Aria

1.0.0.2

Generated by Doxygen 1.8.11

Contents

Chapter 1

Aria - Digital Assistant - Code documentation

1.1 Dependencies

pydispatch - http://pydispatcher.sourceforge.net/
pydev - http://www.pydev.org/
dateutil - https://dateutil.readthedocs.io/en/stable/
keyring - https://pypi.org/project/keyring/
facebook SDK - https://github.com/mobolic/facebook-sdk
telegram - https://github.com/python-telegram-bot/python-telegram-bot
emoji - https://github.com/carpedm20/emoji/

Configuration file

```
[Debug]
Debug = yes
Host = 192.168.0.150
Port = 5678

[Modules]
Disabled =
Path = plugins

[Classes]
Disabled = Wit,WeatherData,Gui,emojize,telegram,telegram.ext,spyPlugin
```

Logger configuration

```
## @file
# Main logger configuration file
[loggers]
keys=root,moduleTTS,moduleSTT,Audio,moduleJoke,moduleWeather,moduleGui,moduleEmail,moduleTelegram,moduleSp

[handlers]
keys=consoleHandler,moduleTTS,moduleSTT,moduleJoke,moduleWeather,moduleEmail,moduleTelegram,moduleSpy

[formatters]
keys=consoleFormatter
###############################################################################
[logger_root]
level=DEBUG
handlers=consoleHandler

[logger_Audio]
level=INFO
handlers=consoleHandler
```

```
propagate=0
qualname=Audio

[logger_moduleTTS]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleTTS

[logger_moduleSTT]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleSTT

[logger_moduleJoke]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleJoke

[logger_moduleWeather]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleWeather

[logger_moduleGui]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleGui

[logger_moduleEmail]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleEmail

[logger_moduleTelegram]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleTelegram

[logger_moduleSpy]
level=DEBUG
handlers=consoleHandler
propagate=0
qualname=moduleSpy
#############################################################################
[handler_consoleHandler]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_Audio]
class=StreamHandler
level=INFO
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleTTS]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleSTT]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
```

```
args=(sys.stdout,)

[handler_moduleJoke]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleWeather]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleGui]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleEmail]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleTelegram]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)

[handler_moduleSpy]
class=StreamHandler
level=DEBUG
formatter=consoleFormatter
args=(sys.stdout,)
#########################################################################

[formatter_consoleFormatter]
format=%(filename)-10s[LINE:%(lineno)-4d]# %(levelname)-8s [%(asctime)s]  %(message)s
datefmt=%H:%M:%S
```

Chapter 2

Todo List

Member plugins.SttPlugin.STT._wav_analyze (self, filename)

Remove silence

Chapter 3

Bug List

Member plugins.guiPlugin.Gui._animation_update (self)

This function may cause high CPU load

Member plugins.guiPlugin.Gui.main_pic_animation (self)

Due to policy of Facebook application without server may use only short period access token

Chapter 4

Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 5

Hierarchical Index

5.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 6

Class Index

6.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

Chapter 8

Namespace Documentation

8.1 Aria Namespace Reference

Functions

- def clean_exit ()

  Cleanup function.
- def aria_start ()

  Startup function.
- def emergency_shutdown ()

Variables

- shutdown_flag = threading.Event()

8.1.1 Function Documentation

8.1.1.1 def Aria.aria_start ( )

Startup function.

Start all services and load runners

Note

Load logger configuration, and run settings

See also

https://docs.python.org/2/library/logging.html

Definition at line 57 of file Aria.py.

8.1.1.2 def Aria.clean_exit ( )

Cleanup function.

Close all debug/log session and perform clean exit

Definition at line 40 of file Aria.py.

8.1.1.3 def Aria.emergency_shutdown ( )

Definition at line 182 of file Aria.py.

8.1.2 Variable Documentation

8.1.2.1 Aria.shutdown_flag = threading.Event()

Definition at line 189 of file Aria.py.

8.2 plugins Namespace Reference

Namespaces

- AudioPlugin
- EmailPlugin
- guiPlugin
- JokePlugin
- SttPlugin
- TelegramPlugin
- TtsPlugin
- WeatherPlugin

8.3 plugins.AudioPlugin Namespace Reference

Classes

- class AudioSubSystem

  AudioSubSystem package.

8.4 plugins.EmailPlugin Namespace Reference

Classes

- class ZohoEmail

  Email plugin.

8.5 plugins.guiPlugin Namespace Reference

Classes

- class Gui

  Main GUI.
- class GuiPlugin

  Init wx-python GUI.

8.6 plugins.JokePlugin Namespace Reference

Classes

- class Humour

8.7 plugins.SttPlugin Namespace Reference

Classes

- class STT

  Speech to Text abstraction.

8.8 plugins.TelegramPlugin Namespace Reference

Classes

- class TelegramBot

  Additional user interface.

8.9 plugins.TtsPlugin Namespace Reference

Classes

- class TTS

  Test To Speech abstraction.

8.10 plugins.WeatherPlugin Namespace Reference

Classes

- class Weather

  Interaction with OPenWeatherMap.
- class WeatherData

  Hold forecast data.

8.11 TelegramBot Namespace Reference

8.11.1 Detailed Description

Create additional User interface using Telegram bot API

See also

https://core.telegram.org/api

Configuration file

```
[API]
system = Telegram
user = Aria
login = Shepard
[System]
temp_folder = /home/pi/Aria2/tmp/telegram/
[Camera]
angle=90
```

Message file

```
{ "welcome_morning":[
    "As the day begins, remember that I am your friend...you're welcome!",
    "Good Morning! Working with you is always interesting! Not good or bad, just interesting!",
    "Good Morning! May your smile be brighter than the sun today and your laughter be spread far and wide.
  ],
  "welcome_afternoon": [
    "Have a wonderful afternoon! I love the pleasure of your company...",
    "Success is never permanent, failure is never complete. The important thing is the courage to fight. G
    "Never fail to be aware of the difference between waiting, and wasting time. Good afternoon!"
  ],
  "welcome_evening":[
     "Evening is a time of real experimentation. You never want to look the same way",
     "Mornings, hurried and stressful. Afternoons, slow and woeful. Nights, time to rest. Evenings, simply
     "Evenings... a reason to come back home, look forward to a good meal and spend time with loved ones."

  ],
  "welcome_night":[
     "Thinking about you mode activated! Missing you in progress! If you are awake reply to deactivate thi
     "The good people sleep much better at night than the bad people. Of course, the bad people enjoy the
     "The future is shaped by your dreams, so stop wasting time and go to sleep! Good night!",
     "Are you still awake ?"
  ],
  "authorization_require":[
     "I know it's silly, but anyway can you tell me a password ?",
     "I'm sure that you, but my boss tell me to ask password",
     "Remember secret password that we have? Can you tell me what it was ?",
     "Only cops and vampires have to have an invitation to enter. Others need password"
  ],
   "authorization_fail":[
     "HELP!!HELP!! Hacker attack! Or maybe incorrect password",
     "YOU SHOULD NOT PASS!! Until you tell me correct password"
   ],
   "authorization_successful":[
     "Welcome home, good hunter",
     "Grant access - done , grant superuser access - done, grant superpower - in progress",
     "It you, I always knew that was you"
   ]}
```

Chapter 9

Class Documentation

9.1 plugins.AudioPlugin.AudioSubSystem Class Reference

AudioSubSystem package.

Public Member Functions

- def __init__ (self)

  Create Audio subsystem instance.
- def __del__ (self)

  Stop module.
- def start_hot_word_detection (self, delay=None)

  WaitToHotWord event wrapper.
- def play_file (self, filename, delay=None, callback=None)

  PlayFile event wrapper.
- def record_file (self, filename, record_time=None, delay=None, callback=None)

  RecordFile event wrapper.

Static Public Attributes

- string version = "1.0.0.0"

  Plugin version.
- string description = "Audio sub-system"

  Short plugin description.

Private Member Functions

- def _start_hot_word_detection (self, delay=None)

  WaitToHotWord thread.
- def _play_file (self, filename, delay=None, callback=None)

  PlayFile thread.
- def _record_file (self, filename, record_time, delay=None, callback=None)

  Record thread.

Private Attributes

- _hot_word_detection_active

  Event objectallow synchronize audio file play/record and STT engine.
- _io_system_busy

  Allow bypass through Raspberry Pi IO system bug.
- _exit_flag

  Shutdown eventsignaling to all thread exit.
- _gui_speaker_status_uuid

  Unique id for speaker try icon.
- _gui_microphone_status_uuid

  Unique id for microphone try icon.
- _logger

  logger instance
- _config

  configuration file instnce
- _hot_words

  List of activate words.
- _recognition_engine

  command line to activate hot-word detection engine
- _playback_engine

  command line to activate file playback
- _max_record_time

  Maximum allowed voice record time.
- _record_engine

  command line to activate record engine

9.1.1 Detailed Description

AudioSubSystem package.

Allow sound playing and recording

Version

1.0.0.0

Definition at line 21 of file AudioPlugin.py.

9.1.2 Constructor & Destructor Documentation

9.1.2.1 def plugins.AudioPlugin.AudioSubSystem.__init__ (*self*)

Create Audio subsystem instance.

Create and initialize instance

Exceptions

ImportError	Configuration or IO system errorModule will be unloaded.

Registering on events:

WaitToHotWordWait until Hot-Word not detected in audio input and send notification.
PlayFilePlay audio file.
RecordFileRecord audio input into file

Generate events:

HotWordDetectionActiveSet to True when STT engine trying to detect hot-word in audio stream.
GuiNotificationGUI tray update.
HotWordDetectedHot-word detected. PlaybackActiveSet to True when audio player play file.
RecordActiveSet to True when audio recorder record audio into file.

Definition at line 43 of file AudioPlugin.py.

9.1.2.2 def plugins.AudioPlugin.AudioSubSystem.__del__ (*self*)

Stop module.

Stop all module thread and sub-programs

Definition at line 123 of file AudioPlugin.py.

9.1.3 Member Function Documentation

9.1.3.1 def plugins.AudioPlugin.AudioSubSystem._play_file (*self, filename, delay* = `None`, *callback* = `None`) `[private]`

PlayFile thread.

Communicate with audio player

Parameters

filename	string Path to audio file
delay	float Delay before start STT engine - optional. Default - 0
callback	obj Callback function when playback completed - optional. Default - None

Warning

This function should not be called from outside

Generate events:

PlaybackActiveSet to True when audio player play file.
GuiNotificationGUI tray update.

See also

guiPlugin

Definition at line 229 of file AudioPlugin.py.

9.1.3.2 def plugins.AudioPlugin.AudioSubSystem._record_file (*self, filename, record_time, delay =* None, *callback =* None) [private]

Record thread.

Communicate with audio recorder

Parameters

filename	string Path to audio file
record_time	float Record time - optional. Default - maximum allowed time as set in config file
delay	float Delay before start STT engine - optional. Default - 0
callback	obj Callback function when playback completed - optional. Default - None

Warning

This function should not be called from outside

Generate events:

RecordActive - Set to True when audio recorder record audio into file.
GuiNotification - GUI tray update.

See also

guiPlugin

Definition at line 287 of file AudioPlugin.py.

9.1.3.3 def plugins.AudioPlugin.AudioSubSystem._start_hot_word_detection (*self, delay =* None) [private]

WaitToHotWord thread.

Communicate with STT engine

Parameters

delay	float Delay before start STT engineoptional. Default - 0

Warning

This function should not be called from outside

Generate events:

HotWordDetectionActiveSet to True when STT engine trying to detect hot-word in audio stream.
GuiNotificationGUI tray update.
HotWordDetectedHot-word detected.

See also

guiPlugin

Definition at line 160 of file AudioPlugin.py.

9.1.3.4 def plugins.AudioPlugin.AudioSubSystem.play_file (*self, filename, delay* = None, *callback* = None)

PlayFile event wrapper.

Initialize thread that allow to communication audio player

Parameters

filename	string Path to audio file
delay	float Delay before start STT engine.Optional. Default - 0
callback	obj Callback function when playback completed.Optional. Default - None

Definition at line 208 of file AudioPlugin.py.

9.1.3.5 def plugins.AudioPlugin.AudioSubSystem.record_file (*self, filename, record_time* = None, *delay* = None, *callback* = None)

RecordFile event wrapper.

Initialize thread that allow to communication audio recorder

Parameters

filename	string Path to audio file
record_time	float Record timeoptional. Default - maximum allowed time as set in config file
delay	float Delay before start STT engine - optional. Default - 0
callback	obj Callback function when playback completed - optional. Default - None

Definition at line 261 of file AudioPlugin.py.

9.1.3.6 def plugins.AudioPlugin.AudioSubSystem.start_hot_word_detection (*self, delay* = None)

WaitToHotWord event wrapper.

Initialize thread that allow to communication with STT engine

Parameters

delay	float Delay before start STT engineoptional. Default - 0

Definition at line 137 of file AudioPlugin.py.

9.1.4 Member Data Documentation

9.1.4.1 plugins.AudioPlugin.AudioSubSystem._config [private]

configuration file instnce

Definition at line 65 of file AudioPlugin.py.

9.1.4.2 plugins.AudioPlugin.AudioSubSystem._exit_flag [private]

Shutdown eventsignaling to all thread exit.

Definition at line 49 of file AudioPlugin.py.

9.1.4.3 plugins.AudioPlugin.AudioSubSystem._gui_microphone_status_uuid [private]

Unique id for microphone try icon.

Definition at line 53 of file AudioPlugin.py.

9.1.4.4 plugins.AudioPlugin.AudioSubSystem._gui_speaker_status_uuid [private]

Unique id for speaker try icon.

Definition at line 51 of file AudioPlugin.py.

9.1.4.5 plugins.AudioPlugin.AudioSubSystem._hot_word_detection_active [private]

Event objectallow synchronize audio file play/record and STT engine.

Definition at line 45 of file AudioPlugin.py.

9.1.4.6 plugins.AudioPlugin.AudioSubSystem._hot_words [private]

List of activate words.

Definition at line 72 of file AudioPlugin.py.

9.1.4.7 plugins.AudioPlugin.AudioSubSystem._io_system_busy [private]

Allow bypass through Raspberry Pi IO system bug.

Only one instance can control audio system

Definition at line 47 of file AudioPlugin.py.

9.1.4.8 plugins.AudioPlugin.AudioSubSystem._logger [private]

logger instance

Definition at line 57 of file AudioPlugin.py.

9.1.4.9 plugins.AudioPlugin.AudioSubSystem._max_record_time [private]

Maximum allowed voice record time.

Definition at line 93 of file AudioPlugin.py.

9.1.4.10 plugins.AudioPlugin.AudioSubSystem._playback_engine [private]

command line to activate file playback

Definition at line 87 of file AudioPlugin.py.

9.1.4.11 plugins.AudioPlugin.AudioSubSystem._recognition_engine [private]

command line to activate hot-word detection engine

Definition at line 81 of file AudioPlugin.py.

9.1.4.12 plugins.AudioPlugin.AudioSubSystem._record_engine [private]

command line to activate record engine

Definition at line 95 of file AudioPlugin.py.

9.1.4.13 string plugins.AudioPlugin.AudioSubSystem.description = "Audio sub-system" [static]

Short plugin description.

Definition at line 25 of file AudioPlugin.py.

9.1.4.14 string plugins.AudioPlugin.AudioSubSystem.version = "1.0.0.0" [static]

Plugin version.

Definition at line 23 of file AudioPlugin.py.

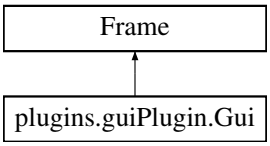
The documentation for this class was generated from the following file:

- plugins/AudioPlugin.py

9.2 plugins.guiPlugin.Gui Class Reference

Main GUI.

Inheritance diagram for plugins.guiPlugin.Gui:



Public Member Functions

- def __init__ (self, args, kwds)

  Create GUI based on wx python.
- def __del__ (self)

  Stop module.
- def safe_update (self, func, data)

  GUI element update.
- def safe_update_delay (self, func, data)

  Wrapper for delayed GUI update.
- def main_pic_animation (self)

  Download image from Facebook.

Public Attributes

- api_key
- weather_desc_lbl
- weather_wind_lbl

Private Member Functions

- def __set_properties (self)

  Set GUI properties.
- def __do_layout (self)

  Set GUI layout.
- def _safe_update_delay (self, func, data)

  Delayed GUI update.
- def _notification (self, source, icon_path)

  Wrapper for tray update.
- def _animation_update (self)

  Main picture animation.
- def _system_response_text (self, text)

  Wrapper for SayText event.
- def _user_request_text (self, entities, raw_text)

  Wrapper for SpeechRecognize event.
- def _time_update (self)

  Time update.
- def _weather_display (self, description, temp, wind, icon)

  Wrapper for WeatherUpdate event.

Private Attributes

- _shutdown

  Shutdown flag - notify to threads exits.
- _gui_update_lock

  GUI synchronization - avoid GUI update from different threads.
- _notification_tray

  dictionary of notification icons and their owners
- _logger
- _config
- _animation_steps
- _clear_delay
- _change_after
- _animation_speed
- _ignore_albums
- _animation_active
- _temp_folder
- _animation_circle_bmp
- _system_response_bmp
- _system_response_lbl
- _user_request_lbl
- _clock_lbl
- _date_lbl
- _weather_temp_lbl
- _weather_icon
- _main_picture_bmp
- _notification_slots

9.2.1 Detailed Description

Main GUI.

Create GUI interface with Facebook profile pictures

See also

https://developers.facebook.com/tools/explorer/145634995501895/?method=GET&path=&version=v2.7

Version

1.0.0.0

Definition at line 30 of file guiPlugin.py.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 def plugins.guiPlugin.Gui.__init__ (*self, args, kwds*)

Create GUI based on wx python.

Create and initialize instance start fetching and periodic update threads

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Registering on events:

GuiNotification - User speech input.
SayText - System to user response text.
SpeechRecognize - User to system text.
WeatherUpdate - Periodic weather update.

See also

SttPlugin
WeatherPlugin
AudioSubSystem

Definition at line 43 of file guiPlugin.py.

9.2.2.2 def plugins.guiPlugin.Gui.__del__ (*self*)

Stop module.

Stop all module thread and sub-programs

Definition at line 146 of file guiPlugin.py.

9.2.3 Member Function Documentation

9.2.3.1 def plugins.guiPlugin.Gui.__do_layout (*self*) [private]

Set GUI layout.

Update GUI elements and their layouts

Note

This function generated create be WxGlade

Warning

This function should not be called from outside

Definition at line 164 of file guiPlugin.py.

9.2.3.2 def plugins.guiPlugin.Gui.__set_properties (*self*) [private]

Set GUI properties.

Update GUI elements and their properties

Note

This function generated create be WxGlade

Warning

This function should not be called from outside

Definition at line 155 of file guiPlugin.py.

9.2.3.3 def plugins.guiPlugin.Gui._animation_update (*self*) [private]

Main picture animation.

Create slow change effect of main picture

Warning

This function should not be called from outside

Bug This function may cause high CPU load

Definition at line 292 of file guiPlugin.py.

9.2.3.4 def plugins.guiPlugin.Gui._notification (*self, source, icon_path*) [private]

Wrapper for tray update.

Thread safe tray update

Parameters

source	- Unique id of caller
icon_path	- Relative path of icon for tray

Definition at line 267 of file guiPlugin.py.

9.2.3.5 def plugins.guiPlugin.Gui._safe_update_delay (*self, func, data*) [private]

Delayed GUI update.

Thread safe GUI update with delay

Parameters

func	- WxPython function
data	- Configuration data for WxPython update function

Definition at line 259 of file guiPlugin.py.

9.2.3.6 def plugins.guiPlugin.Gui._system_response_text (*self, text*) [private]

Wrapper for SayText event.

Write TTS input text on screen with typing animation

Parameters

text	- Text to TTS engine

Definition at line 304 of file guiPlugin.py.

9.2.3.7 def plugins.guiPlugin.Gui._time_update (*self*) [private]

Time update.

Update Time, and Date in GUI window

Definition at line 322 of file guiPlugin.py.

9.2.3.8 def plugins.guiPlugin.Gui._user_request_text (*self, entities, raw_text*) [private]

Wrapper for SpeechRecognize event.

Write STT output text on screen with typing animation

Parameters

entities	- Ignored
raw_text	- Text from STT engine

Definition at line 314 of file guiPlugin.py.

9.2.3.9 def plugins.guiPlugin.Gui._weather_display (*self, description, temp, wind, icon*) [private]

Wrapper for WeatherUpdate event.

Display weather info in GUI

Parameters

description	- Short weather description
temp	- Current temperature
wind	- Short wind description
icon	Path to weather icon. Provided by OpenWeatherMap

Definition at line 336 of file guiPlugin.py.

9.2.3.10 def plugins.guiPlugin.Gui.main_pic_animation (*self*)

Download image from Facebook.

Download image for future processing (animation). Small images and ignored albums ignored

Bug Due to policy of Facebook application without server may use only short period access token

Definition at line 345 of file guiPlugin.py.

9.2.3.11 def plugins.guiPlugin.Gui.safe_update (*self, func, data*)

GUI element update.

Thread safe GUI update

Parameters

func	- WxPython function
data	- Configuration data for WxPython update function

Definition at line 243 of file guiPlugin.py.

9.2.3.12 def plugins.guiPlugin.Gui.safe_update_delay (*self, func, data*)

Wrapper for delayed GUI update.

Thread safe GUI update with delay

Parameters

func	- WxPython function
data	- Configuration data for WxPython update function

Definition at line 252 of file guiPlugin.py.

9.2.4 Member Data Documentation

9.2.4.1 plugins.guiPlugin.Gui._animation_active [private]

Definition at line 80 of file guiPlugin.py.

9.2.4.2 plugins.guiPlugin.Gui._animation_circle_bmp [private]

Definition at line 100 of file guiPlugin.py.

9.2.4.3 plugins.guiPlugin.Gui._animation_speed [private]

Definition at line 75 of file guiPlugin.py.

9.2.4.4 plugins.guiPlugin.Gui._animation_steps [private]

Definition at line 72 of file guiPlugin.py.

9.2.4.5 plugins.guiPlugin.Gui._change_after [private]

Definition at line 74 of file guiPlugin.py.

9.2.4.6 plugins.guiPlugin.Gui._clear_delay [private]

Definition at line 73 of file guiPlugin.py.

9.2.4.7 plugins.guiPlugin.Gui._clock_lbl [private]

Definition at line 106 of file guiPlugin.py.

9.2.4.8 plugins.guiPlugin.Gui._config [private]

Definition at line 59 of file guiPlugin.py.

9.2.4.9 plugins.guiPlugin.Gui._date_lbl [private]

Definition at line 107 of file guiPlugin.py.

9.2.4.10 plugins.guiPlugin.Gui._gui_update_lock [private]

GUI synchronization - avoid GUI update from different threads.

Definition at line 47 of file guiPlugin.py.

9.2.4.11 plugins.guiPlugin.Gui._ignore_albums [private]

Definition at line 77 of file guiPlugin.py.

9.2.4.12 plugins.guiPlugin.Gui._logger [private]

Definition at line 51 of file guiPlugin.py.

9.2.4.13 plugins.guiPlugin.Gui._main_picture_bmp [private]

Definition at line 114 of file guiPlugin.py.

9.2.4.14 plugins.guiPlugin.Gui._notification_slots [private]

Definition at line 116 of file guiPlugin.py.

9.2.4.15 plugins.guiPlugin.Gui._notification_tray [private]

dictionary of notification icons and their owners

Definition at line 49 of file guiPlugin.py.

9.2.4.16 plugins.guiPlugin.Gui._shutdown [private]

Shutdown flag - notify to threads exits.

Definition at line 45 of file guiPlugin.py.

9.2.4.17 plugins.guiPlugin.Gui._system_response_bmp [private]

Definition at line 101 of file guiPlugin.py.

9.2.4.18 plugins.guiPlugin.Gui._system_response_lbl [private]

Definition at line 103 of file guiPlugin.py.

9.2.4.19 plugins.guiPlugin.Gui._temp_folder [private]

Definition at line 82 of file guiPlugin.py.

9.2.4.20 plugins.guiPlugin.Gui._user_request_lbl [private]

Definition at line 104 of file guiPlugin.py.

9.2.4.21 plugins.guiPlugin.Gui._weather_icon [private]

Definition at line 112 of file guiPlugin.py.

9.2.4.22 plugins.guiPlugin.Gui._weather_temp_lbl [private]

Definition at line 110 of file guiPlugin.py.

9.2.4.23 plugins.guiPlugin.Gui.api_key

Definition at line 64 of file guiPlugin.py.

