DiWaCS Documentation

Release 0.9.2.1

Nick Eriksson

May 20, 2013

CONTENTS

DiWaCS is an application developed for DiWa smart space and should be used **only** inside **Diwaamo**. DiWaCS connects to address **239.128.128.1:5555** using Pragmatic General Multicast (PGM). DiWaCS is built on Python and WxPython is used for UI programming. Currently, only supported platform is **Windows 7**.

Required python modules for DiWaCS:

- Configobj http://www.voidspace.org.uk/python/configobj.html
- PIL http://www.pythonware.com/products/pil/
- Python Pubsub http://pubsub.sourceforge.net/
- SQLAlchemy http://www.sqlalchemy.org/
- Watchdog http://packages.python.org/watchdog/
- WxPython http://www.wxpython.org
- ZeroMQ http://zeromq.org with openpgm support http://code.google.com/p/openpgm/

Contents:

CHAPTER

ONE

AUTOMATED CODE DOCUMENTATION

Documentation generated on 2013-05-20 at 13:41.

1.1 Add file module

Created on 5.6.2012

@author: neriksso

@requires: ZeroMQ

synopsis Used to add a file in the current project.

add_file.**main**()

Main function of the sub program.

Sub program is meant to be bound to windows explorer context menu. Context menu allows the user to quickly add files to project without interacting with DiWaCS directly.

Transmits the add_file command to DiWaCS via interprocess socket.

Parameters **filepath** (*String*) – Path of the file to be added.

Returns windows success code (0 on success).

Return type Integer

1.2 Send file module

Created on 5.6.2012

@author: neriksso

@requires: ZeroMQ

synopsis Used to send a file to another node.

send_file_to.**main**()

Main function of the sub program.

Sub program is meant to be bound to windows explorer context menu. Context menu allows the user to quickly send files without interacting with DiWaCS directly.

Transmits the send_to command to DiWaCS via interprocess connection.

Parameters

- **node_id** (*Integer*) – ID of the node to send the file to.
- **filepath** (*String*) – Path of the file to be sent.

Returns windows success code (0 on success).

Return type Integer

1.3 Controller module

Created on 28.5.2012

@author: neriksso

`controller.`**`AddComputerToSession`**(*session*, *name*, *ip*, *wos_id*)
Adds a computer to a session.

Parameters

- **session** (`models.Session`) – A current session.
- **name** (*String.*) – A name of the computer.
- **ip** (*Integer.*) – Computers IP address.
- **wos_id** (*Integer.*) – Wos id of the computer.

`controller.`**`AddEvent`**(*session_id*, *title*, *desc*)
Adds an event to the database.

Parameters

- **session** (`models.Session`) – The current session.
- **desc** (*String.*) – Description of the event.

`controller.`**`AddFileToProject`**(*file*, *project_id*)
Add a file to project. Copies it to the folder and adds a record to database.

Parameters

- **file** (*String*) – A filepath.
- **project_id** – Project id from database.

Returns New filepath.

Return type String

`controller.`**`AddProject`**(*data*)
Adds a project to database and returns a project instance

Parameters **data** (*A dictionary*) – Project information

Return type an instance of `models.Project`

`controller.`**`ConnectToDatabase`**(*expire=False*)
Connect to the database and return a Session object

`controller.`**`CreateAll`**()
Create tables to the database

`controller.`**`CreateFileaction`**(*path*, *action*, *session_id*, *project_id*)
Logs a file action to the database.

Parameters

- **path** (*String.*) – Filepath.
- **action** (*Integer.*) – File action id.
- **session_id** (*Integer.*) – Current session id.
- **project_id** (*Integer.*) – Project id from database.

controller.**DeleteRecord**(*Model*, *idNum*)
Delete a record from database

Parameters

- **Model** (sqlalchemy.ext.declarative.declarative_base().) – The model for which to delete a record.
- **idNum** (*Integer.*) – Recond id.

controller.**EditProject**(*idNum*, *row*)
Update a project info

Parameters

- **idNum** (*Integer.*) – Database id number of the project.
- **row** (*A dictionary*) – The new project information.

controller.**EndSession**(*session_id*)
Ends a session, sets its endtime to database. Ends file scanner.

