# Neato SmartApps: EJabberd Integration

Raja Software

Last updated: 03/26/2014

|  |  |  |
| --- | --- | --- |
| Date | Version | Author/Comments |
| 03/25/2014 | 0.1 | Initial draft |
| 03/26/2014 | 0.2 | Added more detail, formatting changes |

# 

# Summary

This document summarizes the integration of the Neato SmartApps backend server with the XMPP server.

# Ejabberd Configuration

Ejabberd is one of the most popular open source XMPP server and it is used as XMPP server at the backend. Ejabberd has an apt-get installer available for Ubuntu and can be installed simply with

apt-get install ejabberd

Configuration file of the Ejabberd is located at

/etc/ejabberd/ejabberd.cfg

A detailed installation steps and required configuration steps are written in the Server Setup document.

To secure, Ejabberd, we have disabled registration from the remote clients and a new client can be created only from the same machine.

We are not running Ejabberd over a secure connection but it can be easily configured by setting following parameters in the ejabberd.cfg file.

{s2s\_use\_starttls, true}.

{s2s\_certfile, "/etc/ejabberd/ejabberd.pem"}.

{domain\_certfile, "[example.org](http://example.org/)", "/path/to/example\_org.pem"}.

# Ejabberd User Management

The current implementation has a limitation that the backend application should have the Ejabberd server running on the same instance. When robots and users are created, a shell\_exec calls is executed that creates Chat IDs for the users and robots. These Chat IDs are stored in the DB against the robots and users. If ejabberd server is down for some reason, an appropriate error message is thrown.

exec("sudo ejabberdctl register [CHATID@rajatogo.com](mailto:CHATID@rajatogo.com) CHATPWD");

For creating the Chat IDs, we are using current time in milliseconds as suffix with "robot\_" or "user\_". Note that the Chat passwords are same as the Chat IDs. Although we are storing the Chat IDs and passwords in our DB, we are not using these passwords anywhere.

Similarly when a robot is deleted, the chat ID is removed using Ejabberdctl's unregister method.

# Ejabberd Scalability

For scaling the Ejabberd, we would be doing following changes in the existing implementation:

* Ejabberd stores the data in the Mnesia DB but has easy way to configure it with other DB. For example, if we have to use MySQL instead of Mnesia, we can configure it by setting following parameters in the ejabberd.cfg file.

{odbc\_server, {mysql, "server", "database", "username", "password"}}.

* Ejabberd gives an easy way to create cluster of Ejabberd nodes. This way we would be able to bring in more nodes as and when required and would be able to scale this solution.
* RabbitMQ (producer/consumer platform) is used to decouple the Ejabberd and Apache server.

# XMPP Messages

In order to decouple the Apache and Ejabberd integration, we are not sending the XMPP message directly using the Ejabberd. We are using the RabbitMQ instead.

When an XMPP message needs to be sent (from the SetRobotProfileData3 API), it adds the messages to RabbitMQ queue from where a consumer picks up this message JSON and sends the actual message. An actual message looks like:

<?xml version="1.0" encoding="UTF-8"?>

<packet>

   <header>

      <version>1</version>

      <signature>0xcafebabe</signature>

   </header>

   <payload>

      <request>

         <command>5001</command>

         <requestId>59054</requestId>

         <timeStamp>1395731229</timeStamp>

         <retryCount>0</retryCount>

         <responseNeeded>false</responseNeeded>

         <distributionMode>2</distributionMode>

         <params>

            <robotId>N1</robotId>

            <causeAgentId />

         </params>

      </request>

   </payload>

</packet>

# Presence Status

As it is important to know if a Robot is online, mod\_eventful is configured with the Ejabberd to inform the application when a Robot comes online or goes offline. The callbacks for the mod\_eventful are configured in the ejabberd.cfg and the code snippet looks like:

{mod\_eventful,

        [

              {url,

                [

                {online\_hook, "<http://neatostaging.rajatogo.com/api/robotStatus/online>"},

                {offline\_hook, "<http://neatostaging.rajatogo.com/api/robotStatus/offline>"},

                {unset\_presence\_hook, undefined},

                {set\_presence\_hook, undefined},

                {message\_hook, undefined}

                ]

                },

These callback hooks take care of calling SetRobotProfileData3, which in turn results in an XMPP notification to all the users associated with Robot and the robot itself.

End.