
Sevabot - Skype robot Documentation

Release 1.0

Mikko Ohtamaa, Pete Sevander

March 21, 2016

1	Installing and running on Ubuntu	3
2	Installing and running on OSX	13
3	Installing and running using Vagrant	17
4	Chat commands	19
5	Sending Skype messages via webhooks	23
6	Troubleshooting	33
7	Community and development	37
8	Trademark notice	39



Sevabot is a generic purpose hack-it-together Skype bot

- Has extensible command system based on UNIX scripts
- Send chat message notifications from any system using HTTP requests
- Built-in support for Github commit notifications and other popular services

It is based on [Skype4Py framework](#)

The bot is written in Python 2.7.x programming language, but can be integrated with any programming languages over UNIX command piping and HTTP interface.

The underlying Skype4Py API is free - you do not need to enlist and pay Skype development program fee.

Contents:

Installing and running on Ubuntu

- *Introduction*
- *Installing Skype and xvfb*
- *Setting up Skype and remote VNC*
- *Installing Sevabot*
- *Running sevabot*
- *Test it*
- *Testing HTTP interface*
- *Running sevabot as service*
- *Setting avatar image*
- *Installing on Ubuntu desktop*

1.1 Introduction

These instructions are for setting up a headless (no monitor attached) Sevabot running in Skype on Ubuntu Server. The instructions have been tested on Ubuntu Version **12.04.1** unless mentioned otherwise.

Note: For desktop installation instructions see below.

1.2 Installing Skype and xvfb

Install Ubuntu dependencies needed to run headless Skype.

SSH into your server as a root or do `sudo -i`.

Then install necessary software:

```
apt-get update
apt-get install -y xvfb fluxbox x11vnc dbus libasound2 libqt4-dbus libqt4-network libqt4-core4 libqtgui4
wget http://www.skype.com/go/getskype-linux-beta-ubuntu-64 -O skype-linux-beta.deb
# if there are other unresolved dependencies install missing packages using apt-get install and then
dpkg -i skype-linux-beta.deb
```

More packages and Python modules needed to:

```
apt-get install -y python-gobject-2
apt-get install -y curl git
```

1.3 Setting up Skype and remote VNC

Now we will create the UNIX user `skype` running Sevabot and Skype the client application.

Note: In this phase of installation you will need a VNC remote desktop viewer software on your local computer. On Linux you have XVNCViewer, on OSX you have Chicken of VNC and on Windows you have TinyVNC.

Under `sudo -i`:

```
# Create a random password
openssl rand -base64 32 # Copy this output, write down and use in the input of the following command
adduser skype # We must run Skype under non-root user
```

Exit from the current (root) terminal session.

Login to your server:

```
ssh skype@yourserver.example.com
```

Get Sevabot:

```
git clone git://github.com/opensourcehacker/sevabot.git
```

Note: If you want to live dangerously you can use `git dev` branch where all the development happen. You can switch to this branch with “`git checkout dev`” command in the `sevabot` folder.

Start `xvfb`, `fluxbox` and `Skype`:

```
# This will output some Xvfb warnings to the terminal for a while
SERVICES="xvfb fluxbox skype" ~/sevabot/scripts/start-server.sh start
```

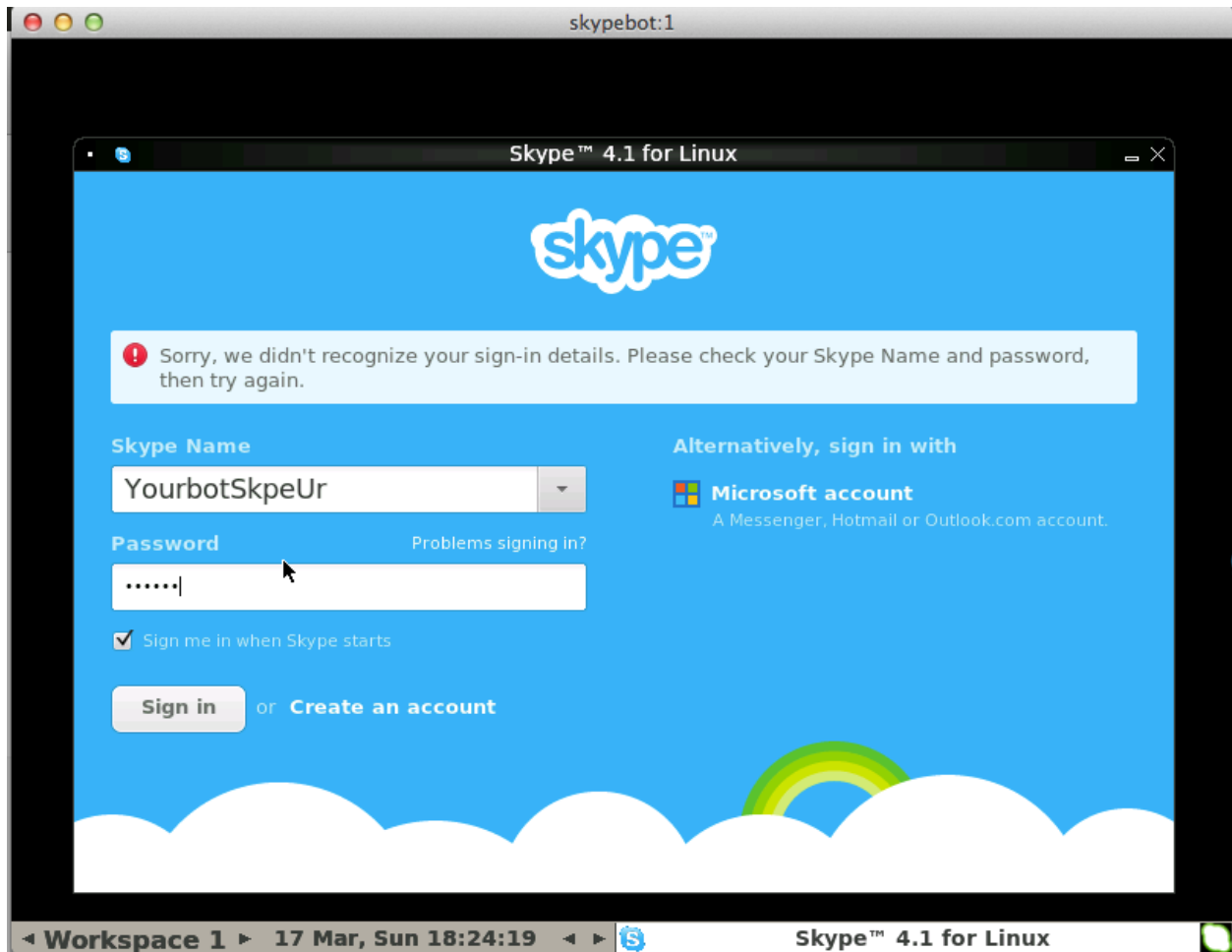
Start VNC server:

```
# This will ask you for the password of VNC remote desktop session.
# Give a password and let it write the password file.
# Delete file ~/.x11vnc/password to reset the password
~/sevabot/scripts/start-vnc.sh start
```