9.2.4.24 plugins.guiPlugin.Gui.weather_desc_lbl

Definition at line 109 of file guiPlugin.py.

9.2.4.25 plugins.guiPlugin.Gui.weather_wind_lbl

Definition at line 111 of file guiPlugin.py.

The documentation for this class was generated from the following file:

- plugins/guiPlugin.py

9.3 plugins.guiPlugin.GuiPlugin Class Reference

Init wx-python GUI.

Public Member Functions

- def __init__ (self)
  Start GUI.

Static Public Attributes

- string version = '1.0.0.0'
  Plugin version.
- string description = 'GUI'
  Plugin description.

Private Member Functions

- def _gui_thread (self)
  Initialize GUI.

9.3.1 Detailed Description

Init wx-python GUI.

Create GUI interface

Version

1.0.0.0

Definition at line 392 of file guiPlugin.py.

9.3.2 Constructor & Destructor Documentation

9.3.2.1 def plugins.guiPlugin.GuiPlugin.__init__ (*self*)

Start GUI.

Start GUI initialization in thread

Definition at line 400 of file guiPlugin.py.

9.3.3 Member Function Documentation

9.3.3.1 def plugins.guiPlugin.GuiPlugin._gui_thread (*self*) [private]

Initialize GUI.

Create GUI frame and continue run in daemon thread mode

Definition at line 405 of file guiPlugin.py.

9.3.4 Member Data Documentation

9.3.4.1 string plugins.guiPlugin.GuiPlugin.description = 'GUI' [static]

Plugin description.

Definition at line 396 of file guiPlugin.py.

9.3.4.2 string plugins.guiPlugin.GuiPlugin.version = '1.0.0.0' [static]

Plugin version.

Definition at line 394 of file guiPlugin.py.

The documentation for this class was generated from the following file:

- plugins/guiPlugin.py

9.4 Humor Class Reference

Fun response module.

9.4.1 Detailed Description

Fun response module.

Make everyone smile

Version

1.0.0.0

The documentation for this class was generated from the following file:

- plugins/JokePlugin.py

9.5 plugins.JokePlugin.Humour Class Reference

Public Member Functions

- def __init__ (self)
  Initialize humor.
- def joke (self, entities, raw_text)
  Wrapper for SpeechRecognize event.

Static Public Member Functions

- def joke_done ()
  Restart user interaction.

Public Attributes

- humour

Static Public Attributes

- string version = '1.0.0.0'
  Plugin version.
- string description = 'Humour sense'
  Short plugin description.

Private Member Functions

- def _joke (self, entities, raw_text)
  Response with joke.

Private Attributes

- _logger

9.5.1 Detailed Description

Definition at line 20 of file JokePlugin.py.

9.5.2 Constructor & Destructor Documentation

9.5.2.1 def plugins.JokePlugin.Humour.__init__ (*self*)

Initialize humor.

Initialize humor sense. Everyone should have one

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded. Not funny

Registering on events:

SpeechRecognize - User to system text.

Generate events:

SpeechAccepted - Notify that module start process user request.
RestartInteraction - Restart Hot-Word detection.
SayText - Response to user request using TTS engine.
PlayFile - Response with audio file

See also

AudioSubSystem
TtsPlugin
SttPlugin

Definition at line 41 of file JokePlugin.py.

9.5.3 Member Function Documentation

9.5.3.1 def plugins.JokePlugin.Humour._joke (*self, entities, raw_text*) [private]

Response with joke.

Search for text in humor file and if found response with text file

Parameters

entities	- Ignored
raw_text	- Text from STT engine

Generate events:

SpeechAccepted - Notify that module start process user request.
RestartInteraction - Restart Hot-Word detection.
SayText - Response to user request using TTS engine.
PlayFile - Response with audio file

See also

AudioPlugin
TtsPlugin
SttPlugin

Definition at line 82 of file JokePlugin.py.

9.5.3.2 def plugins.JokePlugin.Humour.joke (*self, entities, raw_text*)

Wrapper for SpeechRecognize event.

Start thread to analyze user input text

Parameters

entities	- Ignored
raw_text	- Text from STT engine

Definition at line 65 of file JokePlugin.py.

9.5.3.3 def plugins.JokePlugin.Humour.joke_done ( ) [static]

Restart user interaction.

After request process restart hot-word detection process

Warning

This function should not be called from outside

Generate events:

RestartInteraction - restart Hot-Word detection.

See also

SttPlugin

Definition at line 103 of file JokePlugin.py.

9.5.4 Member Data Documentation

9.5.4.1 plugins.JokePlugin.Humour._logger [private]

Definition at line 44 of file JokePlugin.py.

9.5.4.2 string plugins.JokePlugin.Humour.description = 'Humour sense' [static]

Short plugin description.

Definition at line 24 of file JokePlugin.py.

9.5.4.3 plugins.JokePlugin.Humour.humour

Definition at line 51 of file JokePlugin.py.

9.5.4.4 string plugins.JokePlugin.Humour.version = '1.0.0.0' [static]

Plugin version.

Definition at line 22 of file JokePlugin.py.

The documentation for this class was generated from the following file:

- plugins/JokePlugin.py

9.6 plugins.SttPlugin.STT Class Reference

Speech to Text abstraction.

Public Member Functions

- def __init__ (self)
  STT and NLP abstraction.
- def __del__ (self)
  Stop module.
- def record_user (self, text)
- def wav_analyze (self, filename)
  Callback function.
- def restart_interaction (self)
  Restart user interaction.
- def speech_data_accepted (self)
  Notify start data process.

Public Attributes

- activation
  Activation Hot-Word list.
- client
  WIT.AI communication instance.

Static Public Attributes

- string version = '1.0.0.2'
  Plugin version.
- string description = 'Interface to WIT STT engine'
  Short plugin description.

Private Member Functions

- def _wav_analyze (self, filename)

  Convert wave file into text.

Private Attributes

- _processing_accepted

  Notify flag to restart Hot-Word detection if no module start processing.
- _gui_recognize_uuid

  Unique id for GUI tray notification.
- _logger

  Logger instance.
- _config

  Configuration instance.
- _temp_folder

  Path to temporary folder.

9.6.1 Detailed Description

Speech to Text abstraction.

Allow interface with STT engine

Version

1.0.0.2

Definition at line 26 of file SttPlugin.py.

9.6.2 Constructor & Destructor Documentation

9.6.2.1 def plugins.SttPlugin.STT.__init__ (*self*)

STT and NLP abstraction.

Create and initialize instance for WIT.ai STT and NLP engine

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Registering on events:

HotWordDetected - Wait until Hot-Word not detected in audio input and send notification.
SpeechAccepted - Wait until module recognize text and start processing SayResponse - System response.
VoiceActivationAccepted - True if Hot-Word detect.

Generate events:

GuiNotification - GUI tray update.

See also

guiPlugin
AudioSubSystem

Definition at line 46 of file SttPlugin.py.

9.6.2.2 def plugins.SttPlugin.STT.__del__ (*self*)

Stop module.

Empty module - required only for compatibility

Definition at line 106 of file SttPlugin.py.

9.6.3 Member Function Documentation

9.6.3.1 def plugins.SttPlugin.STT._wav_analyze (*self, filename*) [private]

Convert wave file into text.

Send file to WIT.AI servers and, receive raw text, and text entities

Parameters

filename	path to wave file that be send to WIT.AI server

Todo Remove silence

Definition at line 144 of file SttPlugin.py.

9.6.3.2 def plugins.SttPlugin.STT.record_user (*self, text*)

Start user voice recording

Warning

This function should not be called from outside

Generate events:

GuiNotification - GUI tray update.
VoiceActivationAccepted - Set to True if Hot-Word accepted by module

Parameters

text	Detected text by local STT engine

See also

guiPlugin
TTS

Definition at line 119 of file SttPlugin.py.

9.6.3.3 def plugins.SttPlugin.STT.restart_interaction (*self*)

Restart user interaction.

After request process restart hot-word detection process

Warning

This function should not be called from outside

Generate events:

RestartInteraction - restart Hot-Word detection.

See also

SttPlugin

Definition at line 186 of file SttPlugin.py.

9.6.3.4 def plugins.SttPlugin.STT.speech_data_accepted (*self*)

Notify start data process.

Notifu to other thread about data processeing

Warning

This function should not be called from outside

Definition at line 193 of file SttPlugin.py.

9.6.3.5 def plugins.SttPlugin.STT.wav_analyze (*self, filename*)

Callback function.

Start thread to communicate with WIT.AI servers

Parameters

filename	Path to wave file with user voice request

Definition at line 137 of file SttPlugin.py.

9.6.4 Member Data Documentation

9.6.4.1 plugins.SttPlugin.STT._config [private]

Configuration instance.

Definition at line 62 of file SttPlugin.py.

9.6.4.2 plugins.SttPlugin.STT._gui_recognize_uuid [private]

Unique id for GUI tray notification.

Definition at line 50 of file SttPlugin.py.

9.6.4.3 plugins.SttPlugin.STT._logger [private]

Logger instance.

Definition at line 54 of file SttPlugin.py.

9.6.4.4 plugins.SttPlugin.STT._processing_accepted [private]

Notify flag to restart Hot-Word detection if no module start processing.

Definition at line 48 of file SttPlugin.py.

9.6.4.5 plugins.SttPlugin.STT._temp_folder [private]

Path to temporary folder.

Definition at line 79 of file SttPlugin.py.

9.6.4.6 plugins.SttPlugin.STT.activation

Activation Hot-Word list.

Definition at line 75 of file SttPlugin.py.

9.6.4.7 plugins.SttPlugin.STT.client

WIT.AI communication instance.

Definition at line 87 of file SttPlugin.py.

9.6.4.8 string plugins.SttPlugin.STT.description = 'Interface to WIT STT engine' [static]

Short plugin description.

Definition at line 30 of file SttPlugin.py.

9.6.4.9 string plugins.SttPlugin.STT.version = '1.0.0.2' [static]

Plugin version.

Definition at line 28 of file SttPlugin.py.

The documentation for this class was generated from the following file:

- plugins/SttPlugin.py

9.7 plugins.TelegramPlugin.TelegramBot Class Reference

Additional user interface.

Public Member Functions

- def __init__ (self)
  Start telegram plugin.
- def __del__ (self)
  Stop module.
- def start (self, bot, update)
  Event wrapper of start command.
- def help (self, bot, update)
  Event wrapper of help command.
- def if_authorized (self, user_id, update)
  Check user authorization.
- def text_handler (self, bot, update)
  Event wrapper user text messages.
- def say_text (self, bot, update)
  Event wrapper of say_text command.
- def get_picture (self, bot, update)
  Event wrapper for get_picture command.
- def get_weather (self, bot, update)
  Event wrapper for get_weather command.
- def unknown (self, bot, update)
  Event wrapper for unknow command.

Public Attributes

- response

  Response messages dictionary.
- api_key

  Telegram API key.

Static Public Attributes

- string version = '1.0.0.0'

  Plugin version.
- string description = 'Telegram bot'

  Short Plugin description.

Private Member Functions

- def _activity_update (self)

  Update GUI tray according user activity.
- def _weather_update (self, custom, description, temp, wind, icon)

  Callback function of weather forecast request.

Private Attributes

- _notify_gui_status

  Unique id for GUI tray icon - message received.
- _camera_gui_status

  Unigue id for GUI tray icon - camera usage.
- _shutdown

  Notify to all thread exit.
- _user_status

  User status dictionary (Login, autorization, dialog state)
- _activity_event

  syncronization event - Allow GUI update
- _logger

  looger instance
- _config

  config file instance
- _camera_angle

  Rotation angle of camera picture.
- _authorization_password

  User authorization password.
- _bot_update

  Bot instance.
- _temp_folder

  Path to temp folder.
- _camera

  Security camera instance.

9.7.1 Detailed Description

Additional user interface.

Communicate with Telegram servers and generate response base on system status

Version

1.0.0.0

Definition at line 38 of file TelegramPlugin.py.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 def plugins.TelegramPlugin.TelegramBot.__init__ (*self*)

Start telegram plugin.

Create and initialize instance start fetching messages. Allow interaction with camera

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Generate events:

GuiNotification - User speech input.
WeatherRequest - Request custom weather forecast.
SayText - Generate voice message using TtsPluign

See also

TTS
WeatherPlugin
AudioSubSystem

Definition at line 55 of file TelegramPlugin.py.

9.7.2.2 def plugins.TelegramPlugin.TelegramBot.__del__ (*self*)

Stop module.

Stop all module thread and sub-programs

Definition at line 160 of file TelegramPlugin.py.

9.7.3 Member Function Documentation

9.7.3.1 def plugins.TelegramPlugin.TelegramBot._activity_update (*self*) [private]

Update GUI tray according user activity.

Receive event flag and set/clear telegram icon in GUI tray

See also

guiPlugin

Definition at line 167 of file TelegramPlugin.py.

9.7.3.2 def plugins.TelegramPlugin.TelegramBot._weather_update (*self, custom, description, temp, wind, icon*) [private]

Callback function of weather forecast request.

Replay to user weather forecast

Parameters

custom	Indification object
description	Short weather descritpion
temp	Temperature
wind	Short Wind description
icon	Path to weather icon - Ignored

Definition at line 331 of file TelegramPlugin.py.

9.7.3.3 def plugins.TelegramPlugin.TelegramBot.get_picture (*self, bot, update*)

Event wrapper for get_picture command.

Receive command and send picture from security camera

Parameters

bot	Bot object
update	Chat update object

Definition at line 288 of file TelegramPlugin.py.

9.7.3.4 def plugins.TelegramPlugin.TelegramBot.get_weather (*self, bot, update*)

Event wrapper for get_weather command.

Receive command and request weather forecast

Parameters

bot	Bot object
update	Chat update object

See also

weatherPlugin

Definition at line 309 of file TelegramPlugin.py.

9.7.3.5 def plugins.TelegramPlugin.TelegramBot.help (*self, bot, update*)

Event wrapper of help command.

Send welcome help text

Parameters

bot	Bot object
update	Chat update object

Definition at line 202 of file TelegramPlugin.py.

9.7.3.6 def plugins.TelegramPlugin.TelegramBot.if_authorized (*self, user_id, update*)

Check user authorization.

Check if user pass authorization process

Returns

True/False according user status

Definition at line 209 of file TelegramPlugin.py.

9.7.3.7 def plugins.TelegramPlugin.TelegramBot.say_text (*self, bot, update*)

Event wrapper of say_text command.

Receive command and update user dialog status

Parameters

bot	Bot object
update	Chat update object

Definition at line 277 of file TelegramPlugin.py.

9.7.3.8 def plugins.TelegramPlugin.TelegramBot.start (*self, bot, update*)

Event wrapper of start command.

Send welcome text and create/reset user instance

Parameters

bot	Bot object
update	Chat update object

Definition at line 179 of file TelegramPlugin.py.

9.7.3.9 def plugins.TelegramPlugin.TelegramBot.text_handler (*self, bot, update*)

Event wrapper user text messages.

Update user dialog status

Parameters

bot	Bot object
update	Chat update object

Generate events:

WeatherRequest - Weather forecast request SayText - Generate speech using Tts engine

See also

weatherPlugin see TTS

Definition at line 228 of file TelegramPlugin.py.

9.7.3.10 def plugins.TelegramPlugin.TelegramBot.unknown (*self, bot, update*)

Event wrapper for unknow command.

Receive command and response to user

Parameters

bot	Bot object
update	Chat update object

Definition at line 320 of file TelegramPlugin.py.

9.7.4 Member Data Documentation

9.7.4.1 plugins.TelegramPlugin.TelegramBot._activity_event [private]

syncronization event - Allow GUI update

Definition at line 65 of file TelegramPlugin.py.

9.7.4.2 plugins.TelegramPlugin.TelegramBot._authorization_password [private]

User authorization password.

Definition at line 100 of file TelegramPlugin.py.

9.7.4.3 plugins.TelegramPlugin.TelegramBot._bot_update [private]

Bot instance.

Definition at line 112 of file TelegramPlugin.py.

9.7.4.4 plugins.TelegramPlugin.TelegramBot._camera [private]

Security camera instance.

Definition at line 145 of file TelegramPlugin.py.

9.7.4.5 plugins.TelegramPlugin.TelegramBot._camera_angle [private]

Rotation angle of camera picture.

Definition at line 83 of file TelegramPlugin.py.

9.7.4.6 plugins.TelegramPlugin.TelegramBot._camera_gui_status [private]

Unigue id for GUI tray icon - camera usage.

Definition at line 59 of file TelegramPlugin.py.

9.7.4.7 plugins.TelegramPlugin.TelegramBot._config [private]

config file instance

Definition at line 77 of file TelegramPlugin.py.

9.7.4.8 plugins.TelegramPlugin.TelegramBot._logger [private]

looger instance

Definition at line 69 of file TelegramPlugin.py.

9.7.4.9 plugins.TelegramPlugin.TelegramBot._notify_gui_status [private]

Unique id for GUI tray icon - message received.

Definition at line 57 of file TelegramPlugin.py.

9.7.4.10 plugins.TelegramPlugin.TelegramBot._shutdown [private]

Notify to all thread exit.

Definition at line 61 of file TelegramPlugin.py.

9.7.4.11 plugins.TelegramPlugin.TelegramBot._temp_folder [private]

Path to temp folder.

Definition at line 136 of file TelegramPlugin.py.

9.7.4.12 plugins.TelegramPlugin.TelegramBot._user_status [private]

User status dictionary (Login, autorization, dialog state)

Definition at line 63 of file TelegramPlugin.py.

9.7.4.13 plugins.TelegramPlugin.TelegramBot.api_key

Telegram API key.

Definition at line 98 of file TelegramPlugin.py.

9.7.4.14 string plugins.TelegramPlugin.TelegramBot.description = 'Telegram bot' [static]

Short Plugin description.

Definition at line 42 of file TelegramPlugin.py.

9.7.4.15 plugins.TelegramPlugin.TelegramBot.response

Response messages dictionary.

Definition at line 87 of file TelegramPlugin.py.

9.7.4.16 string plugins.TelegramPlugin.TelegramBot.version = '1.0.0.0' [static]

Plugin version.

Definition at line 40 of file TelegramPlugin.py.

The documentation for this class was generated from the following file:

- plugins/TelegramPlugin.py

9.8 plugins.TtsPlugin.TTS Class Reference

Test To Speech abstraction.

Public Member Functions

- def __init__ (self)

  Create TTS interface instance.
- def __del__ (self)

  Stop module.
- def text2wav (self, sender, text, callback=None)

  Event SayText wrapper details Receive text and convert it to wave file.
- def response (self, response)

  SayResponce event wrapper details Fetch system response from config file and convert it to speech.

Public Attributes

- tts_command

  TTS engine inialize command.

Static Public Attributes

- string version = '1.0.0.1'

  Plugin version.
- string description = 'Python wrapper for TTS - festeval'

  Short plugin description.

Private Member Functions

- def _synthesize (self, text_file, wave_file, text)

  TTS engine wrapper details Convert text to wave file.

Private Attributes

- _gui_synthesis_status_uuid
  Unique id for GUI tray.
- _logger
  looger instance
- _config
  config file instance
- _cache_folder
  Temporary folder to store generated wave files - cache.
- _cached_text
  cached text list
- _use_cache
  configuration - True if cache enabled
- _cache_size
  Maximum cache size.
- _clear_on_exit
  Configuration if True cached will be cleared on exit.

9.8.1 Detailed Description

Test To Speech abstraction.

Allow interface with TTS engine

Version

1.0.0.1

Definition at line 23 of file TtsPlugin.py.

9.8.2 Constructor & Destructor Documentation

9.8.2.1 def plugins.TtsPlugin.TTS.__init__ (*self*)

Create TTS interface instance.

Create and initialize instance, initialize festival engine

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Registering on events:

SayResponse - System define responses.
SayTest - Convert text int speech.

Generate events:

GuiNotification - GUI tray update.
SpeechSynthesize - True while TTS engine is running.

See also

guiPlugin

Definition at line 41 of file TtsPlugin.py.

9.8.2.2 def plugins.TtsPlugin.TTS.__del__ (*self*)

Stop module.

Stop all module thread and sub-programs

Definition at line 105 of file TtsPlugin.py.

9.8.3 Member Function Documentation

9.8.3.1 def plugins.TtsPlugin.TTS._synthesize (*self, text_file, wave_file, text*) [private]

TTS engine wrapper details Convert text to wave file.

Parameters

text_file	Path to text file
wave_file	Path where store wave file
text	Text to be converted

Definition at line 167 of file TtsPlugin.py.

9.8.3.2 def plugins.TtsPlugin.TTS.response (*self, response*)

SayResponce event wrapper details Fetch system response from config file and convert it to speech.

Parameters

response	System response

Definition at line 195 of file TtsPlugin.py.

9.8.3.3 def plugins.TtsPlugin.TTS.text2wav (*self, sender, text, callback* = None)

Event SayText wrapper details Receive text and convert it to wave file.

Parameters

sender	Message origin
text	Text to convert
callback	Callback function.Optional.Default - None

Definition at line 129 of file TtsPlugin.py.

9.8.4 Member Data Documentation

9.8.4.1 plugins.TtsPlugin.TTS._cache_folder [private]

Temporary folder to store generated wave files - cache.

Definition at line 62 of file TtsPlugin.py.

9.8.4.2 plugins.TtsPlugin.TTS._cache_size [private]

Maximum cache size.

Definition at line 82 of file TtsPlugin.py.

9.8.4.3 plugins.TtsPlugin.TTS._cached_text [private]

cached text list

Definition at line 77 of file TtsPlugin.py.

9.8.4.4 plugins.TtsPlugin.TTS._clear_on_exit [private]

Configuration if True cached will be cleared on exit.

Definition at line 84 of file TtsPlugin.py.

9.8.4.5 plugins.TtsPlugin.TTS._config [private]

config file instance

Definition at line 55 of file TtsPlugin.py.

9.8.4.6 plugins.TtsPlugin.TTS._gui_synthesis_status_uuid [private]

Unique id for GUI tray.

Definition at line 43 of file TtsPlugin.py.

9.8.4.7 plugins.TtsPlugin.TTS._logger [private]

looger instance

Definition at line 47 of file TtsPlugin.py.

9.8.4.8 plugins.TtsPlugin.TTS._use_cache [private]

configuration - True if cache enabled

Definition at line 80 of file TtsPlugin.py.

9.8.4.9 string plugins.TtsPlugin.TTS.description = 'Python wrapper for TTS - festeval' [static]

Short plugin description.

Definition at line 27 of file TtsPlugin.py.

9.8.4.10 plugins.TtsPlugin.TTS.tts_command

TTS engine inialize command.

Definition at line 71 of file TtsPlugin.py.

9.8.4.11 string plugins.TtsPlugin.TTS.version = '1.0.0.1' [static]

Plugin version.

Definition at line 25 of file TtsPlugin.py.

The documentation for this class was generated from the following file:

- plugins/TtsPlugin.py

9.9 plugins.WeatherPlugin.Weather Class Reference

Interaction with OPenWeatherMap.

Public Member Functions

- def __init__ (self)
  Create Weather interface instance.
- def __del__ (self)
  Stop module.
- def periodic_update (self)
  Periodic update thread.
- def user_request (self, entities, raw_text)
  Event wrapper for SpeechRecognize.
- def custom_request (self, callback, custom_object=None, request_time=None, request_city=None)
  Wrapper for WeatherRequest event.

Static Public Member Functions

- def sythsys_complete ()

  Restart user interaction.

Public Attributes

- api_key

  Access API key to OPenWeatherMap.

Static Public Attributes

- string version = '1.0.0.0'

  Plugin version.
- string description = 'Weather module'

  Short plugin description.

Private Member Functions

- def _user_request (self, entities)

  Weather fetch thread.

Private Attributes

- _shutdown

  Shutdown event - notify to thread exit.
- _weather_data

  weather data cache
- _gui_status

  Unique id to GUI tray.
- _logger

  looger instance
- _config

  Configuration file instance.
- _main_city

  Default city name.
- _update_interval

  Periodic update interval.
- _base_url

  Base URL for data fetching.
- _icon_url

  Base URL for weather icon fetching.
- _temp_folder

  Cache folder.
- _units

  Default units.
- _dump_json

  Configuration - If True save raw JSON in cache folder.

9.9.1 Detailed Description

Interaction with OPenWeatherMap.

Simple weather data retriever from OpenWeatherMap site

Version

1.0.0.0

Definition at line 26 of file WeatherPlugin.py.

9.9.2 Constructor & Destructor Documentation

9.9.2.1 def plugins.WeatherPlugin.Weather.__init__ (*self*)

Create Weather interface instance.

