jabber.xml` ;-)

Save the file and restart the iChat server:

```
$ sudo restart-service ichat
```

"restart-service" is just a Bash shell function I defined like so:

```
$ restart-service () { serveradmin stop $1 && serveradmin start $1; }
```

## Testing

We should be ready to take our gateway for a test drive, so lets fire it up:

```
$ ./PyMSNt
```

To check that it actually works, we'll need an advanced Jabber client, like [Psi](#) that can also do Jabber service browsing and transport registration. After connecting right-click on your account and select *Actions > Service Discovery*. You should see something similar to this:

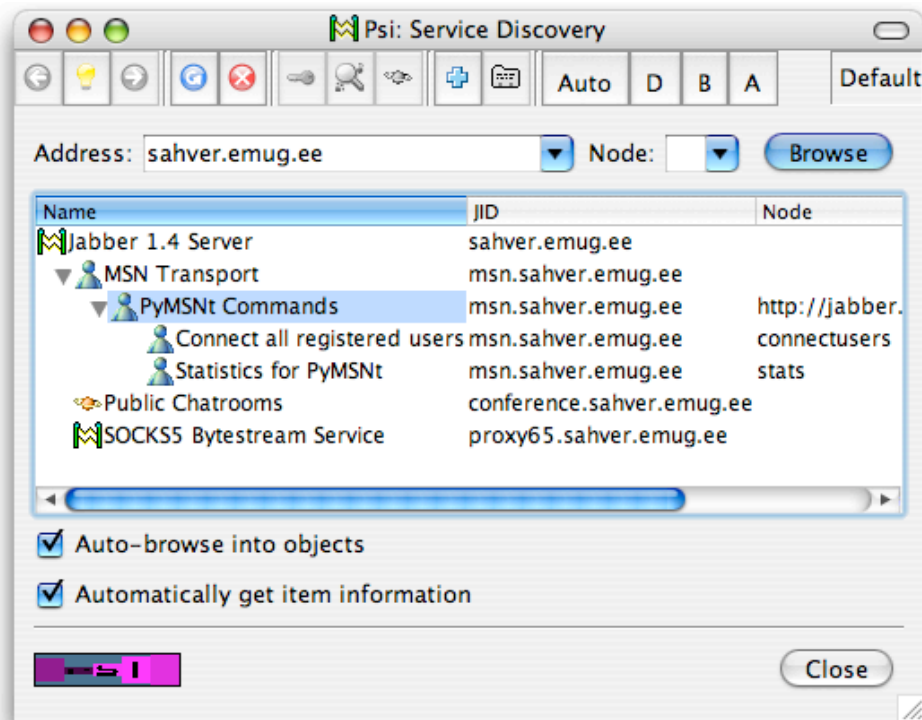


Figure 1: The PyMSNt gateway in Psi's Service Discovery

If you can't see *MSN Transport* in the list, double check that PyMSNt is running and that the configurations set in the previous section match. If you fix a mistake in configuration, just restart the transport, you don't have to restart the iChat server itself.

Right-click on *MSN Transport* and select *Register*. This is where you'll bind your iChat account with your MSN account.