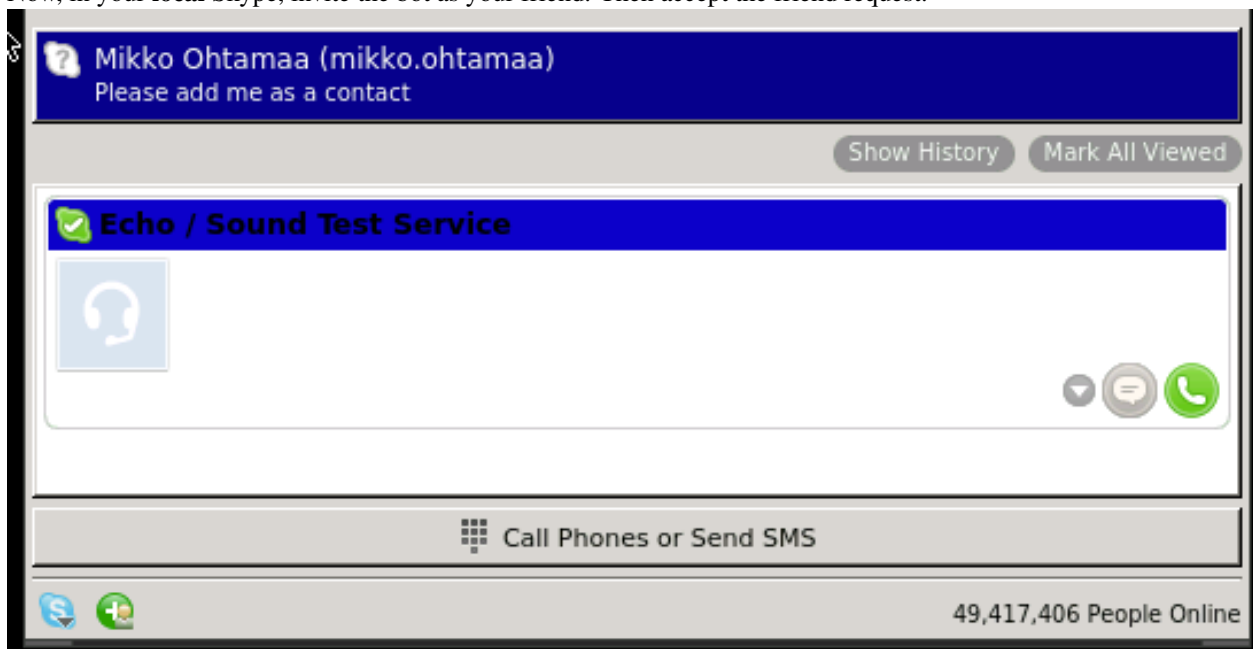
On your **local computer** start the VNC viewing software and connect the server:

```
vncviewer yourserver.example.com # Password as you give it above
```

You see the remote desktop. Login to Skype for the first time. Make Skype to save your username and password. Create Skype account in this point if you don't have one for sevabot.



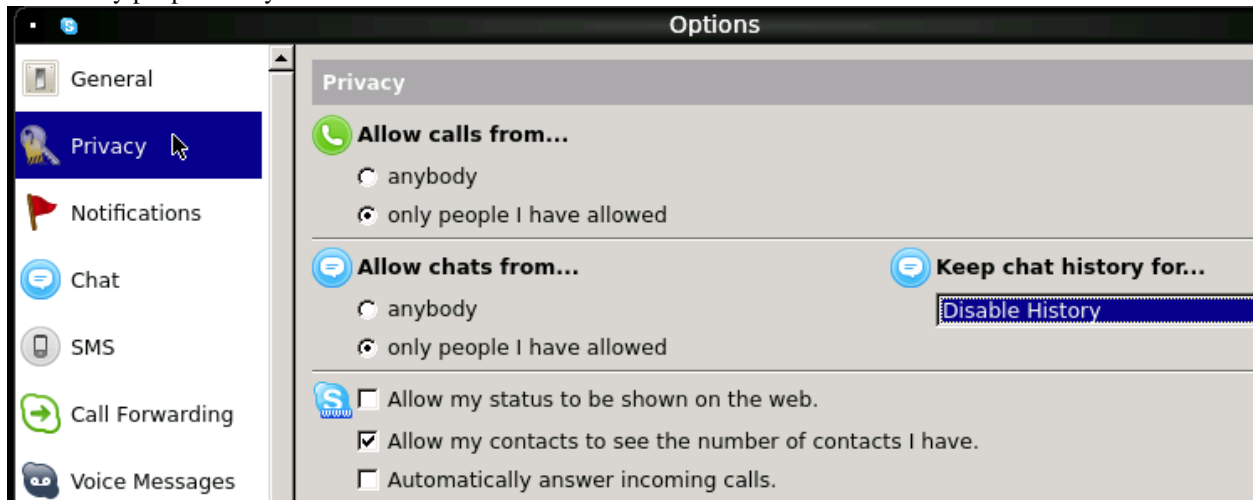
Now, in your **local** Skype, invite the bot as your friend. Then accept the friend request.