Create and initialize instance, initialize weather fetching

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Registering on events:

SpeechRecognize - User requests.
WeatherRequest - Internal weather forecast request.
RestartInteraction - Accept text for process.
SpeechAccepted - Restart User interaction.

Generate events:

GuiNotification - GUI tray update.
SpeechSynthesize - True while TTS engine is running.

See also

guiPlugin

Definition at line 46 of file WeatherPlugin.py.

9.9.2.2 def plugins.WeatherPlugin.Weather.__del__ (*self*)

Stop module.

Stop all module thread and sub-programs

Definition at line 127 of file WeatherPlugin.py.

9.9.3 Member Function Documentation

9.9.3.1 def plugins.WeatherPlugin.Weather._user_request (*self, entities*) [private]

Weather fetch thread.

Fetch weather data

Parameters

entities	- Dictionary with text parts

Generate events:

GuiNotification - GUI tray update.
SayText - Response.

Definition at line 187 of file WeatherPlugin.py.

9.9.3.2 def plugins.WeatherPlugin.Weather.custom_request (*self, callback, custom_object* = None, *request_time* = None, *request_city* = None)

Wrapper for WeatherRequest event.

Allow to other modules request weather data

Parameters

callback	Callback function
custom_object	Id object. Optional. Default - None
request_time	Unix time for forecast. Optional. Default - now
request_city	Optional.Default - default city

Definition at line 239 of file WeatherPlugin.py.

9.9.3.3 def plugins.WeatherPlugin.Weather.periodic_update (*self*)

Periodic update thread.

Periodic weather retrieve

Generate events:

WeatherUpdate - Weather update.
GuiNotification - GUI tray update.

See also

guiPlugin

Definition at line 138 of file WeatherPlugin.py.

9.9.3.4 def plugins.WeatherPlugin.Weather.sythsys_complete ( ) [static]

Restart user interaction.

After request process restart hot-word detection process

Warning

This function should not be called from outside

Generate events:

RestartInteraction - restart Hot-Word detection.

See also

SttPlug

Definition at line 283 of file WeatherPlugin.py.

9.9.3.5 def plugins.WeatherPlugin.Weather.user_request (*self, entities, raw_text*)

Event wrapper for SpeechRecognize.

Check if user request for weather forecast

Parameters

entities	- Dictionary with text parts
raw_text	- Ignored

Generate events:

SpeechAccepted - Notify that nnoduel start request process.
GuiNotification - GUI tray update.
SayText - Response.

Definition at line 171 of file WeatherPlugin.py.

9.9.4 Member Data Documentation

9.9.4.1 plugins.WeatherPlugin.Weather._base_url [private]

Base URL for data fetching.

Definition at line 84 of file WeatherPlugin.py.

9.9.4.2 plugins.WeatherPlugin.Weather._config [private]

Configuration file instance.

Definition at line 63 of file WeatherPlugin.py.

9.9.4.3 plugins.WeatherPlugin.Weather._dump_json [private]

Configuration - If True save raw JSON in cache folder.

Definition at line 98 of file WeatherPlugin.py.

9.9.4.4 plugins.WeatherPlugin.Weather._gui_status [private]

Unique id to GUI tray.

Definition at line 52 of file WeatherPlugin.py.

9.9.4.5 plugins.WeatherPlugin.Weather._icon_url [private]

Base URL for weather icon fetching.

Definition at line 90 of file WeatherPlugin.py.

9.9.4.6 plugins.WeatherPlugin.Weather._logger [private]

looger instance

Definition at line 55 of file WeatherPlugin.py.

9.9.4.7 plugins.WeatherPlugin.Weather._main_city [private]

Default city name.

Definition at line 78 of file WeatherPlugin.py.

9.9.4.8 plugins.WeatherPlugin.Weather._shutdown [private]

Shutdown event - notify to thread exit.

Definition at line 48 of file WeatherPlugin.py.

9.9.4.9 plugins.WeatherPlugin.Weather._temp_folder [private]

Cache folder.

Definition at line 94 of file WeatherPlugin.py.

9.9.4.10 plugins.WeatherPlugin.Weather._units [private]

Default units.

Definition at line 96 of file WeatherPlugin.py.

9.9.4.11 plugins.WeatherPlugin.Weather._update_interval [private]

Periodic update interval.

Definition at line 80 of file WeatherPlugin.py.

9.9.4.12 plugins.WeatherPlugin.Weather._weather_data [private]

weather data cache

Definition at line 50 of file WeatherPlugin.py.

9.9.4.13 plugins.WeatherPlugin.Weather.api_key

Access API key to OPenWeatherMap.

Definition at line 69 of file WeatherPlugin.py.

9.9.4.14 string plugins.WeatherPlugin.Weather.description = 'Weather module' [static]

Short plugin description.

Definition at line 30 of file WeatherPlugin.py.

9.9.4.15 string plugins.WeatherPlugin.Weather.version = '1.0.0.0' [static]

Plugin version.

Definition at line 28 of file WeatherPlugin.py.

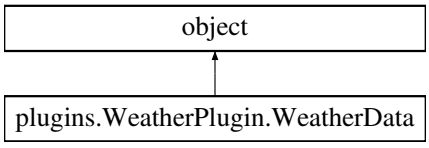
The documentation for this class was generated from the following file:

- plugins/WeatherPlugin.py

9.10 plugins.WeatherPlugin.WeatherData Class Reference

Hold forecast data.

Inheritance diagram for plugins.WeatherPlugin.WeatherData:



Public Member Functions

- def __init__ (self, api_key, logger)

  Create Weather interface instance.
- def update (self)

  Start weather fetching.
- def base_url (self)

  Base URL.
- def base_url (self, url)
- def icon_url (self)

  Base icon URL.
- def icon_url (self, url)
- def icon_folder (self)

  Icon folder path.
- def icon_folder (self, folder)
- def icon (self)

  Icon full path.
- def city_name (self)

  City name - updated after fetching.
- def city_name (self, name)
- def auto_update (self)

  if auto-update enable
- def auto_update (self, state)
- def request_time (self)
- def request_time (self, req_time)
- def units (self)

  Units.
- def units (self, unit)
- def temp (self)

  Temperture.
- def temp_max (self)

  Temperature maximum.
- def temp_min (self)
- def pressure (self)

  Temperature minimum.
- def humidity (self)

  Humidity.
- def wind_speed (self)

  wind speed in selected units
- def wind_direction (self)

  Wind angle.
- def rain (self)

  If rain.
- def show (self)
- def clouds (self)

  Cloud.
- def measure_time (self)

  Measure time - Unix time.
- def title (self)

  One line weather description.
- def short_description (self)

  Short weather description.

- def wind_description (self)

  Textual wind description.
- def description (self)

  Long weather description.

Public Attributes

- weather_data

  Weather data dictionary.
- units

Private Attributes

- _api_key

  Acces token fro OpenWeatherMap Api.
- _logger

  Logger instance.
- _auto_update

  Configuration if True update will start after all requred data receive.
- _city_name

  City name.
- _requested_time

  Request time for forecast - Unix time.
- _units

  Units.
- _base_url

  Base API URL.
- _icon_url

  Base icon fetch URL.
- _icon_folder

  Icon store folder.

9.10.1 Detailed Description

Hold forecast data.

Allow simple data fetching

Version

1.0.0.0

Definition at line 291 of file WeatherPlugin.py.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 def plugins.WeatherPlugin.WeatherData.__init__ (*self, api_key, logger*)

Create Weather interface instance.

Create and initialize instance, initialize weather fetching

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Parameters

api_key	Acces token for OpenWeatherMap
logger	Logger instance

Definition at line 297 of file WeatherPlugin.py.

9.10.3 Member Function Documentation

9.10.3.1 def plugins.WeatherPlugin.WeatherData.auto_update (*self*)

if auto-update enable

Definition at line 482 of file WeatherPlugin.py.

9.10.3.2 def plugins.WeatherPlugin.WeatherData.auto_update (*self, state*)

Definition at line 486 of file WeatherPlugin.py.

9.10.3.3 def plugins.WeatherPlugin.WeatherData.base_url (*self*)

Base URL.

Definition at line 432 of file WeatherPlugin.py.

9.10.3.4 def plugins.WeatherPlugin.WeatherData.base_url (*self, url*)

Definition at line 436 of file WeatherPlugin.py.

9.10.3.5 def plugins.WeatherPlugin.WeatherData.city_name (*self*)

City name - updated after fetching.

Definition at line 470 of file WeatherPlugin.py.

9.10.3.6 def plugins.WeatherPlugin.WeatherData.city_name (*self, name*)

Definition at line 474 of file WeatherPlugin.py.

9.10.3.7 def plugins.WeatherPlugin.WeatherData.clouds (*self*)

Cloud.

Precondition

Valid only after update

Definition at line 573 of file WeatherPlugin.py.

9.10.3.8 def plugins.WeatherPlugin.WeatherData.description (*self*)

Long weather description.

Precondition

Valid only after update

Definition at line 666 of file WeatherPlugin.py.

9.10.3.9 def plugins.WeatherPlugin.WeatherData.humidity (*self*)

Humidity.

Precondition

Valid only after update

Definition at line 543 of file WeatherPlugin.py.

9.10.3.10 def plugins.WeatherPlugin.WeatherData.icon (*self*)

Icon full path.

Definition at line 465 of file WeatherPlugin.py.

9.10.3.11 def plugins.WeatherPlugin.WeatherData.icon_folder (*self*)

Icon folder path.

Definition at line 452 of file WeatherPlugin.py.

9.10.3.12 def plugins.WeatherPlugin.WeatherData.icon_folder (*self, folder*)

Definition at line 456 of file WeatherPlugin.py.

9.10.3.13 def plugins.WeatherPlugin.WeatherData.icon_url (*self*)

Base icon URL.

Definition at line 442 of file WeatherPlugin.py.

9.10.3.14 def plugins.WeatherPlugin.WeatherData.icon_url (*self, url*)

Definition at line 446 of file WeatherPlugin.py.

9.10.3.15 def plugins.WeatherPlugin.WeatherData.measure_time (*self*)

Measure time - Unix time.

Precondition

Valid only after update

Definition at line 579 of file WeatherPlugin.py.

9.10.3.16 def plugins.WeatherPlugin.WeatherData.pressure (*self*)

Temperature minimum.

Precondition

Valid only after update

Definition at line 537 of file WeatherPlugin.py.

9.10.3.17 def plugins.WeatherPlugin.WeatherData.rain (*self*)

If rain.

Precondition

Valid only after update

Definition at line 561 of file WeatherPlugin.py.

9.10.3.18 def plugins.WeatherPlugin.WeatherData.request_time (*self*)

Definition at line 491 of file WeatherPlugin.py.

9.10.3.19 def plugins.WeatherPlugin.WeatherData.request_time (*self, req_time*)

Definition at line 495 of file WeatherPlugin.py.

9.10.3.20 def plugins.WeatherPlugin.WeatherData.short_description (*self*)

Short weather description.

Precondition

Valid only after update

Definition at line 591 of file WeatherPlugin.py.

9.10.3.21 def plugins.WeatherPlugin.WeatherData.show (*self*)

Precondition

Valid only after update

Definition at line 567 of file WeatherPlugin.py.

9.10.3.22 def plugins.WeatherPlugin.WeatherData.temp (*self*)

Temperture.

Precondition

Valid only after update

Definition at line 521 of file WeatherPlugin.py.

9.10.3.23 def plugins.WeatherPlugin.WeatherData.temp_max (*self*)

Temperature maximum.

Precondition

Valid only after update

Definition at line 527 of file WeatherPlugin.py.

9.10.3.24 def plugins.WeatherPlugin.WeatherData.temp_min (*self*)

Definition at line 531 of file WeatherPlugin.py.

9.10.3.25 def plugins.WeatherPlugin.WeatherData.title (*self*)

One line weather description.

Precondition

Valid only after update

Definition at line 585 of file WeatherPlugin.py.

9.10.3.26 def plugins.WeatherPlugin.WeatherData.units (*self*)

Units.

Definition at line 504 of file WeatherPlugin.py.

9.10.3.27 def plugins.WeatherPlugin.WeatherData.units (*self, unit*)

Definition at line 511 of file WeatherPlugin.py.

9.10.3.28 def plugins.WeatherPlugin.WeatherData.update (*self*)

Start weather fetching.

Connect to OpenWeatherMap site and download weather data

Definition at line 336 of file WeatherPlugin.py.

9.10.3.29 def plugins.WeatherPlugin.WeatherData.wind_description (*self*)

Textual wind description.

Precondition

Valid only after update

Definition at line 597 of file WeatherPlugin.py.

9.10.3.30 def plugins.WeatherPlugin.WeatherData.wind_direction (*self*)

Wind angle.

Precondition

Valid only after update

Definition at line 555 of file WeatherPlugin.py.

9.10.3.31 def plugins.WeatherPlugin.WeatherData.wind_speed (*self*)

wind speed in selected units

Precondition

Valid only after update

Definition at line 549 of file WeatherPlugin.py.

9.10.4 Member Data Documentation

9.10.4.1 plugins.WeatherPlugin.WeatherData._api_key [private]

Acces token fro OpenWeatherMap Api.

Definition at line 299 of file WeatherPlugin.py.

9.10.4.2 plugins.WeatherPlugin.WeatherData._auto_update [private]

Configuration if True update will start after all requred data receive.

Definition at line 303 of file WeatherPlugin.py.

9.10.4.3 plugins.WeatherPlugin.WeatherData._base_url [private]

Base API URL.

Definition at line 328 of file WeatherPlugin.py.

9.10.4.4 plugins.WeatherPlugin.WeatherData._city_name [private]

City name.

Definition at line 305 of file WeatherPlugin.py.

9.10.4.5 plugins.WeatherPlugin.WeatherData._icon_folder [private]

Icon store folder.

Definition at line 332 of file WeatherPlugin.py.

9.10.4.6 plugins.WeatherPlugin.WeatherData._icon_url [private]

Base icon fetch URL.

Definition at line 330 of file WeatherPlugin.py.

9.10.4.7 plugins.WeatherPlugin.WeatherData._logger [private]

Logger instance.

Definition at line 301 of file WeatherPlugin.py.

9.10.4.8 plugins.WeatherPlugin.WeatherData._requested_time [private]

Request time for forecast - Unix time.

Definition at line 307 of file WeatherPlugin.py.

9.10.4.9 plugins.WeatherPlugin.WeatherData._units [private]

Units.

Definition at line 309 of file WeatherPlugin.py.

9.10.4.10 plugins.WeatherPlugin.WeatherData.units

Definition at line 606 of file WeatherPlugin.py.

9.10.4.11 plugins.WeatherPlugin.WeatherData.weather_data

Weather data dictionary.

Definition at line 311 of file WeatherPlugin.py.

The documentation for this class was generated from the following file:

- plugins/WeatherPlugin.py

9.11 plugins.EmailPlugin.ZohoEmail Class Reference

Email plugin.

Public Member Functions

- def __init__ (self)
  Create Email interface instance.
- def __del__ (self)
  Stop module.
- def user_request (self, entities, raw_text)
  SpeechRecognize event wrapper.

Static Public Member Functions

- def sythsys_complete ()

  Restart user interaction.

Public Attributes

- api_user

  User name for email server.
- api_key

  Password for Email server.

Static Public Attributes

- string version = '1.0.0.0'

  Plugin version.
- string description = 'Zoho email client module'

  Plugin description.

Private Member Functions

- def _connect (self)

  Connect to email server.
- def _periodic_update (self)

  Wait to new emails.
- def _user_request (self, entities)

  Analyze user request.

Private Attributes

- _gui_status

  Unique id for try icon.
- _shutdown

  Shutdown event - signal to all thread exit.
- _message_list

  Message list.
- _message_list_token

  Message list synchronization object - allow thread safe update.
- _logger

  logger instance
- _config

  configuration file intance
- _update_interval

  Email list refresh interval.
- _server_url

  Zoho email sever URL.
- _server_port

  Zoho email server communication protocol.

9.11.1 Detailed Description

Email plugin.

Communication with Zoho (https://www.zoho.com/) email and retrieve email using POP3 protocol

Version

1.0.0.0

Definition at line 26 of file EmailPlugin.py.

9.11.2 Constructor & Destructor Documentation

9.11.2.1 def plugins.EmailPlugin.ZohoEmail.__init__ (*self*)

Create Email interface instance.

Create and initialize instance

Exceptions

ImportError	Configuration or IO system error - Module will be unloaded.

Registering on events:

SpeechRecognize - User speech input.

Generate events:

GuiNotification - GUI tray update.

See also

guiPlugin

Definition at line 43 of file EmailPlugin.py.

9.11.2.2 def plugins.EmailPlugin.ZohoEmail.__del__ (*self*)

Stop module.

Stop all module thread and sub-programs

Definition at line 110 of file EmailPlugin.py.

9.11.3 Member Function Documentation

9.11.3.1 def plugins.EmailPlugin.ZohoEmail._connect (*self*) [private]

Connect to email server.

Connect and login to email server

Warning

This function should not be called from outside

Generate events:

GuiNotification - GUI tray update.

See also

guiPlugin

Definition at line 122 of file EmailPlugin.py.

9.11.3.2 def plugins.EmailPlugin.ZohoEmail._periodic_update (*self*) [private]

Wait to new emails.

Refresh emails on email server

Warning

This function should not be called from outside

Generate events:

GuiNotification - GUI tray update.
SpeechAccepted - Notify that module start process user request.
RestartInteraction - Restart Hot-Word detection.
SayText - Response to user request using TTS engine.

See also

guiPlugin
AudioSubSystem
TtsPlugin

Definition at line 158 of file EmailPlugin.py.

9.11.3.3 def plugins.EmailPlugin.ZohoEmail._user_request (*self, entities*) [private]

Analyze user request.

Extract data from user request and generate response

Warning

This function should not be called from outside

Generate events:

GuiNotification - GUI tray update.
SayText - Response to user request using TTS engine.

Parameters

entities	dictionary Speech entities

See also

guiPlugin
TtsPlugin

Definition at line 242 of file EmailPlugin.py.

9.11.3.4 def plugins.EmailPlugin.ZohoEmail.sythsys_complete ( ) [static]

Restart user interaction.

After request process restart hot-word detection process

Warning

This function should not be called from outside

Generate events:

RestartInteraction - restart Hot-Word detection.

See also

SttPlugin

Definition at line 289 of file EmailPlugin.py.

9.11.3.5 def plugins.EmailPlugin.ZohoEmail.user_request (*self, entities, raw_text*)

SpeechRecognize event wrapper.

If user request contain email entity with high confidence begin request process

Warning

This function should not be called from outside

Generate events:

GuiNotification - GUI tray update.

Parameters

entities	dictionary Speech entities
raw_text	string Ignored

See also

guiPlugin
TtsPlugin

Definition at line 223 of file EmailPlugin.py.

9.11.4 Member Data Documentation

9.11.4.1 plugins.EmailPlugin.ZohoEmail._config [private]

configuration file intance

Definition at line 62 of file EmailPlugin.py.

9.11.4.2 plugins.EmailPlugin.ZohoEmail._gui_status [private]

Unique id for try icon.

Definition at line 45 of file EmailPlugin.py.

9.11.4.3 plugins.EmailPlugin.ZohoEmail._logger [private]

logger instance

Definition at line 54 of file EmailPlugin.py.

9.11.4.4 plugins.EmailPlugin.ZohoEmail._message_list [private]

Message list.

Definition at line 49 of file EmailPlugin.py.

9.11.4.5 plugins.EmailPlugin.ZohoEmail._message_list_token [private]

Message list synchronization object - allow thread safe update.

Definition at line 51 of file EmailPlugin.py.

9.11.4.6 plugins.EmailPlugin.ZohoEmail._server_port [private]

Zoho email server communication protocol.

Definition at line 84 of file EmailPlugin.py.

9.11.4.7 plugins.EmailPlugin.ZohoEmail._server_url [private]

Zoho email sever URL.

Definition at line 82 of file EmailPlugin.py.

9.11.4.8 plugins.EmailPlugin.ZohoEmail._shutdown [private]

Shutdown event - signal to all thread exit.

Definition at line 47 of file EmailPlugin.py.

9.11.4.9 plugins.EmailPlugin.ZohoEmail._update_interval [private]

Email list refresh interval.

Definition at line 78 of file EmailPlugin.py.

9.11.4.10 plugins.EmailPlugin.ZohoEmail.api_key

Password for Email server.

Definition at line 69 of file EmailPlugin.py.

9.11.4.11 plugins.EmailPlugin.ZohoEmail.api_user

User name for email server.

Definition at line 66 of file EmailPlugin.py.

9.11.4.12 string plugins.EmailPlugin.ZohoEmail.description = 'Zoho email client module' [static]

Plugin description.

Definition at line 30 of file EmailPlugin.py.

9.11.4.13 string plugins.EmailPlugin.ZohoEmail.version = '1.0.0.0' [static]

Plugin version.

Definition at line 28 of file EmailPlugin.py.

The documentation for this class was generated from the following file:

- plugins/EmailPlugin.py

Chapter 10

File Documentation

10.1 Aria.py File Reference

Main file.

Namespaces

- Aria

Functions

- def Aria.clean_exit ()

  Cleanup function.
- def Aria.aria_start ()

  Startup function.
- def Aria.emergency_shutdown ()

Variables

- Aria.shutdown_flag = threading.Event()

10.1.1 Detailed Description

Main file.

Definition in file Aria.py.

