L.I.S.A Documentation

Release 0.1

Julien Syx

May 07, 2017

Contents

CHAPTER 1

Introduction to L.I.S.A

The 30 second summary

L.I.S.A is:

- an intelligent system to help you in your daily life
- a framework to connect objects between them and expose a HTTP API

It was developed in order to have a "J.A.R.V.I.S" like interface (Iron Man).

The name come from "Weird Science" TV Show. Two nerds try to create the perfect woman on their computer and one night she come to life. In France, this TV show is named "Code LISA".

That's why this project has this name.

Building on proven technology

L.I.S.A takes advantage of a number of technologies and techniques. The networking layer is built with the excellent Twisted networking library.

L.I.S.A uses public keys for authentication to identify clients allowed to connect.

The Web interface is built on top of Django which is a web framework for python.

L.I.S.A Community

Join L.I.S.A !

There are many ways to participate in and communicate with the L.I.S.A community.

L.I.S.A has a chat and a Question&Answer website.

Q&A

The Q&A Website is the best place to ask questions about L.I.S.A.

Chat

The chat is hosted on Gitter. You will only need a github account.

Follow on Github

The L.I.S.A code is developed via Github. Follow L.I.S.A for constant updates on what is happening in L.I.S.A development:

- L.I.S.A Server.

You can follow the roadmap on Waffle :

- L.I.S.A Backlog.

And check the build status on Travis :

- L.I.S.A Build.

Other community links

- L.I.S.A Project.
- Google+

Hack the Source

If you want to get involved with the development of source code or the documentation efforts, please review the *hacking section*!

CHAPTER 2

About L.I.S.A and his author

CHAPTER 3

Installation

See also:

Contributing to L.I.S.A

Quick Install

```
# Installing Debian dependencies
sudo apt-get install mongodb python-setuptools libxslt1-dev libxslt1.1 libxml2-dev␣
↪libffi-dev build-essential python-dev libssl-dev python-openssl libyaml-dev
# Installing pip
sudo easy_install pip
# Installing tools to create a python virtual environment
sudo pip install virtualenv virtualenvwrapper
# Create the lisa user (you can choose another username)
sudo useradd -s /bin/bash -m alivelisa
# Login with this user
sudo su - alivelisa
# Set environment variable to use correctly the new virtual environment
export WORKON_HOME=$HOME/.virtualenvs
echo "export WORKON_HOME=$HOME/.virtualenvs" >> ~/.bashrc
export PROJECT_HOME=$HOME/
echo "export PROJECT_HOME=$HOME/" >> ~/.bashrc
source /usr/local/bin/virtualenvwrapper.sh
echo "source /usr/local/bin/virtualenvwrapper.sh" >> ~/.bashrc
# Create the virtual environment named "lisa"
mkproject lisa
# This command let you enter your environment and load all libraries. Now Lisa is␣
↪usable
workon lisa
# Install twisted then the LISA server
pip install twisted
pip install lisa-server
```

```
# Create your user
lisa-cli createsuperuser
```

Platform-specific Installation Instructions

These guides go into detail how to install L.I.S.A on a given platform.

Installation on Arch Linux

Installation on Debian Linux

Installation on Fedora Linux

Installation on FreeBSD

Installation on Gentoo Linux

Installation on Mac OSX

Installation on RedHat Linux

Installation on Solaris

Installation on NAS Synology

Installation on Windows

Installation on Suse Linux

Dependencies

L.I.S.A should run on any Unix-like platform so long as the dependencies are met.

```
mongoengine >= 0.8.7
Django >= 1.6.2
Sphinx >= 1.2.2
Twisted >= 13.2.0
autobahn >= 0.8.7
pymongo >= 2.7
requests >= 2.2.1
django-tastypie >= 0.11.0
django-tastypie-mongoengine >= 0.4.5
django-tastypie-swagger >= 0.1.2
pytz >= 2014.2
pyOpenSSL == 0.13
lisa-plugin-ChatterBot
PyWit >= 0.2.0
```

Upgrading L.I.S.A

To upgrade L.I.S.A manually, you can upgrade the python package (in the virtualenv):

```
pip install --upgrade lisa-server
```

