Zoe Documentation

Release 0.1.0

GUL UC3M

August 11, 2014

Contents

CHAPTER 1

Introduction

1.1 What is Zoe?

Zoe is an assistant bot for the *Grupo de Usuarios de Linux UC3M* (GUL UC3M) non-profit organization. Its aim is to perform repetitive and time-consuming tasks in an asynchronous, uncoupled and lazy manner.

1.2 How does it work?

Zoe has two main components: the main *Zoe server*, and a group of *agents*, small programs performing a few, small and well defined tasks such as capturing statistics from the organization's systems, sending emails, generating reports and controlling alarms.

The *server* implements a *message bus* that the agents use to interchange messages asynchronously. It is also possible for Zoe to receive commands from the system administrators to execute tasks on demand.

1.3 License

The main Zoe server is released under the MIT license. However, the agents' license may vary depending on each agent's developer.

CHAPTER 2

Getting started

This section will help you getting started with the Zoe startup kit.

2.1 Requirements

- OS X or Linux (BSD should be fine too, although it has not been tested)
- Bash
- Java 1.7
- Python 3
- Perl 5

2.2 Directory structure

The Zoe startup kit has the following directory structure:

```
agents/         Installed Zoe agents
cmdproc/        Administrator commands in natural language
etc/            Configuration files
lib/            Libraries and agent dependencies by language
    bash/
    java/
    perl/
    python-dependencies/
    python/
logs/           Agent log files
mailproc/       Actions to execute when receiving mails
server/         Zoe server binaries
var/            Databases and temp files generated by agents
zoe.sh          Zoe script
```

2.3 Installation

Fetch the source for the Zoe startup kit and place it in the directory you want to run *Zoe* from, such as `/home/zoe`. Navigate to that directory and make the following scripts executable:

```
$ chmod +x zoe.sh

$ chmod +x etc/environment.sh
```

2.4 Configuration

Zoe's configuration files are found in the `etc/` directory. These are the base files you will find there:

```
environment.sh          Script to set environment variables
zoe-users.conf          Zoe users configuration
zoe.conf                Configuration of agent ports and topics
```

2.4.1 Environment

The `environment.sh` script sets several variables such as the *server host*, *twitter credentials*, *jabber information*, etc. It has the following content:

```
# Typical values
export ZOE_SERVER_HOST=localhost
export ZOE_SERVER_PORT=30000

# Get this from twitter
export zoe_twitter_consumer_key="..."
export zoe_twitter_consumer_secret="..."
export zoe_twitter_access_token="..."
export zoe_twitter_access_token_secret="..."

# Parameters for Google Talk
export zoe_jabber_host="talk.google.com"
export zoe_jabber_port="5222"
export zoe_jabber_user="..."
export zoe_jabber_password=""

# Parameters for a GMail account
export zoe_mail_smtp="smtp.gmail.com"
export zoe_mail_smtp_port="587"
export zoe_mail_pop3="pop.gmail.com"
export zoe_mail_pop3_port="995"
export zoe_mail_enable_dkim="false"
export zoe_mail_user="$zoe_jabber_user"
export zoe_mail_password="$zoe_jabber_password"
```

You should never give Zoe your own twitter/jabber credentials. Create a new account if you intend to use them.

- The information required to connect to a Twitter account can be obtained from the Twitter profile.
- By default, the Jabber and GMail accounts use the same credentials due to the Google Talk (Hangouts) service using Jabber. You can use other Jabber service and email provider if you wish so.

2.4.2 Users

The `zoe-users.conf` file follows this structure:

```
[subject admin]
name = Admin
twitter = your twitter name without @
preferred = twitter
jabber = your_jabber_id@wherever.com
mail = your_email@address.com
alias = god master

[group admins]
members = admin

[group broadcast]
members = admin
```

Each user is identified using the `[subject USER]` tag and has the following attributes:

- `name`: The user's name (e.g. Zoe).
- `twitter`: The user's twitter username (e.g. gul_zoe).
- `preferred`: The preferred way for Zoe to communicate with the user. Available values are `twitter`, `jabber`, `mail`.
- `jabber`: The user's jabber address. Note that if you are using *Google Hangouts*, the jabber addresses have the format `abcdefghijklmnopqrstuwxyz1@public.talk.google.com`. If you don't know this address, you may try speaking with *Zoe* through Jabber and getting the address from the logs.
- `mail`: The user's mail address.
- `alias`: The user's alias.