Parameters **session** (models.Session) – Current session.

class controller.**FILE_ACTION_SCANNER**(*session_id*, *project_id*, *path*)
A scanner thread for monitoring user actions (Open, Close, Create, etc..) during a session. Utilizes Nirsoft's tools RecentFilesView and OpenedFilesView .

Parameters

- **session_id** (*Integer.*) – Current session id from database.
- **project_id** (*Integer.*) – Current project id from database.
- **path** (*String.*) – Filepath of project folder.

run()
Starts the thread.

stop()
Stops the thread.

controller.**GetOrCreate**(*session*, *model*, ***kwargs*)
Fetches or creates a instance.

Parameters

- **session** (models.Session) – a related session
- **model** (sqlalchemy.ext.declarative.declarative_base().) – The model of which an instance is wanted

controller.**GetProject**(*project_id*)
Fetches projects by a company.

Parameters **company_id** (*Integer.*) – A company id from database.

controller.**GetProjectPath**(*project_id*)
Fetches the project path from database and return it.

Parameters **project_id** (*Integer.*) – Project id for database.

Return type String.

controller.**GetProjectsByCompany**(*company_id*)
Fetches projects by a company.

Parameters **company_id** (*Integer.*) – A company id from database.

controller.**GetRecentFiles**(*project_id*)
Fetches files accessed recently in the project sessions from the database.

Todo

Add a limit parameter, currently fetches all files.

Todo

Duplicate check.

Parameters **project_id** (*Integer.*) – The project id

Return type a list of files

controller.**GetSessionsByProject**(*project_id*)
Fetches sessions for a project.

Parameters **project_id** (*Integer.*) – Project id from database.

controller.**InitSyncProjectDir**(*project_id*)
Initial sync of project dir and database.

Parameters **project_id** (*Integer.*) – Project id from database.

controller.**IsProjectFile**(*filename*, *project_id*)
Checks, if a file belongs to a project. Checks both project folder and database.

Parameters

- **filename** (*String.*) – a filepath.
- **project_id** (*Integer.*) – Project id from database.

Return type Boolean.

class controller.**PROJECT_FILE_EVENT_HANDLER**(*project_id*)
Handler for FileSystem events on project folder.

Parameters **project_id** (*Integer.*) – Project id from database.

on_created(*event*)
On_created event handler. Logs to database.

Parameters **event** (an instance of watchdog.events.FileSystemEvent) – The event.

on_deleted(*event*)
On_deleted event handler. Logs to database.

Parameters **event** (an instance of watchdog.events.FileSystemEvent) – The event.

on_modified(*event*)
On_modified event handler. Logs to database.

Parameters **event** (an instance of watchdog.events.FileSystemEvent) – The event.

class controller.**SCAN_HANDLER**(*project_id*)
Handler for FileSystem events on SCANNING folder.

Parameters **project_id** (*Integer.*) – Project id from database.

on_created(*event*)
On_created event handler. Logs to database.

Parameters **event** (an instance of watchdog.events.FileSystemEvent) – The event.

controller.**StartNewSession**(*project_id*, *session_id=None*, *old_session_id=None*)
Creates a session to the database and return a session object.

Parameters

- **project_id** (*Integer.*) – Project id from database.
- **session_id** (*Integer.*) – an existing session id from database.
- **old_session_id** (*Integer.*) – A session id of a session which will be continued.

1.4 Models module

Created on 23.5.2012

@author: neriksso

@requires: sqlalchemy

@requires: pywin32

synopsis Used to represent the different database structures on DiWa.

class models.**Action**(*name*)
A class representation of a action. A file action uses this to describe the action.

Field:

- id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the action, used as primary key in database table.
- name (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Name of the action (Max 50 characters).

Parameters **name** (String) – Name of the action.

class models.**Activity**(*project*, *session=None*)
A class representation of an activity.

Fields:

- id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of activity, used as primary key in database table.
- session_id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the session activity belongs to.
- session (sqlalchemy.orm.relationship) - Session relationship.
- project_id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the project activity belongs to.
- project (sqlalchemy.orm.relationship) - Project relationship.

- active (sqlalchemy.schema.Column(sqlalchemy.types.Boolean)) - Boolean flag indicating that the project is active.