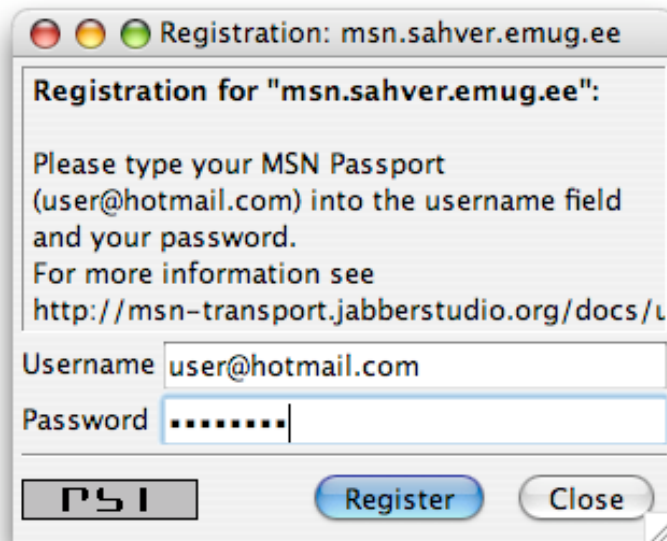


Figure 2: MSN user account registration

The way all of this works is that every user on your iChat server can bind their iChat account to a specific MSN account. So if *usera@ourserver* logs in with Psi and registers *whatever@hotmail.com* with the MSN transport, all of their MSN contacts would be visible through our iChat server. Likewise the MSN users will see *usera* as just another MSN user. The MSN users never see, nor can they directly talk to, the iChat server account, just the iChat user's MSN account.

To differentiate between Jabber and MSN users on the server, the MSN contacts will have a weird naming scheme. For example, to add *joe@schmoe.com*, with our MSN JID set to *msn.host.com*, we'd actually add *joe%schmoe.com@msn.host.com*. Psi will do this name conversion for you when select the Service ID from the service menu, but it's important to keep it mind when you want to use some other IM client.

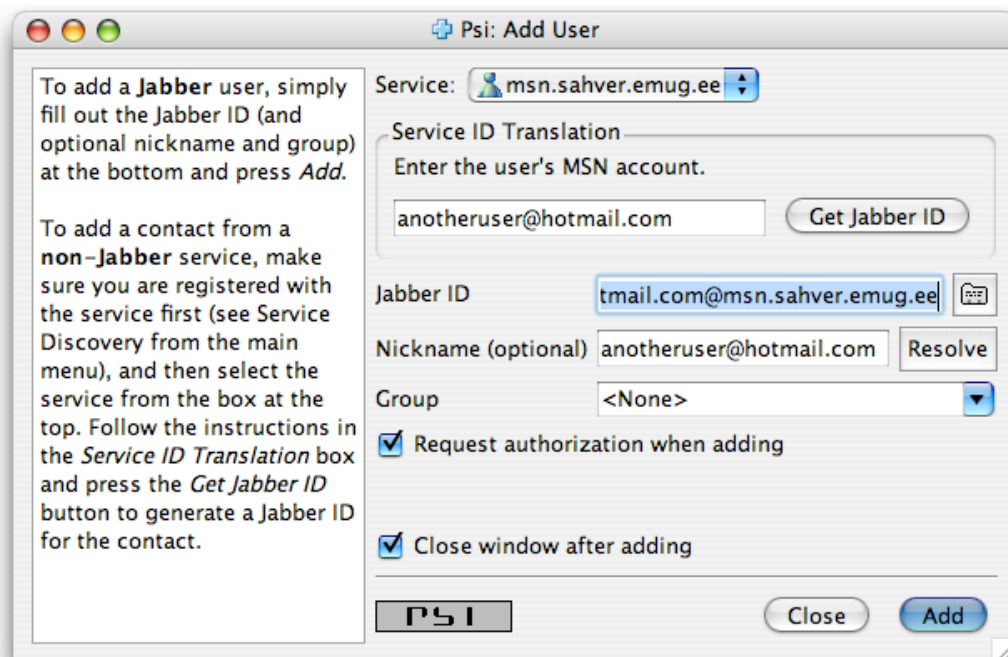


Figure 3: Adding an MSN contact in Psi

This means you can add MSN contacts with any client, but you need something heavier (like Psi) to actually register yourself with the transport. Because this requires each user's MSN credentials, we unfortunately can't automate this process and so every user will have to do this themselves. We could make this easier by creating a web-page from which they could bridge their IM accounts, by copying the XML transactions, but that's something I'll leave for next time.