Note: It is important to add one Skype buddy for your Sevabot instance in this point, so don't forget to do this step.

Nowm, in Sevabot go to Skype's settings and set the following

- No chat history
- Only people on my list can write me
- Only people on my list can call me



1.4 Installing Sevabot

When Skype is up and running on your server, you can attach Sevabot into it.

Sevabot is deployed as [Python virtualenv installation](#).

Login to your server as skype user over SSH:

```
ssh skype@yourserver.example.com
```

Deploy sevabot, as checked out from Github earlier, using [Python virtualenv](#):

```
cd sevabot
curl -L -o virtualenv.py https://raw.githubusercontent.com/pypa/virtualenv/master/virtualenv.py
python virtualenv.py venv
. venv/bin/activate
python setup.py develop
```

This will

- Pull all Python package dependencies from [pypi.python.org](#) package service
- Create Sevabot launch scripts under `~/sevabot/venv/bin/`

Set password and customize other Sevabot settings by creating and editing `settings.py`:

```
# Create a copy of settings.py
cd ~/sevabot
cp settings.py.example settings.py
nano settings.py
```

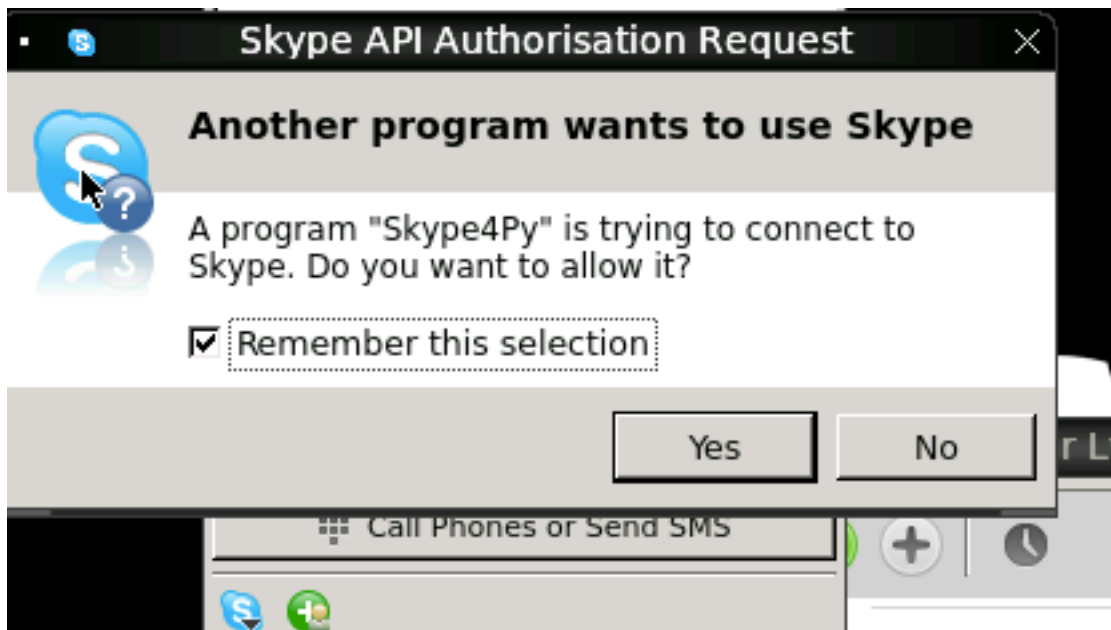
In `settings.py` set

- `SHARED_SECRET`: web interface password
- `HTTP_HOST`: Public IP address you want Sevabot's web interface listen to (on Ubuntu you can figure this out using `"ipconfig` command)

We need one more thing and that's accepting Skype dialog for Sevabot control in VNC session. Make sure Xvfb, Fluxbox, Skype and VNC is running as instructed above. Do:

```
# Start Sevabot and make initial connect attempt to Skype
SERVICES=sevabot ~/sevabot/scripts/start-server.sh start
```

Authorize the connection and tick *Remember* in VNC session



1.5 Running sevabot

To start the Sevabot do:

```
# Following will restart Xvnx, Fluxbox, Skype and Sevabot
~/sevabot/scripts/start-server.sh restart
```

The last line you see should be something like:

```
2013-03-17 18:45:16,270 - werkzeug - INFO - * Running on http://123.123.123.123:5000/
```

Note: Make sure your IP address is right in above

From the log files see that Sevabot starts up:

```
tail -f ~/sevabot/logs/sevabot.log
```

It should end up reading like this:

```
Started Sevabot web server process
```

1.6 Test it

Start chatting with your Sevabot instance with your *local* Skype.

In Skype chat, type:

```
!ping
```

Sevabot should respond to this message with Skype message:

```
pong
```

Note: Sometimes Skype starts up slowly on the server and the initial messages are eaten by something. If you don't get instant reply, wait one minute and type !ping again.

1.7 Testing HTTP interface

Sevabot server interface is listening to port 5000. This interface offers

- Chat list (you need to know group chat id before you can send message into it)
- [Webhooks](#) for integrating external services

Just access the Sevabot server by going with your web browser to:

```
http://yourserver.example.com:5000
```

Sevabot

A generic purpose hack together Skype bot at your service.

Status

Sevabot is running

Documentation

Please visit [Sevabot](#) project page for more information.