CHAPTER 4

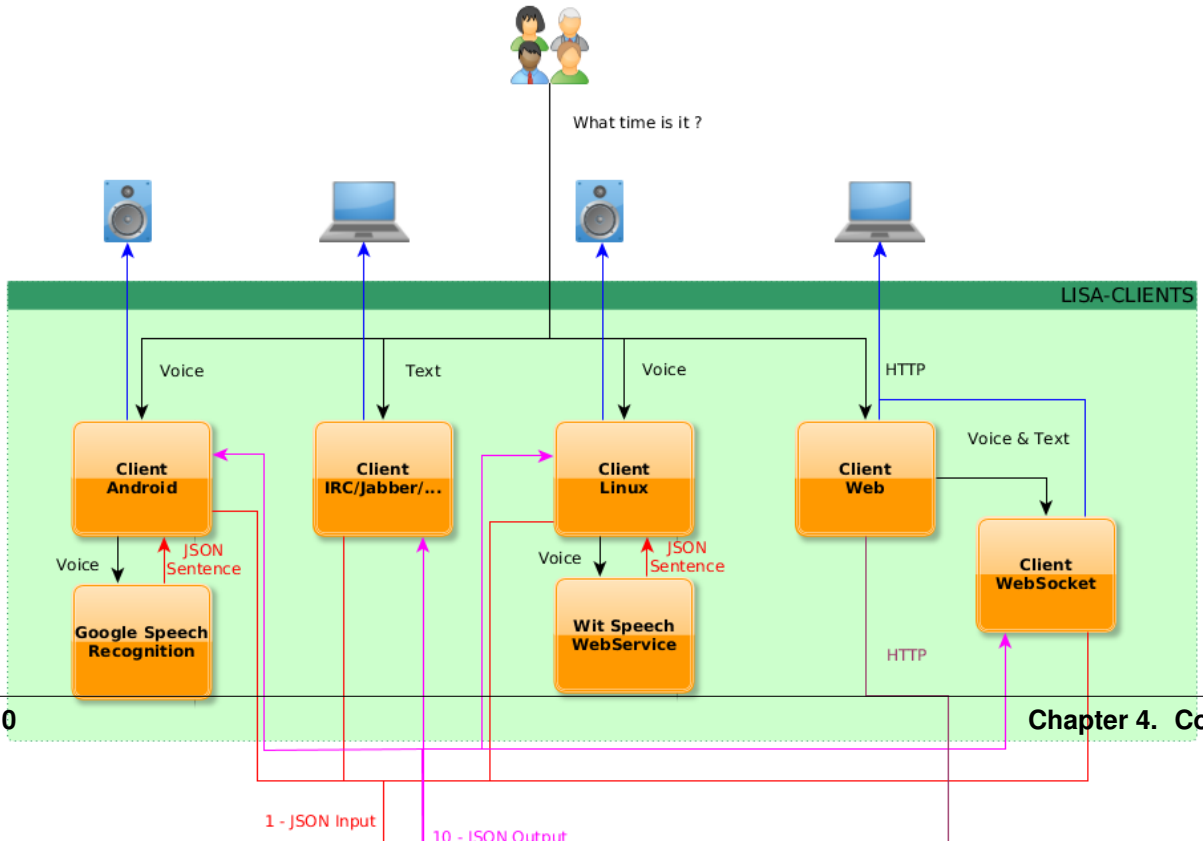
Core

Engine

Plugin Manager

Web Server

Workflow



CHAPTER 5

Plugins

Create a plugin for L.I.S.A

Creating a plugin is very easy, thanks to the cli tool.

To create your plugin, type (in the virtualenv):

```
lisa-cli plugin PLUGINNAME --create
```

Then, answer the questions, and you will find the plugin in your virtualenv.

```
/home/alivelisa/.virtualenvs/lisa/local/lib/python2.7/site-packages/lisa/plugins/
↪PLUGINNAME
```

By creating a new plugin you don't install it automatically. So write the code you want in the plugin, then install it in order to use the plugin. Don't forget to install it in dev mode (by using –dev)

See also:

Distribute your Plugin

Distribute your Plugin

Python package

One you have created your plugin, you may want to distribute it.