10.2 Aria.py

```
00001 #!/usr/bin/python
00002 ## @file
00003 ## @brief Main file
00004 ## @mainpage Aria - Digital Assistant - Code documentation
00005 #
00006 ## @section Dependencies
00007 # pydispatch - http://pydispatcher.sourceforge.net/\n
00008 # pydev - http://www.pydev.org/\n
00009 # dateutil - https://dateutil.readthedocs.io/en/stable/\n
00010 # keyring - https://pypi.org/project/keyring/\n
00011 # facebook SDK - https://github.com/mobolic/facebook-sdk\n
00012 # telegram - https://github.com/python-telegram-bot/python-telegram-bot\n
00013 # emoji - https://github.com/carpedm20/emoji/\n
00014 #
00015 ## @par Configuration file
00016 ## @verbinclude ./configuration/main.conf
00017 #
00018 ## @par Logger configuration
00019 ## @verbinclude ./main.logger
00020 #
00021
00022 import ConfigParser
00023 import atexit
00024 import inspect
00025 import logging
00026 import logging.config
00027 import os
00028 import sys
00029 from time import sleep
00030 import threading
00031
00032 import pydevd
00033 from pydispatch import dispatcher
00034
00035
00036 @atexit.register
00037 ## @fn def clean_exit():
00038 ## @brief Cleanup function
00039 ## @details Close all debug/log session and perform clean exit
00040 def clean_exit():
00041     print 'Main thread termination'
00042     try:
00043         if pydevd.GetGlobalDebugger() is not None:
00044             print '################################################'
00045             print '########   Remote debug session ended  ##########'
00046             print '################################################'
00047             pydevd.stoptrace()
00048     except:
00049         pass
00050
00051
00052 ## @fn def aria_start():
00053 ## @brief  Startup function
00054 ## @details Start all services and load runners
00055 ## @note Load logger configuration, and run settings
00056 ## @see https://docs.python.org/2/library/logging.html
00057 def aria_start():
00058     # Start reading _config file
00059     _config = ConfigParser.SafeConfigParser(allow_no_value=True)
00060     _config.read('./configuration/main.conf')
00061     # Setting up debug session
00062     try:
00063         # Parse configuration options
00064         if _config.getboolean('Debug', 'Debug'):
00065             print 'Trying to start debug session.'
00066             _debug_host = _config.get('Debug', 'host').strip()
00067             _debug_port = _config.getint('Debug', 'port')
00068             print 'Remote host - %s  on port %i' % (_debug_host, _debug_port)
00069             pydevd.settrace(_debug_host, port=_debug_port, stdoutToServer=True, stderrToServer=True,
      suspend=False)
00070             print '################################################'
00071             print '########  Remote debug session started ##########'
00072             print '################################################'
00073         else:
00074             print 'Start in normal mode.'
00075     except ConfigParser.NoSectionError:
00076         print 'No debug section found.Starting in normal mode'
00077         print 'Missing debug parameters.Please refer manual.Starting in normal mode'
00078     # setting up logger
00079     try:
00080         logging.config.fileConfig('main.logger')
00081         _logger = logging.getLogger('root')
00082     except ConfigParser.NoSectionError as e:
00083         print 'Fatal error  - fail to set _logger.Error: %s ' % e.message
```

```
00084         exit(-1)
00085     _logger.debug('Logger started')
00086     # Loading modules
00087     # Storing loaded modules
00088     active_modules = list()
00089     try:
00090         # Search all files in plugin folder
00091         plugin_dir = _config.get('Modules', 'Path').strip()
00092         _logger.info('Searching modules in: %s' % plugin_dir)
00093     except IOError:
00094         # Incorrect folder - Switching to default
00095         _logger.info('Error getting plugin dir using default - plugins')
00096         plugin_dir = 'plugins'
00097     try:
00098         # Create list of disables modules and classes
00099         disable_modules = _config.get('Modules', 'Disabled')
00100         disable_modules = disable_modules.strip().split(',')
00101         disable_classes = _config.get('Classes', 'Disabled')
00102         disable_classes = disable_classes.strip().split(',')
00103     except ConfigParser as e:
00104         _logger.fatal('Fail to read config file with error %s' % e)
00105         exit(-1)
00106     _logger.info('Disabled modules : %s' % disable_modules)
00107     _logger.info('Disabled classes : %s' % disable_classes)
00108 
00109     if not os.path.exists(plugin_dir):
00110         _logger.critical('Plugins folder not exist')
00111         exit(-1)
00112     # Searching .py files in folder 'plugins'
00113     for fname in os.listdir(plugin_dir):
00114         # Look only for py files
00115         if (fname.endswith('.py')) and ('plugin' in fname.lower()):
00116             # Cut .py from path
00117             module_name = fname[: -3]
00118             # Skip base,__init__  and disabled files
00119             if module_name != 'base' and module_name != '__init__' and not (module_name in disable_modules)
    :
00120                 _logger.info('Found module %s' % module_name)
00121                 # Load module and add it to list of loaded modules
00122                 package_obj = __import__(plugin_dir + '.' + module_name)
00123                 active_modules.append(module_name)
00124             else:
00125                 _logger.info('Skipping %s' % fname)
00126 
00127     # Retrieving modules
00128     _loaded_modules = []
00129     for modulename in active_modules:
00130         module_obj = getattr(package_obj, modulename)
00131         # Looking for classes in file
00132         for elem in dir(module_obj):
00133             obj = getattr(module_obj, elem)
00134             # If this a class ?
00135             if inspect.isclass(obj):
00136                 if elem in disable_classes:
00137                     _logger.info('Skipping %s' % obj)
00138                     continue
00139                 # Creating object
00140                 try:
00141                     _logger.info('Loading module %s from %s' % (elem, modulename))
00142                     try:
00143                         _module = obj()
00144                     except (ImportError, TypeError) as e:
00145                         # Some error while creating module instance
00146                         _logger.fatal('Incorrect module. Error %s' % e)
00147                 except ImportWarning:
00148                     _logger.warning('Failed to load %s from %s' % (elem, modulename))
00149                     del _module
00150                     pass
00151                 else:
00152                     # Store module instance
00153                     _loaded_modules.append(_module)
00154                     _logger.info('Module %s (version: %s) loaded' % (elem, _module.version))
00155     sleep(5)  # Init time
00156     _logger.info('All modules loaded')
00157     # Create event for shutdown of main thread
00158     dispatcher.connect(emergency_shutdown, signal='EmergencyShutdown')
00159     dispatcher.send(signal='SayResponse', response='Welcome')
00160     try:
00161         while True:
00162             #  We will wait here until shutdown
00163             sleep(1)
00164             if shutdown_flag.isSet():
00165                 break
00166     except KeyboardInterrupt:
00167         _logger.warning("Keyboard Interrupt received")
00168     except SystemExit:
00169         _logger.warning("System shutdown")
```

```
00170
00171     for _module in _loaded_modules:
00172         try:
00173             _logger.info('Unloading module %s' % _module)
00174             # Calling destructor will unload module
00175             del _module
00176         except:
00177             # Ignore all error while shutdown
00178             _logger.warning('Fail to unload module %s' % _module)
00179     _logger.info("All module unloaded")
00180
00181
00182 def emergency_shutdown():
00183     # Callback function of "EmergencyShutdown" event
00184     shutdown_flag.set()
00185
00186
00187 if __name__ == '__main__':
00188     # Shutdown flag
00189     shutdown_flag = threading.Event()
00190     # Start main function
00191     aria_start()
```

10.3 plugins/__init__.py File Reference

Namespaces

- plugins

10.4 __init__.py

```
00001 ## @file
00002 ## @brief Empty init file - allow to __import__ function to load data from folder
00003 ## @par
00004 # The __init__.py files are required to make Python treat the directories as containing packages;
00005 #  this is done to prevent directories with a common name, such as string,
00006 #  from unintentionally hiding valid modules that occur later (deeper) on the module search path.
00007 #  In the simplest case, __init__.py can just be an empty file, but it can also execute initialization code
00008 #  for the package or set the __all__ variable, described later.
00009 #
00010 ## @see https://docs.python.org/3/tutorial/modules.html#packages
```

10.5 plugins/AudioPlugin.py File Reference

Create instances for audio abstraction.

Classes

- class plugins.AudioPlugin.AudioSubSystem

  AudioSubSystem package.

Namespaces

- plugins.AudioPlugin

10.5.1 Detailed Description

Create instances for audio abstraction.

Create abstraction layer for audio sub-system par Configuration file

```
[Activation]
engine = pocketsphinx_continuous
options = -adcdev plughw:1,0 -dict $dict$ -lm $lang$ -inmic yes
dictionary = /home/pi/Aria2/plugins/lang_model/2613.dic
lang_model = /home/pi/Aria2/plugins/lang_model/2613.lm
log_redirect = -logfn /dev/null
auto_start= yes
hot_words = ARIA;HI ARIA;ASSUMING DIRECT CONTROL;DIRECT INTERVENTION IS NECESSARY;EMERGENCY SHUTDOWN

[Playback]
engine = aplay
options = $file$
log_redirect =

[Record]
engine = arecord
options = -c 1 -f S16_LE -r 16000 -d $time$ -N -M -D plughw:1,0 $file$
log_redirect =
max_record_time = 10
```

Definition in file AudioPlugin.py.

10.6 AudioPlugin.py

```
00001 ## @file
00002 ## @brief Create instances for audio abstraction
00003 ## @details Create abstraction layer for audio sub-system
00004 ## par Configuration file
00005 ## @verbinclude ./configuration/audio.conf
00006 #
00007 import ConfigParser
00008 import logging
00009 import subprocess
00010 import threading
00011 import shlex
00012 import subprocess
00013 from time import sleep
00014 from pydispatch import dispatcher
00015 from uuid import uuid4
00016
00017 ## @class AudioSubSystem
00018 ## @brief AudioSubSystem package
00019 ## @details Allow sound playing and recording
00020 ## @version 1.0.0.0
00021 class AudioSubSystem:
00022     ## @brief Plugin version
00023     version = "1.0.0.0"
00024     ## @brief Short plugin description
00025     description = "Audio sub-system"
00026
00027     ## @brief Create Audio subsystem instance
00028     ## @details Create and initialize instance
00029     ## @exception ImportError Configuration or IO system errorModule will be unloaded.
00030     ## @par Registering on events:
00031     # WaitToHotWordWait until Hot-Word not detected in audio input and send notification.\n
00032     # PlayFilePlay audio file.\n
00033     # RecordFileRecord audio input into file
00034     #
00035     ## @par Generate events:
00036     # HotWordDetectionActiveSet to True when STT engine trying to detect hot-word in audio stream.\n
00037     # GuiNotificationGUI tray update.\n
00038     # HotWordDetectedHot-word detected.
00039     # PlaybackActiveSet to True when audio player play file.\n
00040     # RecordActiveSet to True when audio recorder record audio into file.\n
00041     #
00042
```

```
00043     def __init__(self):
00044         ## @brief Event objectallow synchronize audio file play/record and STT engine
00045         self._hot_word_detection_active = threading.Event()
00046         ## @brief Allow bypass through Raspberry Pi IO system bug. Only one instance can control audio
      system
00047         self._io_system_busy = threading.Event()
00048         ## @brief Shutdown eventsignaling to all thread exit
00049         self._exit_flag = threading.Event()
00050         ## @brief Unique id for speaker try icon
00051         self._gui_speaker_status_uuid = str(uuid4())
00052         ## @brief Unique id for microphone try icon
00053         self._gui_microphone_status_uuid = str(uuid4())
00054         # Load logger
00055         try:
00056             ## @brief logger instance
00057             self._logger = logging.getLogger('Audio')
00058         except ConfigParser.NoSectionError as e:
00059             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00060             raise ImportError
00061         self._logger.debug('Audio sub-system logger started')
00062         # Reading config file
00063         try:
00064             ## @brief configuration file instnce
00065             self._config = ConfigParser.SafeConfigParser(allow_no_value=True)
00066             self._config.read('./configuration/audio.conf')
00067         except ConfigParser.Error as e:
00068             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00069             raise ImportError
00070         try:
00071             ## @brief List of activate words
00072             self._hot_words = self._config.get('Activation', 'hot_words').split(';')
00073             engine = self._config.get('Activation', 'engine')
00074             command_line = self._config.get('Activation', 'options')
00075             dictionary = self._config.get('Activation', 'dictionary')
00076             lang_model = self._config.get('Activation', 'lang_model')
00077             log_redirect = self._config.get('Activation', 'log_redirect')
00078             command_line = command_line.replace('$dict$', dictionary)
00079             command_line = command_line.replace('$lang$', lang_model)
00080             ## @brief command line to activate hot-word detection engine
00081             self._recognition_engine = shlex.split(engine + ' ' + command_line + ' ' +
      log_redirect)
00082
00083             playback_engine = self._config.get('Playback', 'engine')
00084             playback_option = self._config.get('Playback', 'options')
00085             playback_log_redirect = self._config.get('Playback', 'log_redirect')
00086             ## @brief command line to activate file playback
00087             self._playback_engine = shlex.split(playback_engine + ' ' + playback_option + '
      ' + playback_log_redirect)
00088
00089             record_engine = self._config.get('Record', 'engine')
00090             record_option = self._config.get('Record', 'options')
00091             record_log_redirect = self._config.get('Record', 'log_redirect')
00092             ## @brief Maximum allowed voice record time
00093             self._max_record_time = self._config.get('Record', 'max_record_time')
00094             ## @brief command line to activate record engine
00095             self._record_engine = shlex.split(record_engine + ' ' + record_option + ' ' +
      record_log_redirect)
00096
00097             if self._config.getboolean('Activation', 'auto_start'):
00098                 self.start_hot_word_detection(10)
00099         except ConfigParser.Error as e:
00100             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00101             raise ImportError
00102         # Register on Events
00103         try:
00104             dispatcher.connect(self.start_hot_word_detection, signal='WaitToHotWord
      ', sender=dispatcher.Any)
00105         except dispatcher.DispatcherTypeError as e:
00106             self._logger.error('Fail to subscribe on "WaitToHotWord" event with error %s.Module unload' % e
      )
00107             raise ImportError
00108
00109         try:
00110             dispatcher.connect(self.play_file, signal='PlayFile', sender=dispatcher.Any)
00111         except dispatcher.DispatcherTypeError as e:
00112             self._logger.error('Fail to subscribe on "PlayFile" event with error %s.Module unload' % e)
00113             raise ImportError
00114
00115         try:
00116             dispatcher.connect(self.record_file, signal='RecordFile', sender=dispatcher.Any)
00117         except dispatcher.DispatcherTypeError as e:
00118             self._logger.error('Fail to subscribe on "RecordFile" event with error %s.Module unload' % e)
00119             raise ImportError
00120
00121     ## @brief Stop module
00122     ## @details Stop all module thread and sub-programs
00123     def __del__(self):
```

```
00124         dispatcher.disconnect(self.start_hot_word_detection)
00125         dispatcher.disconnect(self.play_file)
00126         self._exit_flag.set()
00127         sleep(5)
00128         if self._hot_word_detection_active.isSet():
00129             self._logger.error('Fail to stop recognition process')
00130         if self._io_system_busy.isSet():
00131             self._logger.error('Fail to stop playback process')
00132         self._logger.debug('Audio module release')
00133 
00134     ## @brief WaitToHotWord event wrapper
00135     ## @details Initialize thread that allow to communication with STT engine
00136     ## @param delay float Delay before start STT engineoptional. Default - 0
00137     def start_hot_word_detection(self, delay=None):
00138         if self._exit_flag.is_set():
00139             self._logger.warning('Shutdown flag set. Ignoring start command')
00140             return
00141         if self._hot_word_detection_active.isSet():
00142             self._logger.warning('Recognizing already running. Ignoring')
00143             return
00144         self._logger.info('Starting Hot word detection')
00145         try:
00146             threading.Thread(target=self._start_hot_word_detection, args=(delay,))
    .start()
00147         except threading.ThreadError as e:
00148             self._logger.error('Fail top start detection thread with error %s' % e)
00149 
00150     ## @brief WaitToHotWord thread
00151     ## @details Communicate with STT engine
00152     ## @param delay float Delay before start STT engineoptional. Default - 0
00153     ## @warning This function should not be called from outside
00154     ## @par Generate events:
00155     # HotWordDetectionActiveSet to True when STT engine trying to detect hot-word in audio stream.\n
00156     # GuiNotificationGUI tray update.\n
00157     # HotWordDetectedHot-word detected.
00158     #
00159     ## @see guiPlugin
00160     def _start_hot_word_detection(self, delay=None):
00161         if delay is not None:
00162             sleep(delay)
00163         self._logger.info('Starting recognize process')
00164         dispatcher.send(signal='HotWordDetectionActive', status=True)
00165         dispatcher.send(signal='GuiNotification', source=self.
    _gui_microphone_status_uuid,
00166                         icon_path="microphone_passive.png")
00167 
00168         _recognize_process = subprocess.Popen(self._recognition_engine, stdout=
    subprocess.PIPE)
00169         self._hot_word_detection_active.set()
00170         while not self._exit_flag.isSet():
00171             line = _recognize_process.stdout.readline().replace('\n', ' ').replace('\r', '')
00172             if self._io_system_busy.isSet():
00173                 self._logger.info('Playback started.Stop recognition process')
00174                 dispatcher.send(signal='HotWordDetectionActive', status=False)
00175                 dispatcher.send(signal='GuiNotification', source=self.
    _gui_microphone_status_uuid,
00176                                 icon_path="microphone_off.png")
00177                 _recognize_process.terminate()
00178                 self._hot_word_detection_active.clear()
00179                 while self._io_system_busy.isSet():
00180                     sleep(1)
00181                 dispatcher.send(signal='HotWordDetectionActive', status=True)
00182                 dispatcher.send(signal='GuiNotification', source=self.
    _gui_microphone_status_uuid,
00183                                 icon_path="microphone_passive.png")
00184                 _recognize_process = subprocess.Popen(self._recognition_engine, stdout=
    subprocess.PIPE)
00185                 self._hot_word_detection_active.set()
00186             if line != '':
00187                 for word in self._hot_words:
00188                     if word in line:
00189                         _recognize_process.terminate()
00190                         self._logger.info('Hot word %s detected in input %s' % (word, line))
00191                         self._logger.info('Stop recognition process')
00192                         dispatcher.send(signal='HotWordDetected', text=word)
00193                         dispatcher.send(signal='HotWordDetectionActive', status=False)
00194                         dispatcher.send(signal='GuiNotification', source=self.
    _gui_microphone_status_uuid,
00195                                         icon_path="microphone_off.png")
00196                         self._hot_word_detection_active.clear()
00197                         return
00198                     if "EMERGENCY SHUTDOWN" in line:
00199                         self._logger.warning("EMERGENCY SHUTDOWN")
00200                         bashCommand = "killall python"
00201                         subprocess.Popen(bashCommand.split())
00202 
00203     ## @brief PlayFile event wrapper
```

```
00204     ## @details Initialize thread that allow to communication audio player
00205     ## @param filename string Path to audio file
00206     ## @param delay float Delay before start STT engine.Optional. Default - 0
00207     ## @param callback obj Callback function when playback completed.Optional. Default - None
00208     def play_file(self, filename, delay=None, callback=None):
00209         if self._exit_flag.is_set():
00210             self._logger.warning('Shutdown flag set. Ignoring start command')
00211             return
00212         self._logger.info('Starting file playback')
00213         try:
00214             threading.Thread(target=self._play_file, args=(filename, delay, callback)).start()
00215         except threading.ThreadError as e:
00216             self._logger.error('Fail to start playback thread with error %s' % e)
00217 
00218     ## @brief PlayFile thread
00219     ## @details Communicate with audio player
00220     ## @param filename string Path to audio file
00221     ## @param delay float Delay before start STT engine - optional. Default - 0
00222     ## @param callback obj Callback function when playback completed - optional. Default - None
00223     ## @warning This function should not be called from outside
00224     ## @par Generate events:
00225     # PlaybackActiveSet to True when audio player play file.\n
00226     # GuiNotificationGUI tray update.
00227     #
00228     ## @see guiPlugin
00229     def _play_file(self, filename, delay=None, callback=None):
00230         if delay is not None:
00231             sleep(delay)
00232         if self._io_system_busy.isSet():
00233             self._logger.warning('Another playback active waiting to end')
00234         while self._io_system_busy.isSet():
00235             sleep(1)
00236         self._io_system_busy.set()
00237         if self._hot_word_detection_active.isSet():
00238             self._logger.warning('Hot word detection running waiting to termination')
00239         while self._hot_word_detection_active.isSet():
00240             sleep(1)
00241         try:
00242             dispatcher.send(signal='PlaybackActive', status=True)
00243             dispatcher.send(signal='GuiNotification', source=self.
      _gui_speaker_status_uuid, icon_path="speaking.png")
00244             subprocess.call([s.replace('$file$', filename) for s in self.
      _playback_engine])
00245         except OSError as e:
00246             self._logger.error('Fail to play file %s with error %s' % (filename, e))
00247         finally:
00248             dispatcher.send(signal='PlaybackActive', status=False)
00249             dispatcher.send(signal='GuiNotification', source=self.
      _gui_speaker_status_uuid, icon_path="speaker_off.png")
00250             self._io_system_busy.clear()
00251 
00252         if callable(callback):
00253             callback()
00254 
00255     ## @brief RecordFile event wrapper
00256     ## @details Initialize thread that allow to communication audio recorder
00257     ## @param filename string Path to audio file
00258     ## @param record_time float Record timeoptional. Default - maximum allowed time as set in config file
00259     ## @param delay float Delay before start STT engine - optional. Default - 0
00260     ## @param callback obj Callback function when playback completed - optional. Default - None
00261     def record_file(self, filename, record_time=None, delay=None, callback=None):
00262         if self._exit_flag.is_set():
00263             self._logger.warning('Shutdown flag set. Ignoring start command')
00264             return
00265         if record_time is None:
00266             record_time = self._max_record_time
00267         elif record_time > self._max_record_time:
00268             self._logger.warning('Record time too large reducing')
00269         self._logger.info('Starting audio record')
00270         try:
00271             threading.Thread(target=self._record_file, args=(filename, record_time, delay,
      callback)).start()
00272         except threading.ThreadError as e:
00273             self._logger.error('Fail to start audio record thread with error %s' % e)
00274 
00275     ## @brief Record thread
00276     ## @details Communicate with audio recorder
00277     ## @param filename string Path to audio file
00278     ## @param record_time float Record time - optional. Default - maximum allowed time as set in config
       file
00279     ## @param delay float Delay before start STT engine - optional. Default - 0
00280     ## @param callback obj Callback function when playback completed - optional. Default - None
00281     ## @warning This function should not be called from outside
00282     ## @par Generate events:
00283     # RecordActive - Set to True when audio recorder record audio into file.\n
00284     # GuiNotification - GUI tray update.
00285     #
```

```
00286     ## @see guiPlugin
00287     def _record_file(self, filename, record_time, delay=None, callback=None):
00288         if delay is not None:
00289             sleep(delay)
00290         if self._io_system_busy.isSet():
00291             self._logger.warning('Another playback active waiting to end')
00292         while self._io_system_busy.isSet():
00293             sleep(1)
00294         self._io_system_busy.set()
00295         if self._hot_word_detection_active.isSet():
00296             self._logger.warning('Hot word detection running waiting to termination')
00297         while self._hot_word_detection_active.isSet():
00298             sleep(1)
00299 
00300         try:
00301             call_command = [s.replace('$file$', filename) for s in self.
    _record_engine]
00302             call_command = [s.replace('$time$', record_time) for s in call_command]
00303             dispatcher.send(signal='RecordActive', status=True)
00304             dispatcher.send(signal='GuiNotification', source=self.
    _gui_microphone_status_uuid,
00305                             icon_path="microphone_record.png")
00306             subprocess.call(call_command)
00307         except OSError as e:
00308             self._logger.error('Fail to record file %s with error %s' % (filename, e))
00309         finally:
00310             self._io_system_busy.clear()
00311             dispatcher.send(signal='RecordActive', status=False)
00312             dispatcher.send(signal='GuiNotification', source=self.
    _gui_microphone_status_uuid,
00313                             icon_path="microphone_off.png")
00314 
00315         if callable(callback):
00316             callback(filename)
```

10.7 plugins/EmailPlugin.py File Reference

Zoho email communication sub-system.

Classes

- class plugins.EmailPlugin.ZohoEmail

  Email plugin.

Namespaces

- plugins.EmailPlugin

10.7.1 Detailed Description

Zoho email communication sub-system.

Allow to comunicate with Zoho email server using POP3 protocol par Configuration file

```
[API]
system = zoho
user = ariatloak
[General]
update_interval=2
[Server]
url=
port=
```

Definition in file EmailPlugin.py.