Admin

Chat list

1.8 Running sevabot as service

Sevabot and all related services can be controller with `scripts/start-server.sh` helper script. Services include

- Xvfb
- Fluxbox
- Skype
- Sevabot itself

Example:

```
scripts/start-server.sh stop
...
scripts/start-server.sh start
...
scripts/start-server.sh status
```

```
Xvfb is running
fluxbox is running
skype is running
Sevabot running
OVERALL STATUS: OK
```

To run sevabot from the server from reboot or do a full bot restart there is an example script [reboot-seva.sh](#) provided. It also does optionally manual SSH key authorization so that the bot can execute remote commands over SSH.

To make your Sevabot bullet-proof add [a cron job to check](#) that Sevabot is running correctly and reboot if necessary.

1.9 Setting avatar image

Sevabot has a cute logo which you want to set as Sevabot's Skype avatar image.

Here are short instructions.

Login as your sevabot user, tunnel VNC:

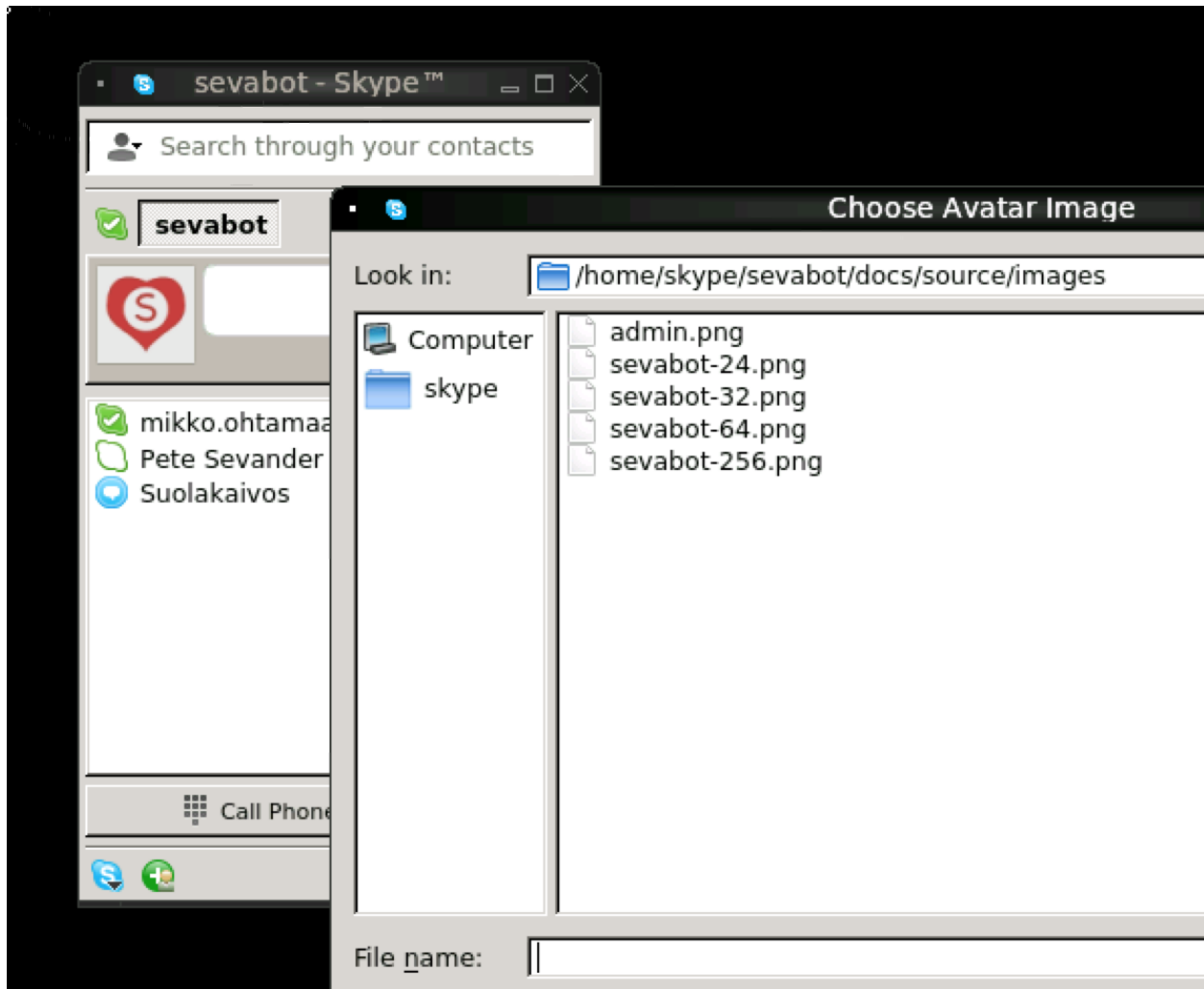
```
ssh -L 5900:localhost:5900 skype@example.com
```

Start VNC:

```
sevabot/scripts/start-vnc.sh start
```

On your local VNC client, connect to `localhost:5900`.

Set the avatar image through Skype UI.



1.10 Installing on Ubuntu desktop

You don't need Xvfb, VNC or fluxbox. These instructions were written for Ubuntu 12.04 64-bit.

Note: These instructions were written for running 32-bit Skype client application in 64-bit Ubuntu. Since writing the instructions the situation have changed and Skype has 64-bit application too. If you have insight of how to install these packages correctly please open an issue on Github and submit an updated recipe.

Install requirements and Skype:

```
sudo -i

apt-get install xvfb fluxbox x11vnc dbus libasound2 libqt4-dbus libqt4-network libqtcore4 libqtgui4

apt-get install python-gobject-2 curl git

wget http://www.skype.com/go/getskype-linux-beta-ubuntu-64 -O skype-linux-beta.deb
# if there are other unresolved dependencies install missing packages using apt-get install and then
dpkg -i skype-linux-beta.deb
```

```
exit
```

Start Skype normally, register a new user or you can also use your own Skype account for testing..