LISA use python package index to manage plugin dependencies and install/uninstall. With this system, plugins can be used as a library in another plugin as it's a python package.

The *create option* does not create the structure to deploy it.

To create the setup structure, you can check how plugins are actually done : ChatterBot Plugin

Register it on the Plugin Store

Not released yet. Contact me to add it on the list of plugins

Plugin Management

The plugin management is usefull to quickly manage plugins from Command Line Interface. It uses the same function than the web interface so it has the same behavior.

To list the functionnalities

```
$ lisa-cli plugin --help
Usage: /home/alivelisa/.virtualenvs/lisa/bin/lisa-cli plugin [options] <plugin_name>

Manage the plugins

Options:
  --list                List all plugins and show their status
  --create              Create a template plugin with a given name
  --enable              Enable a plugin
  --disable             Disable a plugin
  --install             Install a plugin
  --dev                 Dev mode
  --uninstall           Uninstall a plugin
  --upgrade             Upgrade a plugin
  -h, --help            show this help message and exit
```

list This option does not need to be called with a plugin name.

```
$ lisa-cli plugin --list
ChatterBot => [Installed] [Enabled]
BBox => [Installed] [Enabled]
ProgrammeTV => [Installed] [Enabled]
Cinema => [Not installed] [Not enabled]
Meteo => [Not installed] [Not enabled]
SNCF => [Not installed] [Not enabled]
Shopping => [Not installed] [Not enabled]
Wifiledlamps => [Not installed] [Not enabled]
```

create This option will create a plugin in your plugins directory. It will ask some questions to auto-configure the plugin.

```
$ lisa-cli plugin --create PLUGINNAME
What is your full name ? : Lisa
What is your email ? : lisa@lisa-project.net
[OK] Plugin created
```

enable / disable This option will enable or disable a plugin. That means the plugin the plugin still exists in database and on filesystem but will not be loaded by the plugin manager.

```
$ lisa-cli plugin --enable PLUGINNAME
[OK] Plugin enabled
```

```
$ lisa-cli plugin --disable PLUGINNAME
[OK] Plugin disabled
```

dev This option allow you to specify that you are writing a plugin. It will not download it on Python Package Index but use the local plugin instead. It will not do anything on the filesystem but only on database. It is mainly used for install and uninstall function. When used with install, it will update all fields in database according your local plugin files. If you use uninstall with the dev mode, it will not delete the json file, only records in the database.

install This option will install a plugin. By default it download the package from Python Package Index then read the json file and install all components (rules, crons, intents, plugin configuration) in the database.

```
$ lisa-cli plugin --install PLUGINNAME
[OK] Plugin installed
```

uninstall This option will uninstall a plugin. By default it will remove the package and all the files related to the plugin and remove entries related to the plugin in database.

```
$ lisa-cli plugin --uninstall PLUGINNAME
[OK] Plugin uninstalled
```

upgrade This option is not implemented yet.

```
$ lisa-cli plugin --upgrade PLUGINNAME
[OK] Plugin upgraded
```

Structure of a Plugin

CHAPTER 6

How Wit works ?

CHAPTER 7

Tutorials

CHAPTER 8

Troubleshooting

The intent of the troubleshooting section is to introduce solutions to a number of common issues encountered by users and the tools that are available to aid debug the code.

Server Troubleshooting

If your server is having issues such as server not returning data, slow execution times, or a variety of other issues the *Server troubleshooting page* contains details on troubleshooting the most common issues encountered.

Running in the Foreground

A great deal of information is available via the debug logging system, if you are having issues is to run the server in the foreground to display logs in the console:

```
twistd -n lisa-server
```

What Ports do the Server and Clients Need Open?

No ports need to be opened up on clients. For the server, TCP ports 10042, 10043 and 8000 (or 80) need to be open. If you can't connect on you server, and can't access the web interface, it could be a firewall problem.

