

There are several issues regarding using an XMPP protocol (<http://xmpp.org/extensions/xep-0060.html>) to send and receive messages among agents using a pubsub framework.

1. XMPP Server
2. Connect to Server
3. Create pubsub nodes
4. Subscribe to a pubsub node
5. Send messages to a pubsub node
6. Receive messages from a pubsub node

XMPP Server

XMPP (Extensible Messaging and Presence Protocol) is a protocol based on Extensible Markup Language ([XML](#)) and intended for instant messaging ([IM](#)) and online presence detection. It functions between or among servers, and facilitates near-real-time operation. The protocol may eventually allow Internet users to send instant messages to anyone else on the Internet, regardless of differences in operating systems and browsers. XMPP is sometimes called the [Jabber](#) protocol.

There are several XMPP Servers. OpenFire 3.6.0 is chosen for this project.

Connect to Server

Any user who can connect to a XMPP Server is called a client.

The way a simple client can be defined and connected to a server is as follows in XMPP4R (Ruby implementation of XMPP)

```
require "xmpp4r"
require "xmpp4r/pubsub"

include Jabber

jid = "user1@norbit.npc.nicta.com.au"
pass = "123"
client = Client.new(jid)
client.connect
client.auth(pass)

# 2 - Ping the Server (show our presence)
pres = Presence.new
client.send(pres)
```

Create a pubsub node

To create a pubsub node, an instance of a ServiceHelper class can be used. Here is a sample code.

```

# 3 - Create a new node and subscribe to it
pubsubjid="pubsub.norbit.npc.nicta.com.au"
service=PubSub::ServiceHelper.new(client,pubsubjid)
service.create_node("my_node",Jabber::PubSub::NodeConfig.new(nil,{
  "pubsub#title" => "#{node}",
  "pubsub#node_type" => "leaf",
  "pubsub#send_last_published_item" => "never",
  "pubsub#send_item_subscribe" => "0",
  "pubsub#publish_model" => "open"}))

```

In this example, it has been shown how different attributes of a node can be determined at the time of its creation.

Subscribe to a pubsub node

Any other client can subscribe to a pubsub node (if the node allows). Here is a simple example.

```

require "xmpp4r"
require "xmpp4r/pubsub"

include Jabber

jid = "user2@norbit.npc.nicta.com.au"
pass = "123"
client = Client.new(jid)
client.connect
client.auth(pass)

# 2 - Ping the Server (show our presence)
pres = Presence.new
client.send(pres)

# 3 - Subscribe to a node
pubsubjid="pubsub.norbit.npc.nicta.com.au"
service=PubSub::ServiceHelper.new(client,pubsubjid)
service.subscribe_to("my_node")

```

Send messages to a pubsub node

A client can publish a message into a pubsub node as follows.

```

item = Jabber::PubSub::Item.new
message=Jabber::Message.new(nil,"My Message")
item.add(message)
service.publish_item_to("my_node",item)

```

Receive messages from a pubsub node

To receive messages from a pubsub node, every client should have a callback function. It can be defined as follows:

```
service.add_event_callback { leventl
  puts event
    msg=event.first_element("items").first_element("item").first_element("message").first
    _element("body").text
  puts "Original Message is → #{msg}"
}
```

In the above example, the event is an XML message. The body of it can be extracted as shown.