Install Sevabot:

```
git clone git://github.com/opensourcehacker/sevabot.git
cd sevabot
curl -L -o virtualenv.py https://raw.githubusercontent.com/pypa/virtualenv/master/virtualenv.py
python virtualenv.py venv
. venv/bin/activate
python setup.py develop
```

Customize Sevabot settings:

```
cp settings.py.example settings.py
```

Use your text editor to open `settings.py` and set your own password there.

Start sevabot:

```
. venv/bin/activate
sevabot
```

You should now see in your terminal:

```
Skype API connection established
getChats()
* Running on http://localhost:5000/
```

Now enter with your browser to: <http://localhost:5000/>.

Installing and running on OSX

- *Introduction*
- *Installing Skype*
- *Installing sevabot*
- *Set password and other settings*
- *Running sevabot*
- *Test it*
- *Testing HTTP interface*

2.1 Introduction

These instructions are for setting up a Sevabot to run on OSX desktop.

These instructions are mostly useful for Sevabot development and testing and not for actual production deployments.

2.2 Installing Skype

Install Skype for OSX normally. Create your Skype user.

2.3 Installing sevabot

Sevabot is deployed as [Python virtualenv](#) installation.

Install sevabot using [virtualenv](#):

```
git clone git://github.com/opensourcehacker/sevabot.git
cd sevabot
curl -L -o virtualenv.py https://raw.githubusercontent.com/pypa/virtualenv/master/virtualenv.py
arch -i386 python virtualenv.py venv
source venv/bin/activate
arch -i386 python setup.py develop
```

This will

- Pull all Python package dependencies from *pypi.python.org*

- Create a scripts under `venv/bin/` to run Sevabot

Note: If you want to live dangerously you can use git dev branch where all the development happen.

2.4 Set password and other settings

Customize Sevabot settings:

```
# Create a copy of settings.py
cd ~/sevabot
cp settings.py.example settings.py
```

Setup your Skype admin username and HTTP interface password by editing `settings.py`.

2.5 Running sevabot

Type:

```
arch -i386 sevabot
```

When you launch it for the first time you need to accept the confirmation dialog in the desktop environment (over VNC on the server).

or which ever display you're running your skype on your server.

Note: There might be a lot of logging and stdout output when the bot starts and scans all the chats of running Skype instance.

Eventually you see in the console:

```
Running on http://127.0.0.1:5000/
```

2.6 Test it

In Skype chat, type:

```
!ping
```

Sevabot should respond to this message with Skype message:

```
pong
```

2.7 Testing HTTP interface

Sevabot server interface is listening to port 5000. This interface offers

- Chat list (you need to know group chat id before you can send message into it)
- [Webhooks](#) for integrating external services

Just access the Sevabot server by going with your web browser to:

`http://localhost:5000`

Sevabot

A generic purpose hack together Skype bot at your service.

Status

Sevabot is running

Documentation

Please visit [Sevabot](#) project page for more information.

Admin

Chat list

Installing and running using Vagrant

- *Introduction*
- *Vagrant it*

3.1 Introduction

Vagrant is a tool to setup and deploy local virtual machines easily. Sevabot has a script for creating Vagrant deployments.

3.2 Vagrant it

Here is deployment instructions for deployment and automatic virtual machine configuration:

```
git clone https://github.com/opensourcehacker/sevabot.git
cd sevabot
vagrant box add precise64 http://files.vagrantup.com/precise64.box
vagrant up
```

Now you should have a virtual machine running having a runnign Sevabot in it.

TODO (these instructions might need someone to have a look of them as I don't use Vagrant myself -Mikko)

Chat commands

- *Introduction*
- *Out of the box commands*
- *Creating custom commands*
- *Stateful modules*
- *Running commands on remote servers*

4.1 Introduction

Sevabot supports commands you can type into group chat. All commands begin with !.

You can create your own commands easily as Sevabot happily executes any UNIX executable script.

4.2 Out of the box commands

Here are commands sevabot honours out of the box.

You can type them into the sevabot group chat.

- `!reload`: Reload current command scripts and print the list of available commands
- `!ping`: Check the bot is alive
- `!sad`: No woman, no cry
- `!weather`: Get weather by a city from openweathermap.org. Example: `!weather Toholampi`
- `!timeout`: Test timeouting commands
- `!soundcloud`: Get your soundcloud playlist (edit `soundcloud.rb` to make it work)
- `!dice`: Throw a dice
- `!tasks`: A simple ah-hoc group task manager for virtual team sprints
- `!call`: Conference call manager. Type `!call help` for more info.

4.3 Creating custom commands

The bot can use any UNIX executables printing to stdout as commands

- Shell scripts
- Python scripts, Ruby scripts, etc.

All commands must be in one of *modules* folders of the bot. The bot comes with some built-in commands like `ping`, but you can add your own custom commands by

- There is a `custom/` folder where you can place your own modules
- Enable `custom` folder in `settings.py`
- Create a script in `custom` folder. Example `myscript.sh`:

```
#!/bin/sh
echo "Hello world from my sevabot command"
```

- Add UNIX execution bit on the script using `chmod u+x myscript.sh`
- In Sevabot chat, type command `!reload` to reload all scripts
- Now you should see command `!myscript` in the command list
- The following environment variables are exposed to scripts `SKYPE_USERNAME`, `SKYPE_FULLNAME` from the person who executed the command

4.4 Stateful modules

You can have Python modules which maintain their state and have full access to Skype4Py instance. These modules can e.g.

- Perform timed background tasks with Skype
- Parse full Skype chat text, not just `!commands`
- Reach to calls, initiate calls
- Send SMS, etc.