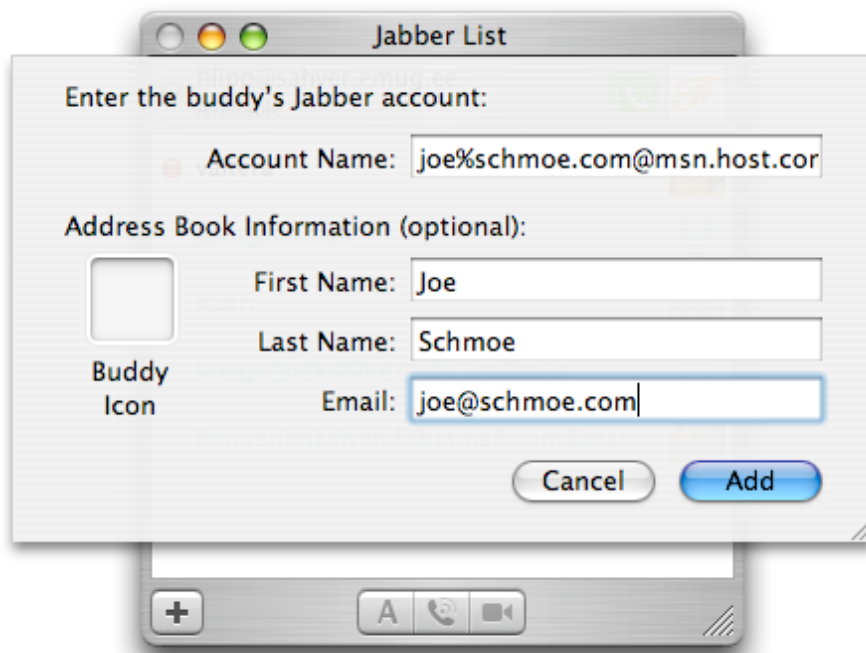


Figure 4: Adding an MSN contact in iChat AV

If your iChat server is running on *host.domain.com*, then make sure your users use *user@host.domain.com*, not *user@domain.com*, even if you've set just *domain.com* in *Host Domains* in the iChat server config. If a user connects as *user@domain.com*, they won't be able to add MSN contacts, they'll just be stuck at "Waiting for authorization" forever.

OK, once you've established that everything (messaging, avatars and file transfers) is working as expected, it's time to clean up after ourselves.

## Cleaning up

First let's uncomment the `<background />` directive in the transport configuration and re-set logging. Then we should put the transport someplace safe, like `/var/jabber/modules`:

```
$ sudo mv pymsnt-0.11.2 /var/jabber/modules/pymsnt
$ chown -R jabber:jabber /var/jabber/modules/pymsnt/
$ chmod -R 750 /var/jabber/modules/pymsnt/
```

The newly installed gateway software is a service that should start up with the server so let's create a nice [Launch Daemon](#) for it:

```
$ sudo nano /Library/LaunchDaemons/org.jabberstudio.msn-transport.plist
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
  <key>GroupName</key>
  <string>jabber</string>
  <key>UserName</key>
  <string>jabber</string>
  <key>Label</key>
  <string>org.jabberstudio.msn-transport</string>
  <key>ProgramArguments</key>
  <array>
    <string>/var/jabber/modules/pymsnt/PyMSNt.py</string>
  </array>
  <key>RunAtLoad</key>
  <true/>
  <key>WorkingDirectory</key>
  <string>/var/jabber/modules/pymsnt</string>
</dict>
</plist>
```

We can check if we got the XML right by doing:

```
$ plutil /Library/LaunchDaemons/org.jabberstudio.msn-transport.plist
```

OK?, let's load the daemon:

```
$ sudo launchctl load -w
/Library/LaunchDaemons/org.jabberstudio.msn-transport.plist
```

and start the gateway:

```
$ sudo launchctl start org.jabberstudio.msn-transport
```

You can see if the daemon was started by:

```
$ ps aux | grep PyMSNt
$ jabber 23043 0.0 0.8 37720 8696 ?? Ss 10:37PM 0:06.56 python
/var/jabber/modules/pymsnt/PyMSNt.py
```

Finally just connect and make sure everything's working, here's a shot of Messenger and iChat working together over MSN:

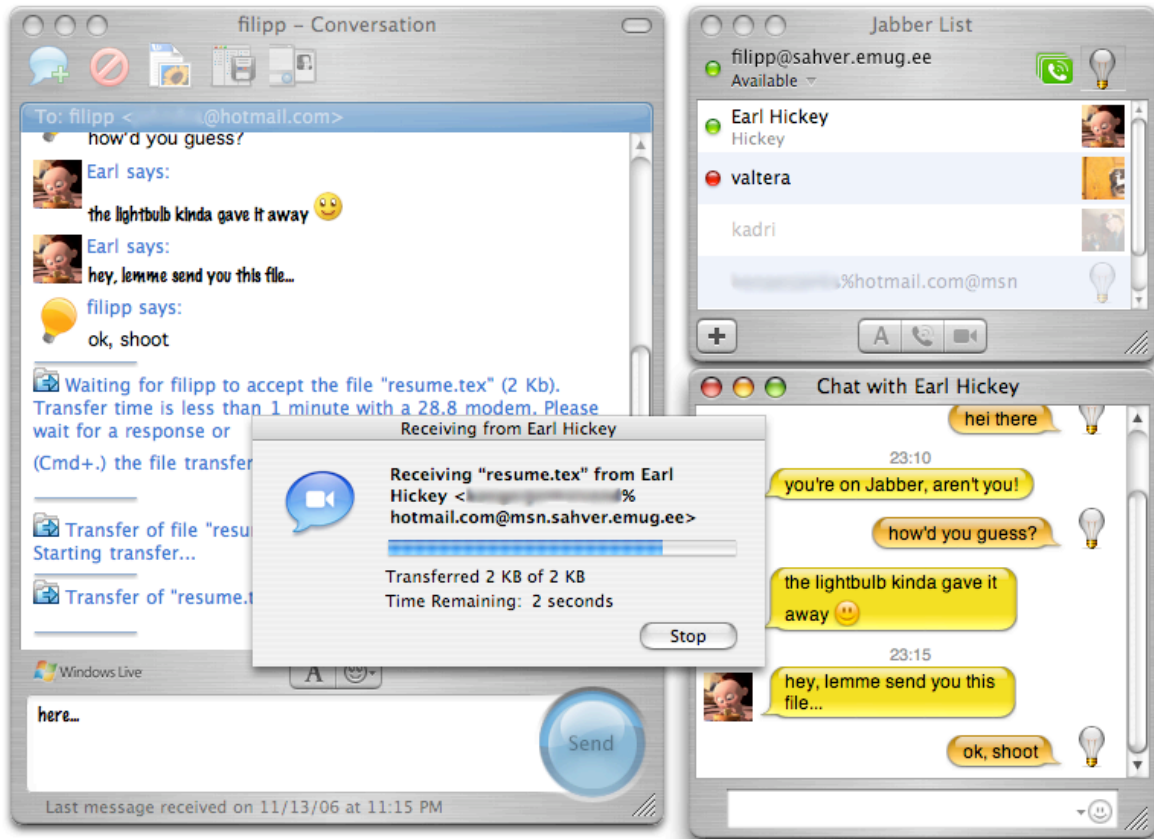


Figure 5: MSN Messenger and iChat chatting and exchanging files

## Conclusion

That's it! You should now have the necessary information to set up your own MSN gateway. I think the effortless file sharing with MSN and better control over MSN usage internally actually makes this worthwhile. Hopefully this was also a useful little insight into the iChat server and the inner workings of Jabber. Don't forget that there are many protocols out there that Jabber is capable of supporting (like Yahoo, AIM or ICQ) and you can adapt a lot of the information presented here in working with them. Even if you choose to use someone else's gateway service, you should now know what goes on behind the scenes.

If you think all of the previous was too messy or you're not particularly keen on Apple's IM server to begin with, then you might want to check out [Wildfire](#) which does all of that and a whole lot more.

Thanks for following and I hope You enjoyed it! All comments welcome, as always.