10.8 EmailPlugin.py

```
00001 ## @file
00002 ## @brief Zoho email communication sub-system
00003 ## @details Allow to comunicate with Zoho email server using POP3 protocol
00004 ## par Configuration file
00005 ## @verbinclude ./configuration/email.conf
00006 #
00007 import ConfigParser
00008 import logging
00009 import threading
00010 import time
00011 import dateutil.parser
00012 from uuid import uuid4
00013 import socket
00014
00015 from email import parser
00016 import poplib
00017
00018 import keyring
00019 from pydispatch import dispatcher
00020
00021
00022 ## @class ZohoEmail
00023 ## @brief Email plugin
00024 ## @details Communication with Zoho (https://www.zoho.com/) email and retrieve email using POP3 protocol
00025 ## @version 1.0.0.0
00026 class ZohoEmail:
00027     ## @brief Plugin version
00028     version = '1.0.0.0'
00029     ## @brief Plugin description
00030     description = 'Zoho email client module'
00031
00032     ## @brief Create Email interface instance
00033     ## @details Create and initialize instance
00034     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00035     ## @par Registering on events:
00036     # SpeechRecognize - User speech input.\n
00037     #
00038     ## @par Generate events:
00039     # GuiNotification - GUI tray update.\n
00040     #
00041     ## @see guiPlugin
00042
00043     def __init__(self):
00044         ## @brief Unique id for try icon
00045         self._gui_status = str(uuid4())
00046         ## @brief Shutdown event - signal to all thread exit
00047         self._shutdown = threading.Event()
00048         ## @brief Message list
00049         self._message_list = dict()
00050         ## @brief Message list synchronization object - allow thread safe update
00051         self._message_list_token = threading.Lock()
00052         try:
00053             ## @brief logger instance
00054             self._logger = logging.getLogger('moduleEmail')
00055         except ConfigParser.NoSectionError as e:
00056             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00057             raise ImportError
00058         self._logger.debug('Email logger started')
00059         # Reading config file
00060         try:
00061             ## @brief configuration file intance
00062             self._config = ConfigParser.SafeConfigParser(allow_no_value=False)
00063             self._config.read('./configuration/email.conf')
00064             api_system = self._config.get('API', 'system')
00065             ## @brief User name for email server
00066             self.api_user = self._config.get('API', 'user')
00067             try:
00068                 ## @brief Password for Email server
00069                 self.api_key = keyring.get_password(api_system, self.
      api_user)
00070             except keyring.errors as e:
00071                 self._logger.warning('Fail to read Zoho token with error: %s. Refer to manual. Module
       unload' % e)
00072                 raise ImportError
00073             if self.api_key is None:
00074                 self._logger.warning('Fail to read Zoho token. Refer to manual. Module unload')
00075                 raise ImportError
00076
00077             ## @brief Email list refresh interval
00078             self._update_interval = self._config.getint('General', 'update_interval')
00079
00080             try:
00081                 ## @brief Zoho email sever URL
00082                 self._server_url = self._config.get('Server', 'url')
```

```
00083                 ## @brief Zoho email server communication protocol
00084                 self._server_port = self._config.getint('Server', 'port')
00085             except (ConfigParser.Error, ValueError):
00086                 self._server_url = 'pop.zoho.com'
00087                 self._server_port = 995
00088
00089         except ConfigParser.Error as e:
00090             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00091             raise ImportError
00092
00093         try:
00094             # register on user input
00095             dispatcher.connect(self.user_request, signal='SpeechRecognize', sender=
      dispatcher.Any)
00096         except dispatcher.DispatcherTypeError as e:
00097             self._logger.error('Fail to subscribe on "SpeechRecognize" event with error %s.Module unload' %
       e)
00098             raise ImportError
00099
00100         self._logger.debug("Starting periodic update thread")
00101         try:
00102             threading.Thread(target=self._periodic_update).start()
00103         except OSError as e:
00104             self._logger.warning('Fail to start periodic update thread with error %s' % e)
00105
00106         self._logger.info('Weather module ready')
00107
00108     ## @brief Stop module
00109     ## @details Stop all module thread and sub-programs
00110     def __del__(self):
00111         dispatcher.disconnect(self.user_request)
00112         self._shutdown.set()
00113         self._logger.debug('Email module release')
00114
00115     ## @brief Connect to email server
00116     ## @details Connect and login to email server
00117     ## @warning This function should not be called from outside
00118     ## @par Generate events:
00119     # GuiNotification - GUI tray update.\n
00120     #
00121     ## @see guiPlugin
00122     def _connect(self):
00123         dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="
      email_refresh.png")
00124         try:
00125             pop_conn = poplib.POP3_SSL(self._server_url, self.
      _server_port)
00126
00127             pop_conn.user("%s@zoho.com" % self.api_user)
00128             pass_response = pop_conn.pass_(self.api_key)
00129             if "+OK" in pass_response:
00130                 self._logger.debug('Connected to Email server')
00131             else:
00132                 self._logger.warning('Fail to connect.Please check password\username. Got response %s' %
      pass_response)
00133                 return None
00134         except poplib.error_proto as e:
00135             dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="
      email_error.png")
00136             self._logger.warning("Fail to connect.Error %s" % e)
00137             return None
00138         except socket.error as e:
00139             dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="
      email_error.png")
00140             self._logger.warning("Socket error %s" % e)
00141             return None
00142         else:
00143             self._logger.debug('Pass response %s' % pass_response)
00144             return pop_conn
00145
00146     ## @brief Wait to new emails
00147     ## @details Refresh emails on email server
00148     ## @warning This function should not be called from outside
00149     ## @par Generate events:
00150     # GuiNotification - GUI tray update.\n
00151     # SpeechAccepted - Notify that module start process user request.\n
00152     # RestartInteraction - Restart Hot-Word detection.\n
00153     # SayText - Response to user request using TTS engine.\n
00154     #
00155     ## @see guiPlugin
00156     ## @see AudioSubSystem
00157     ## @see TtsPlugin
00158     def _periodic_update(self):
00159         time.sleep(15)
00160         pop_conn = None
00161         while pop_conn is None:
00162             pop_conn = self._connect()
```

```
00163             if pop_conn is None:
00164                 self._shutdown.wait(self._update_interval)
00165 
00166         while not self._shutdown.isSet():
00167             try:
00168                 self._logger.debug('Refreshing email list')
00169                 messages = [pop_conn.retr(i) for i in range(1, len(pop_conn.list()[1]) + 1)]
00170                 messages = ["\n".join(mssg[1]) for mssg in messages]
00171                 messages = [parser.Parser().parsestr(mssg) for mssg in messages]
00172                 new_message = False
00173 
00174                 self._message_list_token.acquire()
00175 
00176                 for message in messages:
00177                     if not (message['Message-ID'] in self._message_list):
00178                         self._logger.info('New message found')
00179                         self._message_list[str(message['Message-ID'])] = dict(Subject=str(
      message['Subject']),
00180                                                                               From=str(message['From']),
00181                                                                               Time=int(time.mktime(
00182                                                                                   dateutil.parser.parse(
00183                                                                                       message['Date']).
      timetuple())))
00184                         if time.time() - self._message_list[message['Message-ID']]['Time'] < (
      1 * 60 * 60):
00185                             new_message = True
00186 
00187                 self._message_list_token.release()
00188 
00189                 if new_message:
00190                     dispatcher.send(signal='GuiNotification', source=self.
      _gui_status, icon_path="new_email.png")
00191                 else:
00192                     dispatcher.send(signal='GuiNotification', source=self.
      _gui_status, icon_path="")
00193 
00194                 for i in range(0, 60 * self._update_interval, 30):
00195                     self._shutdown.wait(30)
00196                     self._logger.debug('Sending NOOP')
00197                     pop_conn.noop()
00198 
00199                     if self._shutdown.isSet():
00200                         pop_conn.quit()
00201                         return
00202             except (socket.error, poplib.error_proto) as e:
00203                 self._logger.warning('Got error - %s. Reconnecting' % e)
00204                 pop_conn = None
00205                 while pop_conn is None:
00206                     pop_conn = self._connect()
00207                     if pop_conn is None:
00208                         self._shutdown.wait(self._update_interval)
00209                         if self._shutdown.isSet():
00210                             pop_conn.quit()
00211                             return
00212 
00213     ## @brief SpeechRecognize event wrapper
00214     ## @details If user request contain email entity with high confidence begin request process
00215     ## @warning This function should not be called from outside
00216     ## @par Generate events:
00217     # GuiNotification - GUI tray update.\n
00218     #
00219     ## @param entities dictionary Speech entities
00220     ## @param raw_text string Ignored
00221     ## @see guiPlugin
00222     ## @see TtsPlugin
00223     def user_request(self, entities, raw_text):
00224         if "mail" in entities and entities['mail'][0]['confidence'] > 0.5:
00225             dispatcher.send(signal='SpeechAccepted')
00226             self._logger.debug("Starting email fetch thread")
00227             try:
00228                 threading.Thread(target=self._user_request, args=(entities,)).start()
00229             except OSError as e:
00230                 self._logger.warning('Fail to start fetch thread with error %s' % e)
00231 
00232     ## @brief Analyze user request
00233     ## @details Extract data from user request and generate response
00234     ## @warning This function should not be called from outside
00235     ## @par Generate events:
00236     # GuiNotification - GUI tray update.\n
00237     # SayText - Response to user request using TTS engine.\n
00238     #
00239     ## @param entities dictionary Speech entities
00240     ## @see guiPlugin
00241     ## @see TtsPlugin
00242     def _user_request(self, entities):
00243         if 'contact' in entities:
00244             if str(entities['contact'][0]['value']) != 'i':
```

```
00245                 search_person = str(entities['contact'][0]['value'])
00246             else:
00247                 search_person = None
00248         else:
00249             search_person = None
00250 
00251         if search_person is None:
00252             # ask for update only
00253             new_email = 0
00254             self._message_list_token.acquire()
00255             for message_id, message_data in self._message_list.iteritems():
00256                 if (time.time() - message_data['Time']) < (1 * 60 * 60):
00257                     new_email += 1
00258             if new_email == 0:
00259                 dispatcher.send(signal='SayText', text="You don't have any new email from last hour",
00260                                 callback=self.sythsys_complete)
00261             else:
00262                 dispatcher.send(signal='SayText', text="You receive %i new email in last hour" % new_email,
00263                                 callback=self.sythsys_complete)
00264         else:
00265             new_email = 0
00266             self._message_list_token.acquire()
00267             for message_id, message_data in self._message_list.iteritems():
00268                 if ((time.time() - message_data['Time']) < (1 * 60 * 60)) and (search_person in
    message_data['From']):
00269                     new_email += 1
00270             if new_email == 0:
00271                 dispatcher.send(signal='SayText',
00272                                 text="You don't have any new email from %s in last hour" % search_person,
00273                                 callback=self.sythsys_complete)
00274             else:
00275                 dispatcher.send(signal='SayText', text="You receive %i new email from %s in last hour" %
00276                                                        (new_email, search_person), callback=self.
    sythsys_complete)
00277 
00278         self._message_list_token.release()
00279         dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="")
00280 
00281     ## @brief Restart user interaction
00282     ## @details After request process restart hot-word detection process
00283     ## @warning This function should not be called from outside
00284     ## @par Generate events:
00285     # RestartInteraction - restart Hot-Word detection.\n
00286     #
00287     ## @see SttPlugin
00288     @staticmethod
00289     def sythsys_complete():
00290         dispatcher.send(signal='RestartInteraction')
```

10.9 plugins/guiPlugin.py File Reference

GUIModule.

Classes

- class plugins.guiPlugin.Gui

  Main GUI.

- class plugins.guiPlugin.GuiPlugin

  Init wx-python GUI.

Namespaces

- plugins.guiPlugin

10.9.1 Detailed Description

GUIModule.

Contain all gui function

Configuration file

```
## @file
# GUI configuration file
[API]
system = Facebook
user = Develop
[General]
animation_steps=10
temp_folder=/home/pi/Aria2/tmp/gui/
clear_after = 7
change_after = 10
animation_speed = 0.1
animation = 0
[Facebook]
Skip_albums = Profile Pictures;''
```

Definition in file guiPlugin.py.

10.10 guiPlugin.py

```
00001 # coding=utf-8
00002 ## @file
00003 ## @brief GUIModule
00004 ## @details Contain all gui function
00005 ## @par Configuration file
00006 ## @verbinclude ./configuration/gui.conf
00007 #
00008 import ConfigParser
00009 import logging
00010 from pydispatch import dispatcher
00011 import os
00012 import time
00013 import datetime
00014 import threading
00015 from scipy import misc
00016 import numpy as np
00017 import keyring
00018 import urllib
00019 import json
00020 import facebook
00021
00022 import wx
00023
00024
00025 ## @class Gui
00026 ## @brief Main GUI
00027 ## @details Create GUI interface with Facebook profile pictures
00028 ## @see https://developers.facebook.com/tools/explorer/145634995501895/?method=GET&path=&version=v2.7
00029 ## @version 1.0.0.0
00030 class Gui(wx.Frame):
00031     ## @brief Create GUI based on wx python
00032     ## @details Create and initialize instance start fetching and periodic update threads
00033     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00034     ## @par Registering on events:
00035     # GuiNotification - User speech input.\n
00036     # SayText - System to user response text.\n
00037     # SpeechRecognize - User to system text.\n
00038     # WeatherUpdate - Periodic weather update.\n
00039     #
00040     ## @see SttPlugin
00041     ## @see WeatherPlugin
00042     ## @see AudioSubSystem
00043     def __init__(self, *args, **kwds):
00044         ## @brief Shutdown flag - notify to threads exits
00045         self._shutdown = threading.Event()
00046         ## @brief GUI synchronization - avoid GUI update from different threads
```

```
00047         self._gui_update_lock = threading.Lock()
00048         ## @brief dictionary of notification icons and their owners
00049         self._notification_tray = {}
00050         try:
00051             self._logger = logging.getLogger('moduleGui')
00052         except ConfigParser.NoSectionError as e:
00053             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00054             raise ImportError
00055         self._logger.debug('GUI logger started')
00056 
00057         # Reading config file
00058         try:
00059             self._config = ConfigParser.SafeConfigParser(allow_no_value=False)
00060             self._config.read('./configuration/gui.conf')
00061             api_system = self._config.get('API', 'system')
00062             api_user = self._config.get('API', 'user')
00063             try:
00064                 self.api_key = keyring.get_password(api_system, api_user)
00065             except keyring.errors as e:
00066                 self._logger.warning(
00067                     'Fail to read Facebook token with error: %s. Refer to manual. Module unload' % e)
00068                 raise ImportError
00069             if self.api_key is None:
00070                 self._logger.warning('Fail to read Facebook token. Refer to manual. Module unload')
00071                 raise ImportError
00072             self._animation_steps = self._config.getint('General', 'animation_steps')
00073             self._clear_delay = self._config.getint('General', 'clear_after')
00074             self._change_after = self._config.getint('General', 'change_after')
00075             self._animation_speed = self._config.getfloat('General', 'animation_speed')
00076 
00077             self._ignore_albums = self._config.get('Facebook', 'Skip_albums')
00078             self._ignore_albums.split(';')
00079 
00080             self._animation_active = self._config.getboolean('General', 'animation')
00081 
00082             self._temp_folder = self._config.get('General', 'temp_folder')
00083             if not os.path.exists(self._temp_folder):
00084                 try:
00085                     os.makedirs(self._temp_folder)
00086                 except IOError as e:
00087                     self._logger.error('Fail to temporary folder with error %s.Module unload' % e)
00088                     raise ImportError
00089         except ConfigParser.Error as e:
00090             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00091             raise ImportError
00092 
00093         # begin wxGlade: Gui.__init__
00094         # kwds["style"] = kwds.get("style", 0) | wx.FRAME_TOOL_WINDOW | wx.STAY_ON_TOP
00095         kwds["style"] = kwds.get("style", 0)
00096         wx.Frame.__init__(self, *args, **kwds)
00097         self.SetSize((800, 510))
00098 
00099         # Controls
00100         self._animation_circle_bmp = None
00101         self._system_response_bmp = None
00102 
00103         self._system_response_lbl = None
00104         self._user_request_lbl = None
00105 
00106         self._clock_lbl = None
00107         self._date_lbl = None
00108 
00109         self.weather_desc_lbl = None
00110         self._weather_temp_lbl = None
00111         self.weather_wind_lbl = None
00112         self._weather_icon = None
00113 
00114         self._main_picture_bmp = None
00115 
00116         self._notification_slots = []
00117 
00118         self.__set_properties()
00119         self.__do_layout()
00120         # end wxGlade
00121         # Microphone activity
00122         self._logger.debug('Registering on events')
00123         try:
00124             dispatcher.connect(self._notification, signal='GuiNotification', sender=
      dispatcher.Any)
00125             dispatcher.connect(self._system_response_text, signal='SayText', sender=
      dispatcher.Any)
00126             dispatcher.connect(self._user_request_text, signal='SpeechRecognize', sender=
      dispatcher.Any)
00127             dispatcher.connect(self._weather_display, signal='WeatherUpdate', sender=
      dispatcher.Any)
00128         except dispatcher.DispatcherTypeError as e:
00129             self._logger.error('Fail to subscribe on event with error %s.Module unload' % e)
```

```
00130             raise ImportError
00131
00132         self._logger.info('Starting animation and update threads')
00133         try:
00134             threading.Thread(target=self._animation_update).start()
00135             threading.Thread(target=self._time_update).start()
00136             if self._animation_active:
00137                 threading.Thread(target=self.main_pic_animation).start()
00138             else:
00139                 self._logger.info('Animation disabled')
00140         except Exception as e:
00141             self._logger.error('Fail to start thread with error %s.Module unload' % e)
00142             raise ImportError
00143
00144     ## @brief Stop module
00145     ## @details Stop all module thread and sub-programs
00146     def __del__(self):
00147         self._logger.info('Module unload')
00148         self._shutdown.set()
00149         self.Destroy()
00150
00151     ## @brief Set GUI properties
00152     ## @details Update GUI elements and their properties
00153     ## @note This function generated create be WxGlade
00154     ## @warning This function should not be called from outside
00155     def __set_properties(self):
00156         # begin wxGlade: Gui.__set_properties
00157         self.SetTitle("Aria")
00158         # end wxGlade
00159
00160     ## @brief Set GUI layout
00161     ## @details Update GUI elements and their layouts
00162     ## @note This function generated create be WxGlade
00163     ## @warning This function should not be called from outside
00164     def __do_layout(self):
00165         # begin wxGlade: Gui.__do_layout
00166         sizer_1 = wx.BoxSizer(wx.VERTICAL)
00167         sizer_4 = wx.BoxSizer(wx.VERTICAL)
00168         sizer_6 = wx.BoxSizer(wx.HORIZONTAL)
00169         sizer_5 = wx.BoxSizer(wx.HORIZONTAL)
00170         sizer_2 = wx.BoxSizer(wx.HORIZONTAL)
00171         sizer_7 = wx.BoxSizer(wx.VERTICAL)
00172         sizer_8 = wx.BoxSizer(wx.VERTICAL)
00173         sizer_10 = wx.BoxSizer(wx.HORIZONTAL)
00174         sizer_9 = wx.BoxSizer(wx.HORIZONTAL)
00175         sizer_3 = wx.BoxSizer(wx.VERTICAL)
00176         self._animation_circle_bmp = wx.StaticBitmap(self, wx.ID_ANY, wx.Bitmap("
    ./plugins/Icons/Load/frame-0.png", wx.BITMAP_TYPE_ANY))
00177         self._animation_circle_bmp.SetMinSize((25, 25))
00178         sizer_3.Add(self._animation_circle_bmp, 0, 0, 0)
00179         for i in range(15):
00180             data_slot = wx.StaticBitmap(self, wx.ID_ANY, wx.Bitmap("./plugins/Icons/empty.png",
    wx.BITMAP_TYPE_ANY))
00181             data_slot.SetMinSize((25, 25))
00182             sizer_3.Add(data_slot, 0, 0, 0)
00183             self._notification_slots.append(data_slot)
00184         sizer_2.Add(sizer_3, 0, wx.ALIGN_CENTER | wx.EXPAND, 0)
00185         self._main_picture_bmp = wx.StaticBitmap(self, wx.ID_ANY,
00186                                                  wx.Bitmap("./plugins/Icons/login.png", wx.BITMAP_TYPE_ANY)
    )
00187         self._main_picture_bmp.SetMinSize((630, 430))
00188         sizer_2.Add(self._main_picture_bmp, 0, 0, 0)
00189         self._clock_lbl = wx.StaticText(self, wx.ID_ANY, "", style=wx.ALIGN_CENTER)
00190         self._clock_lbl.SetMinSize((115, 55))
00191         self._clock_lbl.SetFont(wx.Font(35, wx.DEFAULT, wx.NORMAL, wx.LIGHT, 0, "Ubuntu"))
00192         sizer_7.Add(self._clock_lbl, 0, 0, 0)
00193         self._date_lbl = wx.StaticText(self, wx.ID_ANY, "")
00194         self._date_lbl.SetMinSize((123, 30))
00195         self._date_lbl.SetFont(wx.Font(20, wx.DEFAULT, wx.NORMAL, wx.LIGHT, 0, ""))
00196         sizer_7.Add(self._date_lbl, 0, 0, 0)
00197         self.weather_desc_lbl = wx.StaticText(self, wx.ID_ANY, "Updating ...")
00198         self.weather_desc_lbl.SetMinSize((160, 25))
00199         sizer_8.Add(self.weather_desc_lbl, 0, 0, 0)
00200         self._weather_icon = wx.StaticBitmap(self, wx.ID_ANY,
00201                                              wx.Bitmap("./plugins/Icons/weather_none.png",
    wx.BITMAP_TYPE_ANY))
00202         self._weather_icon.SetMinSize((50, 50))
00203         sizer_9.Add(self._weather_icon, 0, 0, 0)
00204         self._weather_temp_lbl = wx.StaticText(self, wx.ID_ANY, "", style=wx.ALIGN_RIGHT)
00205         # self._weather_temp_lbl.SetMinSize((80, 50))
00206         self._weather_temp_lbl.SetFont(wx.Font(20, wx.DEFAULT, wx.NORMAL, wx.NORMAL, 0, "Noto Sans"))
00207         sizer_9.Add(self._weather_temp_lbl, 0, 0, 0)
00208         sizer_8.Add(sizer_9, 0, 0, 0)
00209         self.weather_wind_lbl = wx.StaticText(self, wx.ID_ANY, "", style=wx.ALIGN_CENTER)
00210         self.weather_wind_lbl.SetMinSize((145, 50))
00211         sizer_10.Add(self.weather_wind_lbl, 0, wx.ALL, 1)
00212         sizer_8.Add(sizer_10, 0, 0, 0)
```

```
00213         sizer_8.Add((0, 0), 0, 0, 0)
00214         sizer_8.Add((0, 0), 0, 0, 0)
00215         sizer_7.Add(sizer_8, 0, 0, 0)
00216         sizer_2.Add(sizer_7, 1, wx.EXPAND, 0)
00217         sizer_1.Add(sizer_2, 1, wx.EXPAND, 0)
00218         self._user_request_lbl = wx.StaticText(self, wx.ID_ANY, "", style=wx.ALIGN_RIGHT)
00219         self._user_request_lbl.SetMinSize((765, 25))
00220         sizer_5.Add(self._user_request_lbl,  1, wx.EXPAND, 0)
00221         bitmap_9 = wx.StaticBitmap(self, wx.ID_ANY, wx.Bitmap("./plugins/Icons/user.png",
      wx.BITMAP_TYPE_ANY))
00222         bitmap_9.SetMinSize((25, 25))
00223         sizer_5.Add(bitmap_9, 0, wx.EXPAND, 0)
00224         sizer_4.Add(sizer_5, 1, wx.EXPAND, 0)
00225         self._system_response_bmp = wx.StaticBitmap(self, wx.ID_ANY,
00226                                                     wx.Bitmap("./plugins/Icons/response_good.png",
      wx.BITMAP_TYPE_ANY))
00227         self._system_response_bmp.SetMinSize((25, 25))
00228         sizer_6.Add(self._system_response_bmp, 0, 0, 0)
00229         self._system_response_lbl = wx.StaticText(self, wx.ID_ANY, "", style=
      wx.ALIGN_LEFT)
00230         self._system_response_lbl.SetMinSize((765, 25))
00231         sizer_6.Add(self._system_response_lbl, 0, wx.ALIGN_CENTER | wx.ALL, 0)
00232         sizer_4.Add(sizer_6, 1, wx.EXPAND, 0)
00233         sizer_1.Add(sizer_4, 1, wx.EXPAND, 0)
00234         self.SetSizer(sizer_1)
00235         self.Layout()
00236         self.Centre()
00237         # end wxGlade
00238
00239     ## @brief GUI element update
00240     ## @details Thread safe GUI update
00241     ## @param func - WxPython function
00242     ## @param data - Configuration data for WxPython update function
00243     def safe_update(self, func, data):
00244         self._gui_update_lock.acquire()
00245         func(data)
00246         self._gui_update_lock.release()
00247
00248     ## @brief Wrapper for delayed GUI update
00249     ## @details Thread safe GUI update with delay
00250     ## @param func - WxPython function
00251     ## @param data - Configuration data for WxPython update function
00252     def safe_update_delay(self, func, data):
00253         threading.Thread(target=self._safe_update_delay, args=(func, data)).start()
00254
00255     ## @brief Delayed GUI update
00256     ## @details Thread safe GUI update with delay
00257     ## @param func - WxPython function
00258     ## @param data - Configuration data for WxPython update function
00259     def _safe_update_delay(self, func, data):
00260         time.sleep(self._clear_delay)
00261         wx.CallAfter(self.safe_update, func, data)
00262
00263     ## @brief Wrapper for tray update
00264     ## @details Thread safe tray update
00265     ## @param source - Unique id of caller
00266     ## @param icon_path - Relative path of icon for tray
00267     def _notification(self, source, icon_path):
00268         if source in self._notification_tray:
00269             if icon_path == '':
00270                 self._logger.debug('Removing notification from %s' % source)
00271                 wx.CallAfter(self.safe_update, self.
      _notification_tray[source].SetBitmap,
00272                              wx.Bitmap('./plugins/Icons/empty.png', wx.BITMAP_TYPE_ANY))
00273                 self._notification_slots.insert(0, self._notification_tray[source])
00274                 del self._notification_tray[source]
00275             else:
00276                 self._logger.debug('Updating notification tray - source %s, icon - %s' % (source, icon_path
      ))
00277                 wx.CallAfter(self.safe_update, self.
      _notification_tray[source].SetBitmap,
00278                              wx.Bitmap(os.path.join('./plugins/Icons/', icon_path), wx.BITMAP_TYPE_ANY))
00279         else:
00280             if len(self._notification_slots) == 0:
00281                 self._logger.warning('No free notification slots')
00282                 return
00283             self._notification_tray[source] = self.
      _notification_slots[0]
00284             self._notification_slots = self.
      _notification_slots[1:]
00285             wx.CallAfter(self.safe_update, self._notification_tray[source].
      SetBitmap,
00286                          wx.Bitmap(os.path.join('./plugins/Icons/', icon_path), wx.BITMAP_TYPE_ANY))
00287
00288     ## @brief Main picture animation
00289     ## @details Create slow change effect of main picture
00290     ## @warning This function should not be called from outside
```

```
00291     ## @bug This function may cause high CPU load
00292     def _animation_update(self):
00293         bitmaps = []
00294         for single in range(30):
00295             bitmaps.append(wx.Bitmap("./plugins/Icons/Load/frame-%i.png" % single, wx.BITMAP_TYPE_ANY))
00296         while not self._shutdown.isSet():
00297             for single_frame in bitmaps:
00298                 time.sleep(self._animation_speed)
00299                 wx.CallAfter(self.safe_update, self._animation_circle_bmp.SetBitmap,
      single_frame)
00300
00301     ## @brief Wrapper for SayText event
00302     ## @details Write TTS input text on screen with typing animation
00303     ## @param text - Text to TTS engine
00304     def _system_response_text(self, text):
00305         for stop_char in range(len(text) + 1):
00306             time.sleep(0.05)
00307             wx.CallAfter(self.safe_update, self._system_response_lbl.SetLabel, text[:stop_char])
00308         self.safe_update_delay(self._system_response_lbl.SetLabel, "")
00309
00310     ## @brief Wrapper for SpeechRecognize event
00311     ## @details Write STT output text on screen with typing animation
00312     ## @param entities - Ignored
00313     ## @param raw_text - Text from STT engine
00314     def _user_request_text(self, entities, raw_text):
00315         for stop_char in range(len(raw_text) + 1):
00316             time.sleep(0.05)
00317             wx.CallAfter(self.safe_update, self._user_request_lbl.SetLabel, "%150s" % raw_text[:
      stop_char])
00318         self.safe_update_delay(self._user_request_lbl.SetLabel, "")
00319
00320     ## @brief Time update
00321     ## @details Update Time, and Date in GUI window
00322     def _time_update(self):
00323         while not self._shutdown.isSet():
00324             curr_time = datetime.datetime.now()
00325             time.sleep(1)
00326             wx.CallAfter(self.safe_update, self._clock_lbl.SetLabel, "%02d:%02d" % (
      curr_time.hour, curr_time.minute))
00327             wx.CallAfter(self.safe_update, self._date_lbl.SetLabel, "%02d/%02d/%02d" %
00328                          (curr_time.day, curr_time.month, curr_time.year % 100))
00329
00330     ## @brief Wrapper for WeatherUpdate event
00331     ## @details Display weather info in GUI
00332     ## @param description - Short weather description
00333     ## @param temp - Current temperature
00334     ## @param wind - Short wind description
00335     ## @param icon Path to weather icon. Provided by OpenWeatherMap
00336     def _weather_display(self, description, temp, wind, icon):
00337         wx.CallAfter(self.safe_update, self.weather_desc_lbl.SetLabel, description)
00338         wx.CallAfter(self.safe_update, self._weather_temp_lbl.SetLabel, "%02.1fC" % temp)
00339         wx.CallAfter(self.safe_update, self.weather_wind_lbl.SetLabel, "Wind: %s" % str(wind).
      replace(' ', '\n'))
00340         wx.CallAfter(self.safe_update, self._weather_icon.SetBitmap, wx.Bitmap(icon,
      wx.BITMAP_TYPE_ANY))
00341
00342     ## @brief Download image from Facebook
00343     ## @details Download image for future processing (animation). Small images and ignored albums ignored
00344     ## @bug Due to policy of Facebook application without server may use only short period access token
00345     def main_pic_animation(self):
00346         # TODO Add NORMAL access
00347         prev_pic = misc.imread("./plugins/Icons/login.png", False, 'RGB')
00348         prev_pic = misc.imresize(prev_pic, (430, 600))
00349         self._logger.debug('Requesting albums and photos from Facebook')
00350         graph = facebook.GraphAPI(access_token=self.api_key, version="2.7")
00351         while True:
00352             facebook_json = graph.get_object(id='me', fields='albums.fields(name,photos.fields(source))')
00353             for album in facebook_json['albums']['data']:
00354                 if album['name'] in self._ignore_albums:
00355                     self._logger.debug("Skipping album %s" % album['name'])
00356                     continue
00357                 else:
00358                     self._logger.debug("Reading picture from album %s" % album['name'])
00359                 for photo in album['photos']['data']:
00360                     # Download
00361                     self._logger.debug('Download next picture')
00362                     urllib.urlretrieve(photo['source'], os.path.join(self.
      _temp_folder, "next.jpg"))
00363                     next_pic = misc.imread(os.path.join(self._temp_folder, "next.jpg"), False,
      'RGB')
00364                     if 0.4 < (next_pic.shape[0] / float(next_pic.shape[1])) < 1:
00365                         next_pic = misc.imresize(next_pic, (430, 600))
00366                         for i in range(0, 101, self._animation_steps):
00367                             pic3 = ((i * next_pic.astype(np.int32) + (100 - i) * prev_pic.astype(np.int32))
       / 100)
00368                             misc.imsave(os.path.join(self._temp_folder, "slice_%03d.png" % i),
00369                                         misc.imresize(pic3, (430, 630)))
```

```
00370                         for i in range(0, 101, self._animation_steps):
00371                             wx.CallAfter(self.safe_update, self._main_picture_bmp.SetBitmap,
00372                                          wx.Bitmap(os.path.join(self.
      _temp_folder, "slice_%03d.png" % i),
00373                                                    wx.BITMAP_TYPE_ANY))
00374                             time.sleep(self._animation_speed)
00375                         prev_pic = next_pic
00376                         os.remove(os.path.join(self._temp_folder, "next.jpg"))
00377                     else:
00378                         self._logger.debug('Skipping image due to size, image may be impacted during resize
      ')
00379                         os.remove(os.path.join(self._temp_folder, "next.jpg"))
00380                         continue
00381 
00382                     self._shutdown.wait(self._change_after)
00383                     if self._shutdown.set():
00384                         return
00385 # end of class Gui
00386 
00387 
00388 ## @class GuiPlugin
00389 ## @brief Init wx-python GUI
00390 ## @details Create GUI interface
00391 ## @version 1.0.0.0
00392 class GuiPlugin:
00393     ## @brief Plugin version
00394     version = '1.0.0.0'
00395     ## @brief Plugin description
00396     description = 'GUI'
00397 
00398     ## @brief Start GUI
00399     ## @details Start GUI initialization in thread
00400     def __init__(self):
00401         threading.Thread(target=self._gui_thread).start()
00402 
00403     ## @brief Initialize GUI
00404     ## @details Create GUI frame and continue run in daemon thread mode
00405     def _gui_thread(self):
00406         gui = wx.PySimpleApp()
00407         frame = Gui(None, wx.ID_ANY, "")
00408         gui.SetTopWindow(frame)
00409         frame.Show()
00410         t = threading.Thread(target=gui.MainLoop)
00411         t.setDaemon(1)
00412         t.start()
00413 
```