You can check port connectivity with the nc command:

```
nc -v -z lisa.server 10042
nc -v -z lisa.server 8000
```

CHAPTER 9

Developing

Contributing

There is a great need for contributions to L.I.S.A and patches are welcome! The goal here is to make contributions clear, make sure there is a trail for where the code has come from, and most importantly, to give credit where credit is due!

There are a number of ways to contribute to L.I.S.A development.

Sending a GitHub pull request

This is the preferred method for contributions. Simply create a GitHub fork, commit changes to the fork, and then open up a pull request.

The following is an example (from Open Comparison Contributing Docs) of an efficient workflow for forking, cloning, branching, committing, and sending a pull request for a GitHub repository.

First, make a local clone of your GitHub fork of the L.I.S.A GitHub repo and make edits and changes locally.

Then, create a new branch on your clone by entering the following commands:

```
git checkout -b fixed-broken-thing

Switched to a new branch 'fixed-broken-thing'
```

Choose a name for your branch that describes its purpose.

Now commit your changes to this new branch with the following command:

```
git commit -am 'description of my fixes for the broken thing'
```

Note: Using `git commit -am`, followed by a quoted string, both stages and commits all modified files in a single command. Depending on the nature of your changes, you may wish to stage and commit them separately. Also, note

that if you wish to add newly-tracked files as part of your commit, they will not be caught using `git commit -am` and will need to be added using `git add` before committing.

Push your locally-committed changes back up to GitHub:

```
git push --set-upstream origin fixed-broken-thing
```

Now go look at your fork of the L.I.S.A repo on the GitHub website. The new branch will now be listed under the "Source" tab where it says "Switch Branches". Select the new branch from this list, and then click the "Pull request" button.

Put in a descriptive comment, and include links to any project issues related to the pull request.

The repo managers will be notified of your pull request and it will be reviewed. If a reviewer asks for changes, just make the changes locally in the same local feature branch, push them to GitHub, then add a comment to the discussion section of the pull request.

Note: Travis

Whenever you make a pull request against the main L.I.S.A repository your changes will be tested. On average these tests take few minutes to run and once they are complete a PASS/FAIL message will be added to your pull request. This message contains a link to https://www.travis-ci.org where you can review the test results. This message will also generate an email which will be sent to the email address associated with your GitHub account informing you of these results. It should be noted that a test failure does not necessarily mean there is an issue in the associated pull request as the entire development branch is tested.

Note: Minor releases

Minor releases normally contain bug fixes.

When submitting a pull-request which should be considered for a minor release, please note in the comments that it should be reviewed for inclusion.

Pull requests that are accepted to L.I.S.A but not merged into a minor release will always be available in the next major release.

Keeping L.I.S.A Forks in Sync

L.I.S.A is advancing quickly. It is therefore critical to pull upstream changes from master into forks on a regular basis. Nothing is worse than putting in a days of hard work into a pull request only to have it rejected because it has diverged too far from master.

To pull in upstream changes:

```
# For ssh github
git remote add upstream git@github.com:Seraf/LISA.git
git fetch upstream

# For https github
git remote add upstream https://github.com/Seraf/LISA.git
git fetch upstream
```

To check the log to be sure that you actually want the changes, run the following before merging:

```
git log upstream/develop
```

Then to accept the changes and merge into the current branch:

```
git merge upstream/develop
```

For more info, see GitHub Fork a Repo Guide or Open Comparison Contributing Docs

Sign off your pull request

When making pull requests, please sign-off the pull request in a pull request comment. You can simply make a comment something like:

```
Signed-off-by: John Doe <john.doe@hisdomain.com>
```

By signing-off you indicate that you are accepting the Developer Certificate Of Origin. For now, we are using them same DCO as Linux kernel developers are using.

Hacking L.I.S.A

Running The Tests

As L.I.S.A is written with Twisted it uses the integrated test system named "trial". To launch it :

```
trial lisa/server
```

Writing Tests

Git Policy

The L.I.S.A team follows a git policy to maintain stability and consistency with the repository.