Further info

- [Stateful module interface is described here](#)
- [Example task manager module is here](#)
- [Example conference call module is here](#)

4.5 Running commands on remote servers

The best way to execute commands on remote servers on UNIX is over SSH. Please read first the [‘basics how to setup SSH keys for the bot <http://opensourcehacker.com/2012/10/24/ssh-key-and-passwordless-login-basics-for-developers/>’](http://opensourcehacker.com/2012/10/24/ssh-key-and-passwordless-login-basics-for-developers/).

Below is an example `backup.sh` which checks

- disk space usage
- the timestamp

of backup folders on a backup server over SSH.

backup.sh:

```
#!/bin/sh

ssh root@example.com '
LOCATION="/srv/backup/backup/duply"
for l in $LOCATION/*; do
    S=`du -sh $l`
    TIME=`stat -c %y $l | cut -d " " -f1`
    BPATH=`echo $S | cut -f2`
    SIZE=`echo $S | cut -f1`

    echo -e "$SIZE\t$TIME\t$BPATH"
done
' #
```

You you need to install SSH keys on skype user to contact these servers:

```
ssh -a skype@sevabotserver.example.com

# Create key for the bot if one doesn't exist in .ssh/id_rsa
# Note: For safety reasons set passphrase. See reboot-seva script
# how passphrase enabled key is handled
ssh-keygen

# Copy the key to the remote server where you indent to run SSH commands
ssh-copy-id targetuser@targetserver.com
```

Sending Skype messages via webhooks

- *Introduction*
- *Supported services and examples*
- *Getting chat list*
- *Sending a message over HTTP interface*
- *Timed messages*

5.1 Introduction

Sevabot webhooks is a way to send Skype messages from external services using HTTP GET and POST requests.

Because there is no “webhook” standard Sevabot supports different ways to parse HTTP message payloads

- Signed and unsigned messages: shared secret MD5 signature prevents sending messages from hostile services
- HTTP GET and HTTP POST requests
- Service specific JSON payloads

To send a message to a chat you must first know to the id of a group chat. Sevabot server HTTP interface has a page to show this list (see below).

5.2 Supported services and examples

Here are some services and examples how to integrate Sevabot

5.2.1 Sending Skype messages from shell scripts

- *Introduction*

Introduction

These examples use an out-dated web API. Until the documentation is properly updated, you can post a message with the following commandline:

```
curl -data-urlencode chat_id="..." -data-urlencode message="..." -data-urlencode shared_secret="..."
http://localhost:5000/message/
```

See examples (bash specific)

- `send.sh`
- `ci-loop.bash`

5.2.2 Sending Skype messages from Python

- *Introduction*
- *Sending messages from separate URL thread*

Introduction

Here is an example how to send messages to Skype chat from external Python scripts and services. They do not need to be Sevabot commands, messages are send over HTTP interface.

Sending messages from separate URL thread

Here is an example (*original code* <<https://github.com/miohtama/collective.logbook/blob/master/collective/logbook/browser/webhook.py>> how to send a message asynchronously (does not executing the original code).

Example:

```
import socket
import threading
import urllib
import urllib2
import logging

logger = logging.getLogger(__name__) # Write errors to PYthon logging output

# Seconds of web service timeout
WEBHOOK_HTTP_TIMEOUT = 30

# Get Skype chat id from Sevabot web inteface
CHAT_ID = "xxx"

class UrlThread(threading.Thread):
    """
    A separate thread doing HTTP POST so we won't block when calling the webhook.
    """
    def __init__(self, url, data):
        threading.Thread.__init__(self)
        self.url = url
        self.data = data

    def run(self):
        original_timeout = socket.getdefaulttimeout()
        try:
            self.data = urllib.urlencode(self.data)
            socket.setdefaulttimeout(WEBHOOK_HTTP_TIMEOUT)
```

```

        r = urllib2.urlopen(self.url, self.data)
        r.read()
    except Exception as e:
        logger.error(e)
        logger.exception(e)
    finally:
        socket.setdefaulttimeout(original_timeout)

message = "Hello world"
t = UrlThread("http://sevabot.something.example.com:5000/message_unsigned/", {'message': message, 'ch

```

5.2.3 Zapier webhook support

- *Introduction*
- *Zapier Web hooks (raw HTTP POSTs)*
 - *Testing Zapier hook*

Introduction

zapier.com offers free mix-and-match different event sources to different triggers. The event sources includes popular services like Github, Dropbox, Salesforce, etc.

Zapier Web hooks (raw HTTP POSTs)

Zapier hook reads HTTP POST `data` variable payload to chat message as is. It is useful for other integrations as well.

- You need to register your *zap* in zapier.com
- *Sevabot* offers support for Zapier web hook HTTP POST requests
- Create a zap in zapier.com. Register. Add Webhooks *URL* with your bot info:

```
http://yourserver.com:5000/message_unsigned/
```

- Go to sevabot web interface and <http://yourserver.com:5000/> get chat id from Skype
- The following Zapier settings must be used: *Send as JSON: no*
- You need fill in HTTP POST fields *message* and *chat_id*

Example of Zapier *Data* field for Github issues:


```

message|New issue  {{title}} by {{user__login}} - {{html_url}}
chat_id|YOURCHATIDHERE

```

URL (required)

http://guinness.twinapex.fi:5000/message_unsigned/

**Payload Type** (optional)


Pay special attention to the proper mapping of the data below.

form

**Data** (optional)

If empty, takes the raw data from the trigger event. Else, line separated and pipe (|) separated key, value pairs sent as data.

msg|New issue ~ {{title}} ~ by {{user__login}} - {{html_url}}
chat|YOURCHATIDHERE|

**Basic Auth** (optional)**Testing Zapier hook**

You can use `curl` to test the hook from your server, for firewall issues and such:

```
curl --data-binary "msg=Hello world" --data-binary "chat=YOURCHATID" http://localhost:5000/message_unsigned/
```

Note: You need new enough curl version for `--data-binary`.