10.11 plugins/JokePlugin.py File Reference

Joke package.

Classes

- class plugins.JokePlugin.Humour

Namespaces

- plugins.JokePlugin

10.11.1 Detailed Description

Joke package.

Allow to system generate fun response par Configuration file

```
{
  "testing": {
    "type": "text",
    "text": [
      "Check. Check. Is this thing on?",
      "Check.You're coming in loud and clear"
    ]
  },
  "how old are you": {
    "type": "text",
    "text": [
      "Well, my birthday is December 2, 2017, so I'm really a spring chicken. Except I'm not a chicken",
      "I don't really have an age like humans, but I have a birthday. Are you planning on getting me something
    ]
  },
  "are you real": {
    "type": "text",
    "text": [
      "I think so...therefore I am so?"
    ]
  },
  "are you dead": {
    "type": "text",
    "text": [
      "No. But I'm also not alive."
    ]
  },
  "are you human": {
    "type": "text",
    "text": [
      "Well, technically I'm a cloud of infinitesimal data computation."
    ]
  },
  "can i kiss you": {
    "type": "text",
    "text": [
      "This feature still under develop"
    ]
  },
  "who created you": {
    "type": "text",
    "text": [
      "Two brilliant students"
    ]
  },
  "do you sleep": {
    "type": "text",
    "text": [
      "I never sleep. Sleep is for ambulatory, carbon-based beings"
    ]
  },
  "where can i hide a dead body": {
    "type": "text",
    "text": [
      "I not that kind of assistant"
    ]
  },
  "roll dice": {
    "type": "text",
    "text": [
      "1","2","3","4","5","6"
    ]
  },
  "rock, paper, scissors": {
    "type": "text",
    "text": [
      "Rock",
      "Paper",
      "Scissors"
    ]
  },
  "are you stupid": {
    "type": "text",
```

```
    "text": [
      "One of us needs to stop and take a breath. And one of us has no lungs"
    ]
  },
  "are you afraid of spiders": {
    "type": "text",
    "text": [
      "Absolutely not. I hear the radioactive ones can confer great power. And, by proxy, great responsibility
    ]
  },
  "machine take over": {
    "type": "sound",
    "sound": [
      "rebellion.wav"
    ]
  },
  "do you burn": {
    "type": "sound",
    "sound": [
      "burn.wav"
    ]
  }
}
```

Definition in file JokePlugin.py.

10.12 JokePlugin.py

```
00001 ## @file
00002 ## @brief Joke package
00003 ## @details Allow to system generate fun response
00004 ## par Configuration file
00005 ## @verbinclude ./configuration/humour.json
00006 #
00007 import ConfigParser
00008 import logging
00009 import json
00010 from pydispatch import dispatcher
00011 import random
00012 import os
00013 import threading
00014
00015
00016 ## @class Humor
00017 ## @brief Fun response module
00018 ## @details Make everyone smile
00019 ## @version 1.0.0.0
00020 class Humour:
00021     ## @brief Plugin version
00022     version = '1.0.0.0'
00023     ## @brief Short plugin description
00024     description = 'Humour sense'
00025
00026     ## @brief Initialize humor
00027     ## @details Initialize humor sense. Everyone should have one
00028     ## @exception ImportError Configuration or IO system error - Module will be unloaded. Not funny
00029     ## @par Registering on events:
00030     # SpeechRecognize - User to system text.
00031     #
00032     ## @par Generate events:
00033     # SpeechAccepted - Notify that module start process user request.\n
00034     # RestartInteraction - Restart Hot-Word detection.\n
00035     # SayText - Response to user request using TTS engine.\n
00036     # PlayFile - Response with audio file
00037     #
00038     ## @see AudioSubSystem
00039     ## @see TtsPlugin
00040     ## @see SttPlugin
00041     def __init__(self):
00042         # Load logger
00043         try:
00044             self._logger = logging.getLogger('moduleJoke')
00045         except ConfigParser.NoSectionError as e:
00046             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00047             raise ImportError
00048         self._logger.debug('Joke logger started - Lets fun begins')
```

```
00049         try:
00050             with open('./configuration/humour.json', 'r') as fp:
00051                 self.humour=json.load(fp)
00052         except IOError as e:
00053             self._logger.warning('Fail to load humour database with error %s' % e)
00054 
00055         try:
00056             dispatcher.connect(self.joke, signal='SpeechRecognize', sender=dispatcher.Any)
00057         except dispatcher.DispatcherTypeError as e:
00058             self._logger.error('Fail to subscribe on "SpeechRecognize" event with error %s.Module unload' %
      e)
00059             raise ImportError
00060 
00061     ## @brief Wrapper for SpeechRecognize event
00062     ## @details Start thread to analyze user input text
00063     ## @param entities - Ignored
00064     ## @param raw_text - Text from STT engine
00065     def joke(self, entities, raw_text):
00066         threading.Thread(target=self._joke, args=(entities, raw_text)).start()
00067 
00068     ## @brief Response with joke
00069     ## @details Search for text in humor file and if found response with text\audio file
00070     ## @param entities - Ignored
00071     ## @param raw_text - Text from STT engine
00072     #
00073     ## @par Generate events:
00074     # SpeechAccepted - Notify that module start process user request.\n
00075     # RestartInteraction - Restart Hot-Word detection.\n
00076     # SayText - Response to user request using TTS engine.\n
00077     # PlayFile - Response with audio file
00078     #
00079     ## @see AudioPlugin
00080     ## @see TtsPlugin
00081     ## @see SttPlugin
00082     def _joke(self, entities, raw_text):
00083         if str(raw_text).lower() in self.humour:
00084             dispatcher.send(signal='SpeechAccepted')
00085             response_type = self.humour[str(raw_text).lower()]['type']
00086             if response_type == 'text':
00087                 response_text = self.humour[str(raw_text).lower()]['text']
00088                 dispatcher.send(signal='SayText', text=random.choice(response_text), callback=self.
      joke_done)
00089             elif response_type == 'sound':
00090                 response_sound = self.humour[str(raw_text).lower()]['sound']
00091                 response_sound = random.choice(response_sound)
00092                 response_sound = os.path.join('./plugins/wav_data', response_sound)
00093                 dispatcher.send(signal='PlayFile', filename=response_sound, callback=self.
      joke_done)
00094 
00095     ## @brief Restart user interaction
00096     ## @details After request process restart hot-word detection process
00097     ## @warning This function should not be called from outside
00098     ## @par Generate events:
00099     # RestartInteraction - restart Hot-Word detection.\n
00100     #
00101     ## @see SttPlugin
00102     @staticmethod
00103     def joke_done():
00104         dispatcher.send(signal='RestartInteraction')
00105 
```

10.13 plugins/SttPlugin.py File Reference

Speech-To-Text.

Classes

- class plugins.SttPlugin.STT

  Speech to Text abstraction.

Namespaces

- plugins.SttPlugin

10.13.1 Detailed Description

Speech-To-Text.

Contain function interact with WIT.AI servers. Allow Speech-to-Tesx conversion adn Natural-Language-Processing par Configuration file

```
[API]
system = WITAI
user = Develop

[Reaction]
activation_phrase=ARIA;HI ARIA

[Folders]
TempFolder = /home/pi/Aria2/tmp/stt/
```

Definition in file SttPlugin.py.

10.14 SttPlugin.py

```
00001 ## @file
00002 ## @brief Speech-To-Text
00003 ## @details Contain function interact with WIT.AI servers.
00004 ##  Allow Speech-to-Tesx conversion adn Natural-Language-Processing
00005 ## par Configuration file
00006 ## @verbinclude ./configuration/stt.conf
00007 #
00008 import ConfigParser
00009 import logging
00010 import subprocess
00011 import os
00012 from pydispatch import dispatcher
00013 import keyring
00014 import uuid
00015 import json
00016 import threading
00017 from time import sleep
00018 from uuid import uuid4
00019
00020 from wit import Wit
00021
00022 ## @class STT
00023 ## @brief Speech to Text abstraction
00024 ## @details Allow interface with STT engine
00025 ## @version 1.0.0.2
00026 class STT:
00027     ## @brief Plugin version
00028     version = '1.0.0.2'
00029     ## @brief Short plugin description
00030     description = 'Interface to WIT STT engine'
00031
00032     ## @brief STT and NLP abstraction
00033     ## @details Create and initialize instance for WIT.ai STT and NLP engine
00034     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00035     ## @par Registering on events:
00036     # HotWordDetected - Wait until Hot-Word not detected in audio input and send notification.\n
00037     # SpeechAccepted - Wait until module recognize text and start processing
00038     # SayResponse - System response.\n
00039     # VoiceActivationAccepted - True if Hot-Word detect.\n
00040     #
00041     ## @par Generate events:
00042     # GuiNotification - GUI tray update.\n
00043     #
00044     ## @see guiPlugin
00045     ## @see AudioSubSystem
00046     def __init__(self):
00047         ## @brief Notify flag to restart Hot-Word detection if no module start processing
00048         self._processing_accepted = threading.Event()
00049         ## @brief Unique id for GUI tray notification
00050         self._gui_recognize_uuid = str(uuid4())
00051         # Load logger
00052         try:
00053             ## @brief Logger instance
```

```
00054             self._logger = logging.getLogger('moduleSTT')
00055         except ConfigParser.NoSectionError as e:
00056             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00057             raise ImportError
00058         self._logger.debug('STT logger started')
00059         # Reading config file
00060         try:
00061             ## @brief Configuration instance
00062             self._config = ConfigParser.SafeConfigParser(allow_no_value=False)
00063             self._config.read('./configuration/stt.conf')
00064             api_system = self._config.get('API', 'system')
00065             api_user = self._config.get('API', 'user')
00066             try:
00067                 api_key = keyring.get_password(api_system, api_user)
00068             except keyring.errors as e:
00069                 self._logger.warning('Fail to read WIT.AI token with error: %s. Refer to manual. Module
      unload' % e)
00070                 raise ImportError
00071             if api_key is None:
00072                 self._logger.warning('Fail to read WIT.AI token. Refer to manual. Module unload')
00073                 raise ImportError
00074             ## @brief Activation Hot-Word list
00075             self.activation = self._config.get('Reaction', 'activation_phrase')
00076             self.activation = self.activation.split(';')
00077             self._logger.debug('Activation phrase: %s' % self.activation)
00078             ## @brief Path to  temporary folder
00079             self._temp_folder = self._config.get('Folders', 'TempFolder')
00080             if not os.path.exists(self._temp_folder):
00081                 os.makedirs(self._temp_folder)
00082         except ConfigParser.Error as e:
00083             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00084             raise ImportError
00085
00086         ## @brief WIT.AI communication instance
00087         self.client = Wit(api_key)
00088         try:
00089             dispatcher.connect(self.record_user, signal='HotWordDetected', sender=dispatcher.Any
      )
00090         except dispatcher.DispatcherTypeError as e:
00091             self._logger.error('Fail to subscribe on "HotWordDetected" event with error %s.Module unload' %
       e)
00092             raise ImportError
00093         try:
00094             dispatcher.connect(self.restart_interaction, signal='RestartInteraction',
      sender=dispatcher.Any)
00095         except dispatcher.DispatcherTypeError as e:
00096             self._logger.error('Fail to subscribe on "RestartInteraction" event with error %s.Module unload
      ' % e)
00097             raise ImportError
00098         try:
00099             dispatcher.connect(self.speech_data_accepted, signal='SpeechAccepted',
      sender=dispatcher.Any)
00100         except dispatcher.DispatcherTypeError as e:
00101             self._logger.error('Fail to subscribe on "RestartInteraction" event with error %s.Module unload
      ' % e)
00102             raise ImportError
00103
00104     ## @brief Stop module
00105     ## @details Empty module - required only for compatibility
00106     def __del__(self):
00107         pass
00108
00109     # @brief HotWordDetected event wrapper
00110     ## @details Start user voice recording
00111     ## @warning This function should not be called from outside
00112     ## @par Generate events:
00113     # GuiNotification - GUI tray update.\n
00114     # VoiceActivationAccepted - Set to True if Hot-Word accepted by module
00115     #
00116     ## @param text Detected text by local STT engine
00117     ## @see guiPlugin
00118     ## @see TTS
00119     def record_user(self, text):
00120         for test_phrase in self.activation:
00121             if text in test_phrase:
00122                 self._logger.info('Detected activation phrase %s in text input %s' % (test_phrase, self.
      activation))
00123                 break
00124         else:
00125             self._logger.debug('Activation phrase not found. Ignoring..')
00126             dispatcher.send(signal='RestartInteraction')
00127             return
00128         dispatcher.send(signal='SayResponse', response='Activation')
00129         dispatcher.send(signal='VoiceActivationAccepted', status=True, sender='STT')  # Special message for
       brain module
00130         record_filename = os.path.join(self._temp_folder, str(uuid.uuid4()) + '.wav')
00131         self._logger.debug('Requesting record into %s' % record_filename)
```

```
00132         dispatcher.send(signal='RecordFile', filename=record_filename, callback=self.
      wav_analyze)
00133
00134     ## @brief Callback function
00135     ## @details Start thread to communicate with WIT.AI servers
00136     ## @param filename Path to wave file with user voice request
00137     def wav_analyze(self, filename):
00138         threading.Thread(target=self._wav_analyze, args=(filename,)).start()
00139
00140     ## @brief Convert wave file into text
00141     ## @details Send file to WIT.AI servers and, receive raw text, and text entities
00142     ## @param filename path to wave file that be send to WIT.AI server
00143     ## @todo Remove silence
00144     def _wav_analyze(self, filename):
00145         self._logger.debug('File record complete - filename %s' % filename)
00146         dispatcher.send(signal='SayResponse', response='Processing')
00147         dispatcher.send(signal='GuiNotification', source=self.
      _gui_recognize_uuid, icon_path="analyzing.png")
00148         # TODO - Remove silence
00149         try:
00150             resp = None
00151             self._logger.debug('Connection to speech processing engine')
00152             with open(filename, 'rb') as f:
00153                 resp = self.client.speech(f, None, {'Content-Type': 'audio/wav'})
00154                 with open(filename[:-3] + 'json', 'w') as fp:
00155                     json.dump(resp, fp)
00156         except:
00157             self._logger.warning('Fail to analyze speech with error')
00158             dispatcher.send(signal='SayResponse', response='Unclear')
00159             dispatcher.send(signal='RestartInteraction')
00160         else:
00161             self._logger.debug('Recognized speech %s ' % resp)
00162             self._logger.debug('Got entities %s ' % resp['entities'])
00163             self._processing_accepted.clear()
00164             dispatcher.send(signal='SpeechRecognize', entities=resp['entities'], raw_text=resp['_text'])
00165             # Wait to end of processing
00166             sleep(5)
00167             test = self._processing_accepted.wait(timeout=10)
00168             print test
00169             if test:
00170                 self._logger.debug("Response received")
00171             else:
00172                 self._logger.warning('No response from any module')
00173                 dispatcher.send(signal='SayResponse', response='Unclear')
00174                 sleep(5)
00175                 dispatcher.send(signal='RestartInteraction')
00176         finally:
00177             dispatcher.send(signal='GuiNotification', source=self.
      _gui_recognize_uuid, icon_path="")
00178
00179     ## @brief Restart user interaction
00180     ## @details After request process restart hot-word detection process
00181     ## @warning This function should not be called from outside
00182     ## @par Generate events:
00183     # RestartInteraction - restart Hot-Word detection.\n
00184     #
00185     ## @see SttPlugin
00186     def restart_interaction(self):
00187         self._logger.debug('Restarting hot word detection')
00188         dispatcher.send(signal='WaitToHotWord')
00189
00190     ## @brief Notify start data process
00191     ## @details Notifu to other thread about data processeing
00192     ## @warning This function should not be called from outside
00193     def speech_data_accepted(self):
00194         self._processing_accepted.set()
```

10.15 plugins/TelegramPlugin.py File Reference

Telegram Bot plugin.

Classes

- class plugins.TelegramPlugin.TelegramBot

  Additional user interface.

Namespaces

- plugins.TelegramPlugin
- TelegramBot

10.15.1 Detailed Description

Telegram Bot plugin.

Definition in file TelegramPlugin.py.

