KasaDaka Development Guide

## Changing the python code

Flask has an useful built-in development mode. It automatically refreshes the server when a change in the code is detected. Change debug to True in config.py, to enable debugging reports, whenever your application crashes.

For more information: <http://flask.pocoo.org/docs/0.10/quickstart/>

## Debugging VXML files

When a vxml document is not functioning as expected, the VXML interpreter does not give any hints on why this is happening. The call is just suddenly terminated.

To get some logs:

1. stop the asterisk service: sudo /etc/init.d/asterisk stop
2. start asterisk manually with a CLI: asterisk -cvvv
3. tell openvxi to give more info: vxml debug interpreter

Now, call your service with your phone until the error occurs. There is a symlink to the log file already made for you. This is accessible at: [http://IP\_ADDRESS/log.txt](http://ip_address/log.txt). Scroll all the way to the bottom and go up until you see errors. Everything the interpreter does is posted, so don’t worry about skipping a lot of lines!

Example of an useful error:

|  |
| --- |
| Jan 18 17:24:57.64|1985049680|0|8010|DEV|Parse error in file "http://127.0.0.1/FlaskKasadaka/placeProductOffer.vxml?lang=en", line 485, column 16 - either the 'src' attribute or inlined content may be provided, not both |

After you are done, close the asterisk CLI and restart the service again.

Tip: when developing, use detaching from a byobu/tmux session to keep the cli open, even when not connected to the RPi!

Tip: an online XML validator can also be useful to find errors!

## Using Text to Speech to generate audio files

When you quickly want to generate some audio files, and do not want to spend a lot of time in Audacity, you can use a TTS engine. A free, lightweight and open-source engine has been installed in the installation guide. This program is called flite. It produces wav files in the correct format for VXI, so they can be used without further work.

Example to generate an audio file:

|  |
| --- |
| flite -t "Hello World!" -o helloworld.wav |

This will save in the current directory.

NOTE: the speech is not very spectacular, but sufficient in English. Other languages are not recommended.

## Audio files availability page

When developing a new use case, it is essential to have voicelabels for all possible spoken subjects in your application. These voicelabels also have to reference an existing wave sound file. Also, all the audio files used in the interface (asking questions, etc) have to exist.

To assist you in this, there is a page that scans all your python files and templates, and scans the triple store for audio files, and checks whether these files are accessible. Subjects in the triple store without a voicelabel are also shown.

The page can be accessed at:

[http://IP\_ADDRESS/FlaskKasadaka/audioreferences.html](http://ip_address/FlaskKasadaka/audioreferences.html)

## Wav files requirements

VXI is picky about audio files. Supported audio files are listed here: <https://wiki.i6net.org/doku.php/vxi_installation_guide:management:start>

# Python code documentation

When you are planning on customizing the example to suit your use-case, some basic knowledge of Flask is recommended. Please read the Flask documentation: <http://flask.pocoo.org/docs/0.10/quickstart/>

There are 3 python files used:

* \_\_init\_\_.py
* config.py
* sparqlInterface.py

### \_\_init\_\_.py

This is the main file, which contains the different possible vxml/html files that can be requested.

For the functions of requesting offers of a product and posting an offer, the user input is sent back to the function through HTTP GET. This is an easy way of sending information, and also makes it easier to debug, by accessing the VXML files through a browser.

### config.py

Contains some configuration variables, and also handles the language support.

### sparqlInterface.py

Contains methods for accessing and updating the triple store through SPARQL queries.

## Templates

### audiofiles.html

Template for checking the status of necessary audio files.

### language.vxml

Template to give the user a choice of language to use. Plays back the language choices in the respective language. When a choice is made, the user is sent to language.vxml, with the language as a parameter (HTTP GET).

### main.vxml

Gives a menu from menu.vxml, but with a welcome message played back first.

### menu.vxml

Gives the user a menu of choices. When a choice is made, the user is sent to the corresponding URL.

### message.vxml

Plays back a message, and forwards the user to the provided URL.

### placeProductOffer.vxml

Asks the user to enter the necessary fields to place a product offer. When everything is filled, it redirects the user to the same URL, but with the inputs as parameters (HTTP GET).

### result.vxml

Plays back a matrix of results, then forwards the user.

### vxmlLayout.vxml

The basic VXML layout, used in all other templates.