Some agents may use *groups* to send mails to several users at the same time or check for user permissions. Define a new group by using the scructure `[group GROUPNAME]` and adding its users to the `members` attribute.

2.5 First run

From a terminal navigate to the *Zoe* root directory (e.g. `/home/zoe`) and do the following:

- Load the *Zoe* configuration:

  ```
  $ . etc/environment.sh
  ```

- Launch *Zoe*:

  ```
  $ ./zoe.sh start
  ```

You will see that the output shows the agents that have been started. You can check the status of these agents by running:

```
$ ./zoe.sh status
```

If there are dead agents, you may want to check their log files (located in the `log/` directory). Most agent errors are due to misconfigurations.

That's it, now you have a functional *Zoe* instance running in your machine. You can stop the server by typing:

```
$ ./zoe.sh stop
```

Which will also stop all the agents running.

CHAPTER 3

Agents

In this section you will find information related to agents.

3.1 What is an agent?

Agents are small programs that *Zoe* has access to in order to perform different actions. Each agent is in charge of specific tasks and communicate between themselves through the server *message bus*. The server creates sockets for each agent either dynamically or by using the `etc/zoe.conf` file, which contains the port for each agent and a list of agents related to a specific topic.

3.1.1 Structure

Agents usually consist of 1-2 files, although this is completely up to the agent's developer. Most of the agents are written in *Python*, but libraries are available for writting agents in languages such as *Bash*, *Java*, *Python*, etc.

The main script is placed in the `agents/AGENT_NAME` directory. The agent can also include *natural language commands* for executing actions on demand without the need of using the command-line to manually send the message to the server. This way, an user can *speak* with *Zoe* through, for instance, Jabber. These natural language commands are placed in the `cmdproc/` directory.

3.2 Messages

3.2.1 Message structure

Agents interchange messages through the server *message bus*. These message strings have a *key-value* structure of the form:

```
key1=val1&key2=val2&key3=val3...
```

For instance, the `broadcast` agent can send a message to a specific user. The message string to do so would look like this:

```
dst=broadcast&tag=send&msg=Hello World!&to=john
```

Now, analyzing the message:

- `dst`: Agent that is supposed to receive this message.
- `tag`: Action to perform. There can be several `tag` entries in the message.

- `msg`: Message to send to the user.
- `to`: User to send the message to. This name is checked against the user list to see what John's address and preferred contact method are.

The message string will vary depending on each agent, but most of them will have common parts such as `dst` and `tag`.

3.2.2 Sending messages to the server

Apart from the agents interchanging messages, it is also possible to manually send a message to the server in order to have an agent execute some action. Taking the previous `broadcast` agent example, the message could be sent like so:

```
$ echo -n "dst=broadcast&tag=send&msg=Hello World!&to=john" | nc ZOE_HOST ZOE_PORT
```

Now, the `logs/broadcast.log` file would have an entry for the received message and the action executed (or any errors while doing so).

3.3 Installing new agents

There are two ways of installing a new agent into your *Zoe* instance.

3.3.1 Manual installation

Installing an agent manually requires that you download the code and place it in the *Zoe* installation according to the agent developer's instructions. Usually, it will just be a matter of copying and pasting files and maybe modifying some configurations. When the server is started the next time, the new agent should be up and running.

3.3.2 Using the agent manager

The Zoe startup kit should include an agent named *zam*. This is the Zoe Agent Manager, which aims to facilitate the installation and management of agents in your *Zoe* instance. If the Zoe Agent Manager is not installed, you can easily install it manually by downloading the source and following the provided instructions.

The manager uses git repositories to fetch and install agents. For instance, the dummy agent has the following git source URL:

```
https://github.com/rmed/dummy_agent.git
```

If you were to tell the manager to install this agent, you could send this message to the server:

```
$ echo -n "dst=zam&tag=install&name=dummy&source=https://github.com/rmed/dummy_agent.git" | nc ZOE_H
```

Or, if you prefer using natural language and Jabber:

```
Zoe, install agent "dummy" from "https://github.com/rmed/dummy_agent.git"
```

For more information on the manager and all the commands it has, please check the zam wiki.