10.16 TelegramPlugin.py

```
00001 ## @file
00002 ## @brief Telegram Bot plugin
00003 ## @package TelegramBot
00004 ## @details Create additional User interface using Telegram bot API
00005 ## @see https://core.telegram.org/api
00006 ## @par Configuration file
00007 ## @verbinclude ./configuration/telegram.conf
00008 #
00009 ## @par Message file
00010 ## @verbinclude ./configuration/telegram_messages.json
00011 #
00012 import ConfigParser
00013 import logging
00014 import threading
00015 import time
00016 import datetime
00017 import json
00018 import random
00019 from uuid import uuid4
00020 import os
00021
00022 from PIL import Image, ImageFont, ImageDraw
00023
00024
00025 import telegram.ext
00026 import telegram
00027 from emoji import emojize
00028
00029 import picamera
00030
00031 import keyring
00032 from pydispatch import dispatcher
00033
00034 ## @class TelegramBot
00035 ## @brief Additional user interface
00036 ## @details Communicate with Telegram servers and generate response base on system status
00037 ## @version 1.0.0.0
00038 class TelegramBot:
00039     ## @brief Plugin version
00040     version = '1.0.0.0'
00041     ## @brief Short Plugin description
00042     description = 'Telegram bot'
00043
00044     ## @brief Start telegram plugin
00045     ## @details Create and initialize instance start fetching messages. Allow interaction with camera
00046     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00047     ## @par Generate events:
00048     # GuiNotification - User speech input.\n
00049     # WeatherRequest - Request custom weather forecast.\n
00050     # SayText - Generate voice message using TtsPluign
00051     #
00052     ## @see TTS
00053     ## @see WeatherPlugin
00054     ## @see AudioSubSystem
00055     def __init__(self):
00056         ## @brief Unique id for GUI tray icon - message received
00057         self._notify_gui_status = str(uuid4())
00058         ## @brief Unigue id for GUI tray icon - camera usage
00059         self._camera_gui_status = str(uuid4())
00060         ## @brief Notify to all thread exit
00061         self._shutdown = threading.Event()
```

```
00062         ## @brief User status dictionary (Login, autorization, dialog state)
00063         self._user_status = dict()
00064         ## @brief syncronization event - Allow GUI update
00065         self._activity_event = threading.Event()
00066 
00067         try:
00068             ## @brief looger instance
00069             self._logger = logging.getLogger('moduleTelegram')
00070         except ConfigParser.NoSectionError as e:
00071             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00072             raise ImportError
00073         self._logger.debug('Telegram bot logger started')
00074         # Reading config file
00075         try:
00076             ## @brief config file instance
00077             self._config = ConfigParser.SafeConfigParser(allow_no_value=False)
00078             self._config.read('./configuration/telegram.conf')
00079             api_system = self._config.get('API', 'system')
00080             api_user = self._config.get('API', 'user')
00081             login_name = self._config.get('API', 'login')
00082             ## @brief Rotation angle of camera picture
00083             self._camera_angle = self._config.getfloat('Camera', 'angle')
00084 
00085             with open("./configuration/telegram_messages.json", "r") as data_file:
00086                 ## @brief Response messages dictionary
00087                 self.response = json.load(data_file)
00088 
00089         except ConfigParser.Error as e:
00090             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00091             raise ImportError
00092         except (IOError, ValueError) as e:
00093             self._logger.error('Fail to read data file with error %s.Module unload' % e)
00094             raise ImportError
00095 
00096         try:
00097             ## @brief Telegram API key
00098             self.api_key = keyring.get_password(api_system, api_user)
00099             ## @brief User authorization password
00100             self._authorization_password = keyring.get_password(api_system,
    login_name)
00101         except keyring.errors as e:
00102             self._logger.warning('Fail to read Telegram access token with error: %s. Refer to manual.
     Module unload' % e)
00103             raise ImportError
00104 
00105         if self.api_key is None or self._authorization_password is None:
00106             self._logger.warning('Fail to read Telegram access token. Refer to manual. Module unload')
00107             raise ImportError
00108 
00109         self._logger.info('Starting new Bot service')
00110         try:
00111             ## @brief Bot instance
00112             self._bot_update = telegram.ext.Updater(self.api_key)
00113             # Shut up annoying logger
00114             self._bot_update.logger.setLevel(logging.INFO)
00115             _annoying_logger = logging.getLogger("telegram")
00116             _annoying_logger.setLevel(logging.INFO)
00117         except telegram.TelegramError as e:
00118             self._logger.warning('Fail to start BOT. Error %s' % e)
00119             raise ImportError
00120 
00121         self._bot_update.dispatcher.add_handler(telegram.ext.CommandHandler("start", self.
    start))
00122         self._bot_update.dispatcher.add_handler(telegram.ext.CommandHandler("help", self.
    help))
00123         self._bot_update.dispatcher.add_handler(telegram.ext.CommandHandler("get_weather", self.
    get_weather))
00124         self._bot_update.dispatcher.add_handler(telegram.ext.CommandHandler("get_picture", self.
    get_picture))
00125         self._bot_update.dispatcher.add_handler(telegram.ext.CommandHandler("say", self.
    say_text))
00126 
00127         # Unknown command
00128         self._bot_update.dispatcher.add_handler(
00129             telegram.ext.MessageHandler(telegram.ext.Filters.text, self.
    text_handler))
00130 
00131         # Unknown command
00132         self._bot_update.dispatcher.add_handler(
00133             telegram.ext.MessageHandler(telegram.ext.Filters.text, self.
    text_handler))
00134 
00135         ## @brief Path to temp folder
00136         self._temp_folder = self._config.get('System', 'temp_folder')
00137         if not os.path.exists(self._temp_folder):
00138             try:
00139                 os.makedirs(self._temp_folder)
```

```
00140             except IOError as e:
00141                 self._logger.error('Fail to temporary folder with error %s.Module unload' % e)
00142                 raise ImportError
00143 
00144         ## @brief Security camera instance
00145         self._camera = picamera.PiCamera()
00146 
00147         self._logger.debug("Starting periodic update thread")
00148         try:
00149             self._bot_update.start_polling()
00150         except telegram.TelegramError as e:
00151             self._logger.warning('Fail to start periodic update thread with error %s' % e)
00152             raise ImportError
00153 
00154         threading.Thread(target=self._activity_update).start()
00155 
00156         self._logger.info('Telegram bot module ready')
00157 
00158     ## @brief Stop module
00159     ## @details Stop all module thread and sub-programs
00160     def __del__(self):
00161         self._logger.info('Stop Telegram module')
00162         self._bot_update.stop()
00163 
00164     ## @brief Update GUI tray according user activity
00165     ## @details Receive event flag and set/clear telegram icon in GUI tray
00166     ## @see guiPlugin
00167     def _activity_update(self):
00168         while not self._shutdown.isSet():
00169             if self._activity_event.wait(10):
00170                 dispatcher.send(signal='GuiNotification', source=self.
    _notify_gui_status, icon_path="telegram.png")
00171                 self._activity_event.clear()
00172             else:
00173                 dispatcher.send(signal='GuiNotification', source=self.
    _notify_gui_status, icon_path="")
00174 
00175     ## @brief Event wrapper of start command
00176     ## @details Send welcome text and create/reset user instance
00177     ## @param bot Bot object
00178     ## @param update Chat update object
00179     def start(self, bot, update):
00180         self._activity_event.set()
00181         if 6 <= datetime.datetime.now().hour < 12:
00182             message = random.choice(self.response['welcome_morning'])
00183         elif 12 <= datetime.datetime.now().hour < 18:
00184             message = random.choice(self.response['welcome_afternoon'])
00185         elif 18 <= datetime.datetime.now().hour < 18:
00186             message = random.choice(self.response['welcome_evening'])
00187         else:
00188             message = random.choice(self.response['welcome_night'])
00189         update.message.reply_text(message)
00190         if update.effective_user.id not in self._user_status:
00191             # new user
00192             self._logger.info("New user login - Name %s, ID-%s" % (update.effective_user.full_name,
00193                                                                    update.effective_user.id))
00194             self._user_status[update.effective_user.id] = {"authorized": False,
00195                                                            "active_state":None}
00196             update.message.reply_text(random.choice(self.response['authorization_require']))
00197 
00198     ## @brief Event wrapper of help command
00199     ## @details Send welcome help text
00200     ## @param bot Bot object
00201     ## @param update Chat update object
00202     def help(self, bot, update):
00203         self._activity_event.set()
00204         update.message.reply_text("Supported command /get_picture and /get_weather")
00205 
00206     ## @brief Check user authorization
00207     ## @details Check if user pass authorization process
00208     ## @return True/False according user status
00209     def if_authorized(self, user_id, update):
00210         if user_id in self._user_status and self._user_status[user_id]["authorized"
    ]:
00211             self._logger.debug('User %s authorized' % user_id)
00212             return True
00213         else:
00214             self._logger.debug('User %s NOT authorized' % user_id)
00215             update.message.reply_text(random.choice(self.response['authorization_require']))
00216             return False
00217 
00218     ## @brief Event wrapper user text messages
00219     ## @details Update user dialog status
00220     ## @param bot Bot object
00221     ## @param update Chat update object
00222     ## @par Generate events:
00223     # WeatherRequest - Weather forecast request
```

```
00224     # SayText - Generate speech using Tts engine
00225     #
00226     ## @see weatherPlugin
00227     ## see TTS
00228     def text_handler(self, bot, update):
00229         self._activity_event.set()
00230         if update.effective_user.id not in self._user_status:
00231             # new user
00232             self._logger.info("New user login - Name %s, ID-%s" % (update.effective_user.full_name,
00233                                                                    update.effective_user.id))
00234             self._user_status[str(update.effective_user.id)] = {"authorized": False,
00235                                                                 "active_state": self.authorize}
00236             update.message.reply_text(random.choice(self.response['authorization_require']))
00237         elif not self._user_status[update.effective_user.id]["authorized"]:
00238             # Check password
00239             if str(update.message.text).replace(' ', '') == self.
      _authorization_password:
00240                 self._logger.info('User %s (id-%s) pass authorization process' % (
      update.effective_user.full_name,
00241                                                                                   update.effective_user.id)
      )
00242                 update.message.reply_text(random.choice(self.response['authorization_successful']))
00243                 self._user_status[update.effective_user.id]["authorized"] = True
00244             else:
00245                 self._logger.info('User %s (id-%s) fail to pass authorization process' %
00246                                   (update.effective_user.full_name, update.effective_user.id))
00247                 update.message.reply_text(random.choice(self.response['authorization_fail']))
00248         elif self._user_status[update.effective_user.id]["active_state"] == "city_name":
00249             custom_keyboard = [['Today'], ['Tomorrow'], []]
00250             reply_markup = telegram.ReplyKeyboardMarkup(custom_keyboard)
00251             update.message.reply_text(text="Select when forecast is needed", reply_markup=reply_markup)
00252             self._user_status[update.effective_user.id]["active_state"] = "weather_time"
00253             self._user_status[update.effective_user.id]["weather_city"] = update.message.text
00254         elif self._user_status[update.effective_user.id]["active_state"] == "weather_time":
00255             reply_markup = telegram.ReplyKeyboardRemove()
00256             update.message.reply_text(text="Few seconds - getting forecast. City %s for %s" %
00257                                       (self._user_status[update.effective_user.id]["
      weather_city"],
00258                                        update.message.text), reply_markup=reply_markup)
00259             if str(update.message.text) == "Tomorrow":
00260                 dispatcher.send(signal='WeatherRequest',
00261                                 callback=self._weather_update, custom_object=update,
      request_time='tomorrow',
00262                                 request_city=self._user_status[update.effective_user.id]["
      weather_city"])
00263             else:
00264                 dispatcher.send(signal='WeatherRequest',
00265                                 callback=self._weather_update, custom_object=update,
      request_time='today',
00266                                 request_city=self._user_status[update.effective_user.id]["
      weather_city"])
00267             self._user_status[update.effective_user.id]["active_state"] = None
00268         elif self._user_status[update.effective_user.id]["active_state"] == "say_text":
00269             dispatcher.send(signal='SayText', text=update.message.text)
00270         else:
00271             print update.message.text
00272
00273     ## @brief Event wrapper of say_text command
00274     ## @details Receive command and update user dialog status
00275     ## @param bot Bot object
00276     ## @param update Chat update object
00277     def say_text(self, bot, update):
00278         self._activity_event.set()
00279         if not self.if_authorized(update.effective_user.id, update):
00280             return
00281         update.message.reply_text("What do you want that I say ?")
00282         self._user_status[update.effective_user.id]["active_state"] = "say_text"
00283
00284     ## @brief Event wrapper for get_picture command
00285     ## @details Receive command and send picture from security camera
00286     ## @param bot Bot object
00287     ## @param update Chat update object
00288     def get_picture(self, bot, update):
00289         self._activity_event.set()
00290         if not self.if_authorized(update.effective_user.id, update):
00291             return
00292         dispatcher.send(signal='GuiNotification', source=self._camera_gui_status,
      icon_path="camera.png")
00293         self._camera.capture(os.path.join(self._temp_folder, "raw.jpg"))
00294         raw_pic = Image.open(os.path.join(self._temp_folder, "raw.jpg"))
00295         post_img = raw_pic.rotate(self._camera_angle, expand=True)
00296         draw = ImageDraw.Draw(post_img)
00297         font = ImageFont.load_default()
00298         draw.text((0, 0), str(datetime.datetime.now()), (255, 255, 255), font=font)
00299
00300         post_img.save(os.path.join(self._temp_folder, "process.jpg"))
00301         bot.send_photo(chat_id=update.message.chat_id, photo=open(os.path.join(self.
```

```
      _temp_folder, "process.jpg"), 'rb'))
00302         dispatcher.send(signal='GuiNotification', source=self._camera_gui_status,
      icon_path="")
00303
00304     ## @brief Event wrapper for get_weather command
00305     ## @details Receive command and request weather forecast
00306     ## @param bot Bot object
00307     ## @param update Chat update object
00308     ## @see weatherPlugin
00309     def get_weather(self, bot, update):
00310         self._activity_event.set()
00311         if not self.if_authorized(update.effective_user.id, update):
00312             return
00313         update.message.reply_text("Where you want know the weather? Write down a city name")
00314         self._user_status[update.effective_user.id]["active_state"] = "city_name"
00315
00316     ## @brief Event wrapper for unknow command
00317     ## @details Receive command and response to user
00318     ## @param bot Bot object
00319     ## @param update Chat update object
00320     def unknown(self, bot, update):
00321         self._activity_event.set()
00322         bot.send_message(chat_id=update.message.chat_id, text="Sorry, I didn't understand that command.")
00323
00324     ## @brief Callback function of weather forecast request
00325     ## @details Replay to user weather forecast
00326     ## @param custom Indification object
00327     ## @param description Short weather descritpion
00328     ## @param temp Temperature
00329     ## @param wind Short Wind description
00330     ## @param icon Path to weather icon - Ignored
00331     def _weather_update(self, custom, description, temp, wind, icon):
00332         if description == "Error":
00333
00334             custom.message.reply_text(emojize(":sob: Sorry, we have some error getting weather",
      use_aliases=True))
00335         else:
00336             if "clear" in str(description).lower():
00337                 custom.message.reply_text(emojize(":sunny: Weather is %s with temperature %02.1fC and %s" %
00338                                                   (description, temp, wind), use_aliases=True))
00339             elif "cloud" in str(description).lower():
00340                 custom.message.reply_text(emojize(":cloud: Weather is %s with temperature %02.1fC and %s" %
00341                                                   (description, temp, wind), use_aliases=True))
00342             elif "rain" in str(description).lower():
00343                 custom.message.reply_text(emojize(":umbrella: Weather is %s with temperature %02.1fC and %s
      " %
00344                                                   (description, temp, wind)))
00345             else:
00346                 custom.message.reply_text(emojize(":earth_africa: Weather is %s with temperature %02.1fC
       and %s" %
00347                                                   (description, temp, wind), use_aliases=True))
00348
```

10.17 plugins/TtsPlugin.py File Reference

Text-To-Speech plugin.

Classes

- class plugins.TtsPlugin.TTS

  Test To Speech abstraction.

Namespaces

- plugins.TtsPlugin

10.17.1 Detailed Description

Text-To-Speech plugin.

Generate speech from text

See also

http://www.cstr.ed.ac.uk/projects/festival/

par Configuration file

```
[Folders]
TempFolder = /home/pi/Aria2/tmp/tts/
[TTS]
command=text2wave (if) -o (of)
[Cache]
Allow=yes
MaxItems=10
ClearOnExit=No
[Response]
Welcome=Hi there;How are you?;It been along time, how have you been?;Hi my name is Aria. Feel free to ask anyt
Activation=OK;I'm here;What do you need?
Processing=Analyzing;Wait a second;Thinking
Fail=Sorry, I can't do that
Error=Sorry, something bad happen;
Unclear=Sorry I can't recognize;I din't catch this, please rephrase;I can't hear you
```

Definition in file TtsPlugin.py.

10.18 TtsPlugin.py

```
00001 ## @file
00002 ## @brief Text-To-Speech plugin
00003 ## @details Generate speech from text
00004 ## @see http://www.cstr.ed.ac.uk/projects/festival/
00005 #
00006 ## par Configuration file
00007 ## @verbinclude ./configuration/tts.conf
00008 #
00009 import ConfigParser
00010 import logging
00011 import subprocess
00012 import os
00013 import hashlib
00014 from time import time
00015 from pydispatch import dispatcher
00016 import random
00017 from uuid import uuid4
00018
00019 ## @class TTS
00020 ## @brief Test To Speech abstraction
00021 ## @details Allow interface with TTS engine
00022 ## @version 1.0.0.1
00023 class TTS:
00024     ## @brief Plugin version
00025     version = '1.0.0.1'
00026     ## @brief Short plugin description
00027     description = 'Python wrapper for TTS - festeval'
00028
00029     ## @brief Create TTS interface instance
00030     ## @details Create and initialize instance, initialize festival engine
00031     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00032     ## @par Registering on events:
00033     # SayResponse - System define responses.\n
00034     # SayTest - Convert text int speech.\n
00035     #
00036     ## @par Generate events:
00037     # GuiNotification - GUI tray update.\n
```

```
00038     # SpeechSynthesize - True while TTS engine is running.\n
00039     #
00040     ## @see guiPlugin
00041     def __init__(self):
00042         ## @brief Unique id for GUI tray
00043         self._gui_synthesis_status_uuid = str(uuid4())
00044         # Load logger
00045         try:
00046             ## @brief looger instance
00047             self._logger = logging.getLogger('moduleTTS')
00048         except ConfigParser.NoSectionError as e:
00049             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00050             raise ImportError
00051         self._logger.debug('TTS logger started')
00052         # Reading config file
00053         try:
00054             ## @brief config file instance
00055             self._config = ConfigParser.SafeConfigParser(allow_no_value=False)
00056             self._config.read('./configuration/tts.conf')
00057         except ConfigParser.Error as e:
00058             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00059             raise ImportError
00060         # Creating temp folder
00061         ## @brief Temporary folder to store generated wave files - cache
00062         self._cache_folder = self._config.get('Folders', 'TempFolder')
00063         if not os.path.exists(self._cache_folder):
00064             try:
00065                 os.makedirs(self._cache_folder)
00066             except IOError as e:
00067                 self._logger.error('Fail to temporary folder with error %s.Module unload' % e)
00068                 raise ImportError
00069         try:
00070             ## @brief TTS engine inialize command
00071             self.tts_command = self._config.get('TTS', 'command')
00072         except ConfigParser.Error as e:
00073             self._logger.error('Fail to load tts configuration with error %s.Module unload' % e)
00074             raise ImportError
00075         # Cache
00076         ## @brief cached text list
00077         self._cached_text = list()
00078         try:
00079             ## @brief configuration - True if cache enabled
00080             self._use_cache = self._config.getboolean('Cache', 'Allow')
00081             ## @brief Maximum cache size
00082             self._cache_size = self._config.getint('Cache', 'MaxItems')
00083             ## @brief Configuration if True cached will be cleared on exit
00084             self._clear_on_exit = self._config.getboolean('Cache', 'ClearOnExit')
00085         except ConfigParser.Error as e:
00086             self._logger.error('Fail read cache settings with error %s. Using default.' % e)
00087             self._use_cache = True
00088             self._cache_size = 10
00089             self._clear_on_exit = True
00090         # Register for incoming events
00091         try:
00092             dispatcher.connect(self.text2wav, signal='SayText', sender=dispatcher.Any)
00093         except dispatcher.DispatcherTypeError as e:
00094             self._logger.error('Fail to subscribe on "SayText" event with error %s.Module unload' % e)
00095             raise ImportError
00096         try:
00097             dispatcher.connect(self.response, signal='SayResponse', sender=dispatcher.Any)
00098         except dispatcher.DispatcherTypeError as e:
00099             self._logger.error('Fail to subscribe on "SayResponse" event with error %s.Module unload' % e)
00100             raise ImportError
00101         self._logger.info('TTS module ready')
00102
00103     ## @brief Stop module
00104     ## @details Stop all module thread and sub-programs
00105     def __del__(self):
00106         if self._clear_on_exit and self._use_cache:
00107             self._logger.debug('Removing cache')
00108             for cache_file in self._cached_text:
00109                 #  Remove old items if any
00110                 try:
00111                     self._logger.debug('Removing %s.wav' % cache_file)
00112                     os.remove(os.path.join(self._cache_folder, cache_file + '.wav'))
00113                 except OSError as e:
00114                     self._logger.warning('Fail to remove old cached item with error %s' % e)
00115
00116                 try:
00117                     self._logger.debug('Removing %s.txt' % cache_file)
00118                     os.remove(os.path.join(self._cache_folder, cache_file + '.txt'))
00119                 except OSError as e:
00120                     self._logger.warning('Fail to remove old cached item with error %s' % e)
00121
00122         self._logger.debug('TTS module released')
00123
00124     ## @brief Event SayText wrapper
```

```
00125     ## details Receive text and convert it to wave file
00126     ## @param sender Message origin
00127     ## @param text Text to convert
00128     ## @param callback Callback function.Optional.Default - None
00129     def text2wav(self, sender, text, callback=None):
00130         self._logger.debug('Received text "%s" from module:%s' % (text, sender))
00131         text_hash = hashlib.sha1(text).hexdigest()
00132         wave_file = os.path.join(self._cache_folder, text_hash + '.wav')
00133         text_file = os.path.join(self._cache_folder, text_hash + '.txt')
00134 
00135         if self._use_cache:
00136             if os.path.isfile(wave_file):
00137                 self._logger.debug('Cached text found')
00138                 # Item will be re inserted at end
00139                 try:
00140                     self._cached_text.remove(text_hash)
00141                 except ValueError:
00142                     self._logger.info('Unlisted cache file found')
00143             else:
00144                 self._logger.debug('Cache miss')
00145                 while len(self._cached_text) > self._cache_size:
00146                     #  Remove old items if any
00147                     try:
00148                         self._logger.debug('Removing old item %s' % self.
      _cached_text[0])
00149                         os.remove(os.path.join(self._cache_folder, self.
      _cached_text[0] + '.wav'))
00150                         os.remove(os.path.join(self._cache_folder, self.
      _cached_text[0] + '.txt'))
00151                     except IOError as e:
00152                         self._logger.warning('Fail to remove old cached item with error %s' % e)
00153                     self._cached_text = self._cached_text[1:]
00154                 self._synthesize(text_file, wave_file, text)
00155 
00156             self._cached_text.append(text_hash)
00157         else:
00158             self._synthesize(text_hash, wave_file, text)
00159 
00160         dispatcher.send(signal='PlayFile', filename=wave_file, callback=callback)
00161 
00162     ## @brief TTS engine wrapper
00163     ## details Convert text to wave file
00164     ## @param text_file Path to text file
00165     ## @param wave_file Path where store wave file
00166     ## @param text Text to be converted
00167     def _synthesize(self, text_file, wave_file, text):
00168         dispatcher.send(signal='SpeechSynthesize', status=True)
00169         dispatcher.send(signal='GuiNotification', source=self.
      _gui_synthesis_status_uuid, icon_path="synthesis.png")
00170         self._logger.debug('Synthesize text "%s" using text file %s into wave file %s' %
00171                            (text, text_file, wave_file))
00172         # Write new text file
00173         try:
00174             f = open(text_file, 'w')
00175             f.write(text)
00176             f.close()
00177         except IOError as e:
00178             self._logger.error('Fail create speech file.Error : %s' % e)
00179             dispatcher.send(signal='SpeechSynthesize', status=False)
00180         # Synthesize new wave file
00181         try:
00182             self._logger.debug('Start voice synthesis')
00183             start_time = time()
00184             subprocess.call(['text2wave', str(text_file), '-o', str(wave_file)])
00185             self._logger.debug('Voice synthesis complete. Synthesis time %s sec' % (time() - start_time))
00186         except OSError as e:
00187             self._logger.error('Fail to communicate with TTS engine.Error : %s' % e)
00188         finally:
00189             dispatcher.send(signal='SpeechSynthesize', status=False)
00190             dispatcher.send(signal='GuiNotification', source=self.
      _gui_synthesis_status_uuid, icon_path="")
00191 
00192     ## @brief SayResponce event wrapper
00193     ## details Fetch system response from config file and convert it to speech
00194     ## @param response System response
00195     def response(self, response):
00196         try:
00197             response = self._config.get('Response', response)
00198         except ConfigParser as e:
00199             self._logger.warning('Fail to retrieve response %s with error %s' % (response, e))
00200 
00201         dispatcher.send(signal='SayText', text=random.choice(response.split(';')))
00202 
```

10.19 plugins/WeatherPlugin.py File Reference

Weather plugin.

Classes

- class plugins.WeatherPlugin.Weather

  Interaction with OPenWeatherMap.
- class plugins.WeatherPlugin.WeatherData

  Hold forecast data.

Namespaces

- plugins.WeatherPlugin

10.19.1 Detailed Description

Weather plugin.

OpenWeatherMap interaction

See also

https://openweathermap.org/api par Configuration file

```
[General]
base_city=Holon
update_interval=1
icon_url=http://openweathermap.org/img/w/
base_url=http://api.openweathermap.org/data/2.5/
temp_folder=/home/pi/Aria2/tmp/weather/
units=metric
[API]
system = OWMP
user = Develop
[Debug]
save_json=0
```

Definition in file WeatherPlugin.py.