5.2.4 Github notifications to Skype

- *Introduction*
- *Commit notifications*
- *Issue notifications*

Introduction

Github notifications are provided through natively through Github and via [Zapier middleman service](#).

Commit notifications

Sevabot has built-in support for Github post-receive hook a.k.a. commit notifications.

To add one

- You need to be the repository admin
- Go *Admin > Service hooks* on Github
- Add Webhooks URL with your bot info:

```
http://yourserver.com:5000/github-post-commit/CHATID/SHAREDSECRET/
```

- Save
- Now you can use *Test hook* button to send a test message to the chat
- Following commits should come automatically to the chat

Issue notifications

Use *Zapier* webhook as described below.

This applies for

- New Github issues
- New Github comments

See generic [Zapier](#) instructions how to set-up the hook.

5.2.5 Subversion commit notifications

- *Introduction*

Introduction

Use the [provided shell script example](#) how to install a post-receive hook on your SVN server to send commit notifications to Skype.

5.2.6 Jenkins continuous integration notifications

- *Introduction*
- *Setting up a webhook*

Introduction

Jenkins is a popular open source continuous integration server.

Jenkins supports webhook notifications by using the Notification plugin: <https://wiki.jenkins-ci.org/display/JENKINS/Notification+Plugin>

The jenkins notifier will emit build status through skype.

Setting up a webhook

Install the plugin as directed in the above wiki link.

In Jenkins, for each build you want to send notifications for, under the ‘Job Notifications’ section, click ‘Add Endpoint’.

Enter your sevabot jenkins-notification endpoint, for example: <http://sevabot.example.com:5000/jenkins-notifier/{your-channel-id}/{your-shared-secret}/>

Trailing slash is important.

When a build completes, you should see the bot emit a message with the build status.

5.2.7 Zabbix alert messages from monitoring

- [Introduction](#)
- [Setting up a webhook](#)
- [Doing a agent alive check](#)

Introduction

Zabbix is a popular open source monitoring solution.

You can get Zabbix monitoring alerts like server down, disk near full, etc. to Skype with *Sevabot*.

Setting up a webhook

First you need to configure *Media* for your Zabbix user. The default user is called *Admin*.

Go to *Administrator > Media types*.

Add new media *Skype* with *Script name* **send.sh**.

Go to *Administrator > Users > Admin*. Open *Media* tab. Enable media *Skype* for this user. In the *Send to* parameter put in your *chat id* (see instructions above).

On the server running the Zabbix server process create a file `/usr/local/share/zabbix/alertscripts/send.sh`:

```
#!/bin/bash
#
# Example shell script for sending a message into sevabot
#
# Give command line parameters [chat id] and [message].
# The message is md5 signed with a shared secret specified in settings.py
# Then we use curl do to the request to sevabot HTTP interface.
#
```



```
#

# Chat id comes as Send To parameter from Zabbix
chat=$1

# Message is the second parameter
msg=$2

# Our Skype bot shared secret
secret="xxx"

# The Skype bot HTTP msg interface
msgaddress="http://yourserver.com:5000/msg/"

md5=`echo -n "$chat$msg$secret" | md5sum`

#md5sum prints a '-' to the end. Let's get rid of that.
for m in $md5; do
    break
done

curl $msgaddress -d "chat=$chat&msg=$msg&md5=$m"
```

Doing a agent alive check

Below is a sample Sevabot script which will do a Zabbix agent daemon check on all the servers.

See [commands](#) for how to configure SSH access for Sevabot to perform this functionality.

- Make a fake alert on all monitor servers, listed in ~/.ssh/config of Sevabot UNIX user
- Zabbix alert script will report back this alert from all servers where Zabbix agent is correctly running
- You need to add a special trigger in Zabbix which checks a timestamp of /home/zabbix/zabbix_test file, as touched by agents.sh run by Sevabot

Example monitoring item which keeps track of the file:

```
.. image:: /images/zabbix-item.png
```

width 500px

Note: Depending on the UNIX user home the touch file may be /var/run/zabbix/zabbix_test or /home/zabbix/zabbix_test You might need to manually switch Item state to Enabled after fixing this.

Example trigger:

Parent triggers [Twinapex](#)

Name

Expression

[Expression constructor](#)

vents generation ☐

Description

URL

Severity

Enabled ☒

Then the script we give to Sevabot to poke the file over SSH to generate Information notification in Zabbix and getting this notification back to our Zabbix monitoring Skype chat, confirming the agent is alive and well.

agents.sh:

```
#!/bin/bash
#

# Detect if we have a public key available
ssh-add -L > /dev/null

if [[ $? != "0" ]] ; then
    echo "Log-in as sevabot UNIX user and authorize SSH key"
    exit 1
fi

# Get list of hosts from SSH config file
HOSTS=`grep "Host " ~/.ssh/config | awk '{print $2}'`

# If some hosts don't have zabbix agents running, there's no need to use this script for them.
# Add this line to ~/.ssh/config:
# #NoAgents host1 host2
NOAGENT=`grep "#NoAgents " ~/.ssh/config | cut -d' ' -f2- | tr ' ' '\n'`

if [ -n "$NOAGENT" ]; then
    HOSTS=`echo -e "$HOSTS\n$NOAGENT" | sort | uniq -u`
fi
```

```
# Tell Sevabot what agents we are going to call
echo "Agents: $HOSTS" | tr '\n' ' '
echo

errors=0

# On each server touch a file to change its timestamp
# Zabbix monitoring system will detect this and
# report the alert back to Skype chat via a hook
for h in $HOSTS; do
    ssh -o "PasswordAuthentication no" $h "touch -m zabbix_test"
    if [[ $? != "0" ]] ; then
        echo "Failed to SSH to $h as sevabot UNIX user"
        errors=1
    fi
done

if [[ $errors == "0" ]] ; then
    echo "Succesfully generated zabbix_test ping on all servers"
fi
```

Example ~/.ssh/config:

```
Host xxx
User zabbix
Hostname xxx.twinapex.fi

Host yyy
User zabbix
Hostname yyy.twinapex.fi
```

Please note that you need to set up bot [SSH keys](#) for this.