The git policy has been developed to encourage contributions and make contributing to L.I.S.A as easy as possible. Code contributors to L.I.S.A projects DO NOT NEED TO READ THIS DOCUMENT, because all contributions come into L.I.S.A via a single gateway to make it as easy as possible for contributors to give us code.

The primary rule of git management in L.I.S.A is to make life easy on contributors and developers to send in code. Simplicity is always a goal!

Conventions

CHAPTER 10

Release notes

L.I.S.A 0.2 Release Notes

release 2014-09-07

0.2 has arrived! This release come with new features !

Major Features

- New interface written in AngularJS
- Everything use API
- Upgrade plugins and core from Interface
- Use sockjs for Websocket

Compatibility

You will have to delete your user from Mongo Database.

```
-seraf@tagada-laptop ~/Sandbox/LISA ‹lisa› ‹master*›
- mongo

MongoDB shell version: 2.4.9
connecting to: test
> use lisa
switched to db lisa
> db.user.remove()

lisa-cli createsuperuser
```

Archive

CHAPTER 11

How to contibute to documentation ?

CHAPTER 12

Frequently Asked Questions

FAQ

Is LISA open-source ?

Yes. LISA is 100% open-source, including all APIs.

And in the future, even if I create an enterprise with LISA, it will stay opensource and free.

Can I fork it ?

Yes you can fork it if you want. Everything will be opensourced, even the "appstore". But of course, I encourage everyone to contribute instead of spreading smart people.

Is LISA full local mode or does it need an Internet connection ?

LISA is actually needing an Internet connection. I work on it to have something fully working even if your connection is broken.

LISA need internet at least for his client (because speech recognition is online) and to use Wit Engine on the server side. Wit is also working on a local server for Wit. Once it will be available, it will be backported in LISA as soon as possible.

Which Speech Engine do you use ?

Even if it is one of the most important part of the project, I don't want to focus on it. The actual Linux Client use pocketsphinx to detect a keyword (the name of the bot). It's a trigger to launch the speech recognition by voice. LISA will listen and stream each chunk to Wit Speech API. It's a online webservice and they use many speech engine in backend to do the speech recognition. They plan to create free accoustic models and use (in the future) only opensource speech engine. They are partner of CMUSphinx.

Not convinced ? Don't want to see your voice go outside your home ?

Well, LISA was built to let the user choose his speech engine. You can create an irc bot client, a hangout bot, a jabber bot, a voice client which use Dragon Natural Speech, Windows Grammar, Android Speech Engine or whatever. The choice is yours, just send a sentence to LISA.

Don't forget that smaller is the device where is hosted the client, less will be the computing capacity to do speech recognition in local mode. But don't be worry, I want for myself something in local and working without internet connection.

Why LISA use mongodb as database ?

I am a DevOps, then when I code, I always think about how will it work in production. Is it scalable ? Mongodb let LISA to scale and support loss of a server. The database can be distribued accross your clients for example. The NoSQL technology allow developers to build plugins easily.

As plugins and configuration are stored in the mongodb database, you can scale the LISA server with a load balancer.

The only problem is a lack of support for ARM. But there's a ticket opened on their JIRA issue tracker.

Can I install LISA on a Raspberry Pi ?

Quick answer : Yes.

Long answer : RPI is an ARM architecture and actually Mongodb is unsupported officially. So it isn't packaged for raspbian for example. But you should be able to compile it (take ~7hours).

There's a ticket already opened on their JIRA for ARM support, and they are working on it.

By now, I suggest to run mongodb (and the LISA server) in a virtual machine or a dedicated x86 server.

What ports should I open on my firewall ?

LISA use by default the port 10042 to communicate between client and server, and port 8000 for the webserver.

Can I do Home Automation with this project ?

Short answer : No. LISA allow you to create a bridge between your connected objects and your home automation box. LISA can also handle some "intelligency" to execute orders on your home automation box, but it won't replace an automation box.

Why LISA is written in Python ?

Firstly, I wanted to learn this language since a long. The code may be not perfect so don't hesit to send a patch. I love Python because there's a lot of API written in Python language and as LISA is a built to connect API between them it was logic to use a language easy to learn and easy to use.