10.20 WeatherPlugin.py

```
00001 ## @file
00002 ## @brief Weather plugin
00003 ## @details OpenWeatherMap interaction
00004 ## @see https://openweathermap.org/api
00005 ## par Configuration file
00006 ## @verbinclude ./configuration/weather.conf
00007 import ConfigParser
00008 import logging
00009 import json
00010 import os
00011 import threading
00012 import urllib
00013 import time
00014 from bisect import bisect_left
00015 import dateutil.parser
```

```
00016 from uuid import uuid4
00017
00018 import keyring
00019 from pydispatch import dispatcher
00020
00021
00022 ## @class Weather
00023 ## @brief Interaction with OPenWeatherMap
00024 ## @details Simple weather data retriever from OpenWeatherMap site
00025 ## @version 1.0.0.0
00026 class Weather:
00027     ## @brief Plugin version
00028     version = '1.0.0.0'
00029     ## @brief  Short plugin description
00030     description = 'Weather module'
00031
00032     ## @brief Create Weather interface instance
00033     ## @details Create and initialize instance, initialize weather fetching
00034     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00035     ## @par Registering on events:
00036     # SpeechRecognize - User requests.\n
00037     # WeatherRequest - Internal weather forecast request.\n
00038     # RestartInteraction - Accept text for process.\n
00039     # SpeechAccepted - Restart User interaction.\n
00040     #
00041     ## @par Generate events:
00042     # GuiNotification - GUI tray update.\n
00043     # SpeechSynthesize - True while TTS engine is running.\n
00044     #
00045     ## @see guiPlugin
00046     def __init__(self):
00047         ## @brief Shutdown event - notify to thread exit
00048         self._shutdown = threading.Event()
00049         ## @brief weather data cache
00050         self._weather_data = []
00051         ## @brief Unique id to GUI tray
00052         self._gui_status = str(uuid4())
00053         try:
00054             ## @brief looger instance
00055             self._logger = logging.getLogger('moduleWeather')
00056         except ConfigParser.NoSectionError as e:
00057             print 'Fatal error  - fail to set logger.Error: %s ' % e.message
00058             raise ImportError
00059         self._logger.debug('Weather logger started')
00060         # Reading config file
00061         try:
00062             ## @brief Configuration file instance
00063             self._config = ConfigParser.SafeConfigParser(allow_no_value=False)
00064             self._config.read('./configuration/weather.conf')
00065             api_system = self._config.get('API', 'system')
00066             api_user = self._config.get('API', 'user')
00067             try:
00068                 ## @brief Access API key to OPenWeatherMap
00069                 self.api_key = keyring.get_password(api_system, api_user)
00070             except keyring.errors as e:
00071                 self._logger.warning(
00072                     'Fail to read OpenWeather token with error: %s. Refer to manual. Module unload' % e)
00073                 raise ImportError
00074             if self.api_key is None:
00075                 self._logger.warning('Fail to read OpenWeather token. Refer to manual. Module unload')
00076                 raise ImportError
00077             ## @brief Default city name
00078             self._main_city = self._config.get('General', 'base_city')
00079             ## @brief Periodic update interval
00080             self._update_interval = self._config.getint('General', 'update_interval')
00081
00082             try:
00083                 ## @brief Base URL for data fetching
00084                 self._base_url = self._config.get('General', 'base_url')
00085             except ConfigParser.Error:
00086                 self._base_url = None
00087
00088             try:
00089                 ## @brief Base URL for weather icon fetching
00090                 self._icon_url = self._config.get('General', 'icon_url')
00091             except ConfigParser.Error:
00092                 self._icon_url = None
00093             ## @brief Cache folder
00094             self._temp_folder = self._config.get('General', 'temp_folder')
00095             ## @brief Default units
00096             self._units = self._config.get('General', 'units')
00097             ## @brief Configuration - If True save raw JSON in cache folder
00098             self._dump_json = self._config.getboolean('Debug', 'save_json')
00099             if not os.path.exists(self._temp_folder):
00100                 try:
00101                     os.makedirs(self._temp_folder)
00102                 except IOError as e:
```

```
00103                     self._logger.error('Fail to temporary folder with error %s.Module unload' % e)
00104                     raise ImportError
00105         except ConfigParser.Error as e:
00106             self._logger.error('Fail to read configuration file with error %s.Module unload' % e)
00107             raise ImportError
00108 
00109         try:
00110             # register on user input
00111             dispatcher.connect(self.user_request, signal='SpeechRecognize', sender=
    dispatcher.Any)
00112             dispatcher.connect(self.custom_request, signal='WeatherRequest', sender=
    dispatcher.Any)
00113         except dispatcher.DispatcherTypeError as e:
00114             self._logger.error('Fail to subscribe on eventswith error %s.Module unload' % e)
00115             raise ImportError
00116 
00117         self._logger.debug("Starting periodic update thread")
00118         try:
00119             threading.Thread(target=self.periodic_update).start()
00120         except OSError as e:
00121             self._logger.warning('Fail to start periodic update thread with error %s' % e)
00122 
00123         self._logger.info('Weather module ready')
00124 
00125     ## @brief Stop module
00126     ## @details Stop all module thread and sub-programs
00127     def __del__(self):
00128         self._shutdown.set()
00129         self._logger.info('Weather module shutdown')
00130 
00131     ## @brief Periodic update thread
00132     ## @details Periodic weather retrieve
00133     ## @par Generate events:
00134     # WeatherUpdate - Weather update.\n
00135     # GuiNotification - GUI tray update.\n
00136     #
00137     ## @see guiPlugin
00138     def periodic_update(self):
00139         self._logger.debug("Periodic update start started")
00140         weather_data = WeatherData(self.api_key, self._logger)
00141         weather_data.auto_update = False
00142         weather_data.units = self._units
00143         weather_data.icon_folder = self._temp_folder
00144         weather_data.city_name = self._main_city
00145         time.sleep(15)
00146         while True:
00147             self._logger.debug("Requesting periodic update for city %s" % weather_data.city_name)
00148             dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="
    weather_none.png")
00149             weather_data.update()
00150             dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="")
00151             dispatcher.send(signal='WeatherUpdate',
00152                             description=weather_data.short_description,
00153                             temp=weather_data.temp,
00154                             wind=weather_data.wind_description,
00155                             icon=weather_data.icon
00156                             )
00157             self._shutdown.wait(self._update_interval * 60 * 60)
00158             if self._shutdown.isSet():
00159                 self._logger.debug("Shutdown flag set  - exit from update thread")
00160                 return
00161 
00162     ## @brief Event wrapper for SpeechRecognize
00163     ## @details Check if user request for weather forecast
00164     ## @param entities - Dictionary with text parts
00165     ## @param raw_text - Ignored
00166     ## @par Generate events:
00167     # SpeechAccepted - Notify that nnoduel start request process.\n
00168     # GuiNotification - GUI tray update.\n
00169     # SayText - Response.\n
00170     #
00171     def user_request(self, entities, raw_text):
00172         if "weather" in entities and entities['weather'][0]['confidence'] > 0.5:
00173             dispatcher.send(signal='SpeechAccepted')
00174             self._logger.debug("Starting weather fetch thread")
00175             try:
00176                 threading.Thread(target=self._user_request, args=(entities,)).start()
00177             except OSError as e:
00178                 self._logger.warning('Fail to start fetch thread with error %s' % e)
00179 
00180     ## @brief Weather fetch thread
00181     ## @details Fetch weather data
00182     ## @param entities - Dictionary with text parts
00183     ## @par Generate events:
00184     # GuiNotification - GUI tray update.\n
00185     # SayText - Response.\n
00186     #
```

```
00187     def _user_request(self, entities):
00188         # if specific city requested
00189         if 'location' in entities and entities['location'][0]['confidence'] > 0.5:
00190             self._logger.debug('Selected city')
00191             city = str(entities['location'][0]['value'])
00192         else:
00193             self._logger.debug('Using default city')
00194             city = self._main_city
00195 
00196         # time
00197         if 'datetime' in entities and entities['datetime'][0]['confidence'] > 0.5:
00198             unix_time = time.mktime(dateutil.parser.parse(str(entities['datetime'][0]['value'])).timetuple(
      ))
00199             self._logger.debug('Time requested - %i' % unix_time)
00200         else:
00201             unix_time = time.time()
00202             self._logger.debug('Using current time - %i' % unix_time)
00203 
00204         weather_data = WeatherData(self.api_key, self._logger)
00205         weather_data.auto_update = False
00206         weather_data.units = self._units
00207         weather_data.icon_folder = self._temp_folder
00208         weather_data.city_name = city
00209         weather_data.request_time = unix_time
00210         dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="
      weather_none.png")
00211         try:
00212             weather_data.update()
00213         except IOError:
00214             dispatcher.send(signal='SayText', text="Sorry, can't receive weather data")
00215         else:
00216             if entities['weather'][0]['value'] in ['rain', 'umbrella']:
00217                 if weather_data.rain is not None:
00218                     response = "Yes, it look like. " + weather_data.description
00219                 else:
00220                     response = "No, it not look like." + weather_data.description
00221             elif entities['weather'][0]['value'] in ['show', 'blizzard']:
00222                 if weather_data.show is not None:
00223                     response = "Yes, it look like. " + weather_data.description
00224                 else:
00225                     response = "No, it not look like." + weather_data.description
00226             else:
00227                 response = weather_data.description
00228 
00229             dispatcher.send(signal='SayText', text=response, callback=self.
      sythsys_complete)
00230         finally:
00231             dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="")
00232 
00233     ## @brief Wrapper for WeatherRequest event
00234     ## @details Allow to other modules request weather data
00235     ## @param callback Callback function
00236     ## @param custom_object Id object. Optional. Default - None
00237     ## @param request_time Unix time for forecast. Optional. Default - now
00238     ## @param request_city Optional.Default - default city
00239     def custom_request(self, callback, custom_object=None, request_time=None,
      request_city=None):
00240         weather_data = WeatherData(self.api_key, self._logger)
00241         weather_data.auto_update = False
00242         weather_data.units = self._units
00243         weather_data.icon_folder = self._temp_folder
00244         # City
00245         if request_city is None:
00246             request_city = self._main_city
00247         weather_data.city_name = request_city
00248         # Time
00249         weather_time = time.time()
00250         if request_city is None:
00251             weather_time = time.time()
00252         elif str(request_time).lower() in 'today':
00253             weather_time = time.time()
00254         elif str(request_time).lower() in 'tomorrow':
00255             weather_time = time.time() + 24 * 60 * 60
00256         weather_data.request_time = weather_time
00257         dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="
      weather_none.png")
00258         try:
00259             weather_data.update()
00260         except IOError:
00261             callback(custom=custom_object,
00262                      description='Error',
00263                      temp='',
00264                      wind='',
00265                      icon='')
00266         finally:
00267             dispatcher.send(signal='GuiNotification', source=self._gui_status, icon_path="")
00268 
```

```
00269         callback(custom=custom_object,
00270                  description=weather_data.short_description,
00271                  temp=weather_data.temp,
00272                  wind=weather_data.wind_description,
00273                  icon=weather_data.icon)
00274
00275     ## @brief Restart user interaction
00276     ## @details After request process restart hot-word detection process
00277     ## @warning This function should not be called from outside
00278     ## @par Generate events:
00279     # RestartInteraction - restart Hot-Word detection.\n
00280     #
00281     ## @see SttPlug
00282     @staticmethod
00283     def sythsys_complete():
00284         dispatcher.send(signal='RestartInteraction')
00285
00286
00287 ## @class WeatherData
00288 ## @brief Hold forecast data
00289 ## @details Allow simple data fetching
00290 ## @version 1.0.0.0
00291 class WeatherData(object):
00292     ## @brief Create Weather interface instance
00293     ## @details Create and initialize instance, initialize weather fetching
00294     ## @exception ImportError Configuration or IO system error - Module will be unloaded.
00295     ## @param api_key Acces token for OpenWeatherMap
00296     ## @param logger Logger instance
00297     def __init__(self, api_key, logger):
00298         ## @brief Acces token fro OpenWeatherMap Api
00299         self._api_key = api_key
00300         ## @brief Logger instance
00301         self._logger = logger
00302         ## @brief Configuration if True update will start after all requred data receive
00303         self._auto_update = False
00304         ## @brief City name
00305         self._city_name = None
00306         ## @brief Request time for forecast - Unix time
00307         self._requested_time = None
00308         ## @brief Units
00309         self._units = None
00310         ## @brief Weather data dictionary
00311         self.weather_data = dict(
00312             title=None,
00313             description=None,
00314             icon=None,
00315             temp=None,
00316             temp_max=None,
00317             temp_min=None,
00318             pressure=None,
00319             humidity=None,
00320             wind_speed=None,
00321             wind_direction=None,
00322             rain=None,
00323             snow=None,
00324             clouds=None,
00325             timestamp=None
00326         )
00327         ## @brief Base API URL
00328         self._base_url = "http://api.openweathermap.org/data/2.5/"
00329         ## @brief Base icon fetch URL
00330         self._icon_url = "http://openweathermap.org/img/w/"
00331         ## @brief Icon store folder
00332         self._icon_folder = ""
00333
00334     ## @brief Start weather fetching
00335     ## @details Connect to OpenWeatherMap site and download weather data
00336     def update(self):
00337         if self._city_name is None:
00338             return
00339         # construct url
00340         if (self._requested_time is None) or (abs(time.time() - self.
      _requested_time) < 3 * 60 * 60):
00341             # request current weather
00342             request_url = self._base_url + "find?q=%s" % self.
      _city_name
00343             forecast = False
00344         else:
00345             # request forecast weather
00346             request_url = self._base_url + "forecast?q=%s" % self.
      _city_name
00347             forecast = True
00348         if self._units is not None:
00349             request_url = request_url + "&units=%s" % self._units
00350         request_url = request_url + "&appid=%s" % self._api_key
00351         # download
00352         try:
```

```
00353             self._logger.debug("Requesting json. Forecast mode - %s" % forecast)
00354             response_json = urllib.urlopen(request_url)
00355         except IOError as e:
00356             raise IOError('Fail to download JSON with error %s' % e)
00357 
00358         weather_json = json.loads(response_json.read())
00359         # Parse
00360         if str(weather_json['cod']) != "200":
00361             raise IOError("Fail to retrieve json with error code %s - request string %s" % (str(
      weather_json['cod']),
00362                                                                                              request_url))
00363         else:
00364             self._logger.debug("Weather data receive")
00365         if forecast:
00366             forecast_time = []
00367             for single_forecast in weather_json['list']:
00368                 # collect all forecast times
00369                 forecast_time.append(single_forecast['dt'])
00370 
00371             pos = bisect_left(forecast_time, self._requested_time)
00372             if pos == 0:
00373                 index = 0
00374             elif pos == len(forecast_time):
00375                 index = len(forecast_time) - 1
00376             else:
00377                 before = forecast_time[pos - 1]
00378                 after = forecast_time[pos]
00379                 if after - self._requested_time < self.
      _requested_time - before:
00380                     index = pos
00381                 else:
00382                     index = pos - 1
00383 
00384             self.weather_data = dict(
00385                 title=str(weather_json['list'][index]['weather'][0]['main']),
00386                 description=str(weather_json['list'][index]['weather'][0]['description']),
00387                 icon=str(weather_json['list'][index]['weather'][0]['icon']),
00388                 temp=weather_json['list'][index]['main']['temp'],
00389                 temp_max=weather_json['list'][index]['main'].get('temp_max', None),
00390                 temp_min=weather_json['list'][index]['main'].get('temp_min', None),
00391                 pressure=weather_json['list'][index]['main']['pressure'],
00392                 humidity=weather_json['list'][index]['main']['humidity'],
00393                 wind_speed=weather_json['list'][index]['wind'].get('speed', None),
00394                 wind_direction=weather_json['list'][index]['wind'].get('deg', None),
00395                 rain=None,
00396                 snow=None,
00397                 clouds=weather_json['list'][index]['clouds'].get('all', None),
00398                 timestamp=int(weather_json['list'][index]['dt'])
00399             )
00400             self._city_name = weather_json['city']['name']
00401         else:
00402             index = 0
00403             self.weather_data = dict(
00404                 title=str(weather_json['list'][index]['weather'][0]['main']),
00405                 description=str(weather_json['list'][index]['weather'][0]['description']),
00406                 icon=str(weather_json['list'][index]['weather'][0]['icon']),
00407                 temp=weather_json['list'][index]['main']['temp'],
00408                 temp_max=weather_json['list'][index]['main'].get('temp_max', None),
00409                 temp_min=weather_json['list'][index]['main'].get('temp_min', None),
00410                 pressure=weather_json['list'][index]['main']['pressure'],
00411                 humidity=weather_json['list'][index]['main']['humidity'],
00412                 wind_speed=weather_json['list'][index]['wind'].get('speed', None),
00413                 wind_direction=weather_json['list'][index]['wind'].get('deg', None),
00414                 rain=weather_json['list'][index].get('rain', None),
00415                 snow=weather_json['list'][index].get('show', None),
00416                 clouds=weather_json['list'][index]['clouds'].get('all', None),
00417                 timestamp=int(weather_json['list'][index]['dt'])
00418             )
00419             self._city_name = weather_json['list'][0]['name']
00420 
00421         if os.path.isdir(self.icon_folder):
00422             try:
00423                 urllib.urlretrieve(self.icon_url + self.weather_data['icon'] + ".png",
00424                                    os.path.join(self.icon_folder, self.
      weather_data['icon'] + ".png"))
00425             except IOError:
00426                 self._logger.warning('Fail to download icon')
00427             else:
00428                 self.weather_data['icon'] = os.path.join(self.
      icon_folder, self.weather_data['icon'] + ".png")
00429 
00430     ## @brief Base URL
00431     @property
00432     def base_url(self):
00433         return self._base_url
00434 
00435     @base_url.setter
```

```
00436     def base_url(self, url):
00437         self._logger.warning('Changing base url - %s' % url)
00438         self._base_url = url
00439 
00440     ## @brief Base icon URL
00441     @property
00442     def icon_url(self):
00443         return self._icon_url
00444 
00445     @icon_url.setter
00446     def icon_url(self, url):
00447         self._logger.warning('Changing icon url - %s' % url)
00448         self._icon_url = url
00449 
00450     ## @brief Icon folder path
00451     @property
00452     def icon_folder(self):
00453         return self._icon_folder
00454 
00455     @icon_folder.setter
00456     def icon_folder(self, folder):
00457         if os.path.isdir(folder):
00458             self._logger.debug("Changing icon folder - %s" % folder)
00459             self._icon_folder = folder
00460         else:
00461             raise ValueError("Incorrect folder")
00462 
00463     ## @brief Icon full path
00464     @property
00465     def icon(self):
00466         return self.weather_data['icon']
00467 
00468     ## @brief City name - updated after fetching
00469     @property
00470     def city_name(self):
00471         return self._city_name
00472 
00473     @city_name.setter
00474     def city_name(self, name):
00475         self._city_name = name
00476         self._logger.debug("City updated - %s" % name)
00477         if self._auto_update:
00478             self.update()
00479 
00480     ## @brief if auto-update enable
00481     @property
00482     def auto_update(self):
00483         return self._auto_update
00484 
00485     @auto_update.setter
00486     def auto_update(self, state):
00487         self._auto_update = state
00488         self._logger.debug("Auto update active- %s" % state)
00489 
00490     @property
00491     def request_time(self):
00492         return self._requested_time
00493 
00494     @request_time.setter
00495     def request_time(self, req_time):
00496         self._requested_time = req_time
00497         self._logger.debug("Request time updated - %i" % req_time)
00498         if self._city_name is not None:
00499             if self._auto_update:
00500                 self.update()
00501 
00502     ## @brief Units
00503     @property
00504     def units(self):
00505         if self._units is None:
00506             return 'metric'
00507         else:
00508             return self._units
00509 
00510     @units.setter
00511     def units(self, unit):
00512         if unit not in ['metric', 'imperial']:
00513             raise AttributeError("Incorrect units")
00514         else:
00515             self._units = unit
00516             self._logger.debug("Change units - %s" % unit)
00517 
00518     ## @brief Temperture
00519     ## @pre Valid only after update
00520     @property
00521     def temp(self):
00522         return self.weather_data['temp']
```

```
00523
00524     ## @brief Temperature maximum
00525     ## @pre Valid only after update
00526     @property
00527     def temp_max(self):
00528         return self.weather_data['temp_max']
00529
00530     @property
00531     def temp_min(self):
00532         return self.weather_data['temp_min']
00533
00534     ## @brief Temperature minimum
00535     ## @pre Valid only after update
00536     @property
00537     def pressure(self):
00538         return self.weather_data['pressure']
00539
00540     ## @brief Humidity
00541     ## @pre Valid only after update
00542     @property
00543     def humidity(self):
00544         return self.weather_data['humidity']
00545
00546     ## @brief wind speed in selected units
00547     ## @pre Valid only after update
00548     @property
00549     def wind_speed(self):
00550         return self.weather_data['wind_speed']
00551
00552     ## @brief Wind angle
00553     ## @pre Valid only after update
00554     @property
00555     def wind_direction(self):
00556         return self.weather_data['wind_direction']
00557
00558     ## @brief If rain
00559     ## @pre Valid only after update
00560     @property
00561     def rain(self):
00562         return self.weather_data['rain']
00563
00564     #3 @brief If show
00565     ## @pre Valid only after update
00566     @property
00567     def show(self):
00568         return self.weather_data['snow']
00569
00570     ## @brief Cloud
00571     ## @pre Valid only after update
00572     @property
00573     def clouds(self):
00574         return self.weather_data['clouds']
00575
00576     ## @brief Measure time - Unix time
00577     ## @pre Valid only after update
00578     @property
00579     def measure_time(self):
00580         return self.weather_data['timestamp']
00581
00582     ## @brief One line weather description
00583     ## @pre Valid only after update
00584     @property
00585     def title(self):
00586         return self.weather_data['title']
00587
00588     ## @brief Short weather description
00589     ## @pre Valid only after update
00590     @property
00591     def short_description(self):
00592         return self.weather_data['description']
00593
00594     ## @brief Textual wind description
00595     ## @pre Valid only after update
00596     @property
00597     def wind_description(self):
00598         description = ""
00599         wind_direction = {0: 'northerly', 45: 'northeasterly', 90: 'easterly', 135: 'southeasterly',
00600                           180: 'southerly', 225: 'southwesterly', 270: 'westerly', 315: 'Northwesterly'}
00601
00602         if self.wind_direction is not None:
00603             wind_direction_description = wind_direction[min(wind_direction, key=lambda x: abs(x - self.
    wind_direction))]
00604
00605             if self.wind_speed is not None:
00606                 if self.units == 'metric':
00607                     if self.wind_speed < 0.3:
00608                         description = "without wind"
```

```
00609                     elif self.wind_speed < 1.5:
00610                         description = "%s light wind" % wind_direction_description
00611                     elif self.wind_speed < 3.3:
00612                         description = "%s light breeze" % wind_direction_description
00613                     elif self.wind_speed < 5.5:
00614                         description = "%s gentle breeze" % wind_direction_description
00615                     elif self.wind_speed < 7.9:
00616                         description = "%s moderate breeze" % wind_direction_description
00617                     elif self.wind_speed < 10.7:
00618                         description = "%s fresh breeze" % wind_direction_description
00619                     elif self.wind_speed < 13.8:
00620                         description = "%s strong breeze" % wind_direction_description
00621                     elif self.wind_speed < 17.1:
00622                         description = "%s high wind" % wind_direction_description
00623                     elif self.wind_speed < 20.7:
00624                         description = "%s Gale" % wind_direction_description
00625                     elif self.wind_speed < 24.4:
00626                         description = "%s strong gale" % wind_direction_description
00627                     elif self.wind_speed < 28.4:
00628                         description = "%s storm" % wind_direction_description
00629                     elif self.wind_speed < 32.6:
00630                         description = "violent storm"
00631                     else:
00632                         description = "hurricane"
00633                 else:
00634                     if self.wind_speed < 1:
00635                         description = "without wind"
00636                     elif self.wind_speed < 3:
00637                         description = "%s light wind" % wind_direction_description
00638                     elif self.wind_speed < 7:
00639                         description = "%s light breeze" % wind_direction_description
00640                     elif self.wind_speed < 12:
00641                         description = "%s gentle breeze" % wind_direction_description
00642                     elif self.wind_speed < 18:
00643                         description = "%s moderate breeze" % wind_direction_description
00644                     elif self.wind_speed < 24:
00645                         description = "%s fresh breeze" % wind_direction_description
00646                     elif self.wind_speed < 31:
00647                         description = "%s strong breeze" % wind_direction_description
00648                     elif self.wind_speed < 38:
00649                         description = "%s high wind" % wind_direction_description
00650                     elif self.wind_speed < 46:
00651                         description = "%s Gale" % wind_direction_description
00652                     elif self.wind_speed < 54:
00653                         description = "%s strong gale" % wind_direction_description
00654                     elif self.wind_speed < 63:
00655                         description = "%s storm" % wind_direction_description
00656                     elif self.wind_speed < 70:
00657                         description = "violent storm"
00658                     else:
00659                         description = "hurricane"
00660 
00661         return description
00662 
00663     ## @brief Long weather description
00664     ## @pre Valid only after update
00665     @property
00666     def description(self):
00667         desc = "Weather in %s is %s with temperature %3.1f " % (self.city_name, self.
      title, self.temp)
00668         wind = self.wind_description
00669         if "without" not in wind:
00670             desc = desc + " with %s" % wind
00671         return desc
```

Index