Diagnosing

- If none of the agents is not replying your Zabbix host is probably messed up, reboot it: `/etc/init.d/zabbix-server restart`
- If some of the agents are replying manually restart non-replying agents

5.3 Getting chat list

To send messages through the bot you need to know

- Skype chat id - we use MD5 encoded ids to conveniently pass them in URLs.
- Sevabot shared secret in `settings.py` (only if your service supports MD5 signing, like your own custom shell script)

To get list of the chat ids visit in the Sevabot server hosted address:

```
http://localhost:5000/
```

It will return a HTTP page containing a list of Sevabot internal chat ids.

5.4 Sending a message over HTTP interface

One can send MD5 signed (safer) or unsigned messages (optional due to constraints in external services)

We provide

- signed endpoint <http://localhost:5000/msg/YOURCHATIT/> - see Bash example for more info
- unsigned endpoint http://localhost:5000/message_unsigned/ - takes in HTTP POST data parameters *chat_id* and *message*

5.5 Timed messages

Use external clocking service like [UNIX cron](#) to send regular or timed messages to Sevabot Skype chat over HTTP webhooks interface.

Troubleshooting

- *Logging*
- *Double messages*
- *Segfaults*
- *Skype4Py distribution for OSX*
- *Skype messages not coming through to bot interface*
- *Crashing on a startup on Ubuntu server*
- *Sevabot ignores commands and logs hang in sevabot - DEBUG - Attaching to Skype*

6.1 Logging

By default, Sevabot writes logging output to file `logs/sevabot.log`.

You can watch this log in real time with UNIX command:

```
tail -f logs/sevabot.log
```

To increase log level to max, edit `settings.py` and set:

```
LOG_LEVEL = "DEBUG"

DEBUG_HTTP = True
```

This will dump everything + HTTP request to the log.

6.2 Double messages

Sevabot replies to all commands twice.

Still no idea what could be causing this. Restarting everything helps.

6.3 Segfaults

If you get segfault on OSX make sure you are using 32-bit Python.

Debugging segmentation faults with Python.

Related gdb dump:

```
Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x000000001243b68
0x00007fff8c12d878 in CFRetain ()
(gdb) bt
#0  0x00007fff8c12d878 in CFRetain ()
#1  0x00000001007e07ec in ffi_call_unix64 ()
#2  0x00007fff5fbfb50 in ?? ()
(gdb) c
Continuing.

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x000000001243b68
0x00007fff8c12d878 in CFRetain ()
```

6.4 Skype4Py distribution for OSX

Currently Skype4Py distribution is broken.

To fix this do:

```
source venv/bin/activate
git clone git://github.com/stigkj/Skype4Py.git
cd Skype4Py
arch -i386 python setup.py install
```

6.5 Skype messages not coming through to bot interface

Symptoms

- Skype is running in Xvfb
- Sevabot logs in screen don't see incoming chat messages

Seems to happen if you reboot the bot in too fast cycle. Maybe Skype login has something which makes it not working if you log several times in a row.

Looks like it fixes itself if you just wait a bit before sending messages to the chat.

6.6 Crashing on a startup on Ubuntu server

Segfault when starting up the bot:

```
File "build/bdist.linux-i686/egg/Skype4Py/skype.py", line 250, in __init__
File "build/bdist.linux-i686/egg/Skype4Py/api/posix.py", line 40, in SkypeAPI
File "build/bdist.linux-i686/egg/Skype4Py/api/posix_x11.py", line 254, in __in
Skype4Py.errors.SkypeAPIError: Could not open XDisplay
Segmentation fault (core dumped)
```

This usually means that your DISPLAY environment variable is wrong.

Try:

```
export DISPLAY=:1
```

or:

```
export DISPLAY=:0
```

depending on your configuration before running Sevabot.

6.7 Sevabot ignores commands and logs hang in sevabot - DEBUG - Attaching to Skype

This concerns only Ubuntu headless server deployments.

Your fluxbox might have hung. Kill it with fire:

```
killall -SIGKILL fluxbox
```

Restart.

Community and development

- *Introduction*
- *IRC*
- *Support tickets and issues*
- *Installing development version*
- *Debugging*
- *Contributions*
- *Releases*

7.1 Introduction

How to participate to the spectacular future of Sevabot. You can make the life of Sevabot better - and yours too!

7.2 IRC

For chatting

/join #opensourcehacker @ irc.freenode.net

Note: due to low activity of the channel prepare to idle there for 24 hours to wait for the answer.

7.3 Support tickets and issues

Use Github issue tracker

7.4 Installing development version

All development happens in `dev` branch.

How to install and run the development version (trunk) of Sevabot:

```
git clone git://github.com/opensourcehacker/sevabot.git
cd sevabot
git checkout dev
curl -L -o virtualenv.py https://raw.githubusercontent.com/pypa/virtualenv/master/virtualenv.py
python virtualenv.py venv # prefix with arch -i386 on OSX
source venv/bin/activate
python setup.py develop # prefix with arch -i386 on OSX
```

7.5 Debugging

You might want to turn on `DEBUG_HTTP` setting to dump out incoming HTTP requests if you are testing / developing your own hooks.

7.6 Contributions

All contributions must come with accompanying documentation updates.

All Python files must follow PEP-8 coding conventions and be [flake8](#) valid.

Submit pull request at Github.

For any changes update [CHANGES.rst](#).

7.7 Releases

Use [zest.releaser](#)

See [Github](#) for more project information.

Trademark notice

The Skype name, associated trade marks and logos and the “S” logo are trade marks of Skype or related entities. Sevabot is an open source project and not associate of Microsoft Corporation or Skype.