Parameters

- **project** (models.Project) – Project activity belongs to.
- **session** (models.Session) – Optional session activity belongs to.

class models.**Company**(*name*)

A class representation of a company.

Fields:

- id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the company, used as primary key in database table.
- name (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Name of the company (Max 50 characters).

Parameters **name** (String) – The name of the company.

class models.**Computer**(***kwargs*)

A class representation of a computer.

Fields:

- id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of computer, used as primary key in database table.
- name (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Name of the computer.
- ip(sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.INTEGER)) - Internet Protocol address of the computer (Defined as unsigned).
- mac (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Media Access Control address of the computer.
- time (sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.DATETIME)) - Time of the last network activity from the computer.
- screens(sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.SMALLINT)) - Number of screens on the computer.
- responsive(sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.TINYINT)) - The responsive value of the computer.
- user_id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the user currently using the computer.
- user(sqlalchemy.orm.relationship) - The current user.
- wos_id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - **WOS** ID.

class models.**Event**(***kwargs*)

A class representation of Event. A simple note with timestamp during a session.

Fields:

- id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the event, used as primary key in database table.

- `title` (`sqlalchemy.schema.Column(sqlalchemy.types.String)`) - Title of the event (Max 40 characters).
- `desc` (`sqlalchemy.schema.Column(sqlalchemy.types.String)`) - More in-depth description of the event (Max 500 characters).
- `time` (`sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.DATETIME)`) - Time the event took place.
- `session_id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the session this event belongs to.
- `session` (`sqlalchemy.orm.relationship`) - Session this event belongs to.

class `models.`**`File`**(***kwargs*)

A class representation of a file.

Fields:

- `id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the file, used as primary key in database table.
- `path` (`sqlalchemy.schema.Column(sqlalchemy.types.String)`) - Path of the file on DiWa (max 255 chars).
- `project_id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the project this file belongs to.
- `project` (`sqlalchemy.orm.relationship`) - Project this file belongs to.

class `models.`**`FileAction`**(*file*, *action*, *session=None*, *computer=None*, *user=None*)

A class representation of a fileaction.

Fields:

- `id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the FileAction, used as primary key in the database table.
- `file_id` (`sqlalchemy.schema.Column(sqlaclhemy.types.Integer)`) - ID of the file this FileAction affects.
- `file` (`sqlalchemy.orm.relationship)`) - The file this FileAction affects.
- `action_id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the action affecting the file.
- `action` (`sqlalchemy.orm.relationship)`) - Action affecting the file.
- `action_time` (`sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.DATETIME)`) - Time the action took place on.
- `user_id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the user performing the action.
- `user` (`sqlalchemy.orm.relationship`) - User peforming the action.
- `computer_id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the computer user performed the action on.
- `computer` (`sqlalchemy.orm.relationship`) - Computer user performed the action on.
- `session_id` (`sqlalchemy.schema.Column(sqlalchemy.types.Integer)`) - ID of the session user performed the action in.
- `session` (`sqlalchemy.orm.relationship`) - Session user performed the action in.

Parameters

- **file** (models.File) – The file which is subjected to the action.
- **action** (models.Action) – The action which is applied to the file.
- **session** (models.Session) – The session in which the FileAction took place on.
- **computer** (models.Computer) – The computer from which the user performed the action.
- **user** (models.User) – The user performing the action.

class models.**Project**(*name*, *company*, *password*)

A class representation of a project.

Fields:

- id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of project, used as primary key in database table.
- name (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Name of the project (Max 50 characters).
- company_id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the company that owns the project.
- company (sqlalchemy.orm.relationship) - The company that owns the project.
- dir (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Directory path for the project files (Max 255 characters).
- password (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Password for the project (Max 40 characters).
- members (sqlalchemy.orm.relationship) - The users that work on the project.

Parameters

- **name** (String) – Name of the project.
- **company** (models.Company) – The owner of the project.

class models.**Session**(*project*)

A class representation of a session.

Fields:

- id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of session, used as primary key in database table.
- name (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Name of session (Max 50 characters).
- project_id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the project the session belongs to.
- project (sqlalchemy.orm.relationship) - The project the session belongs to.
- starttime(sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.DATETIME)) - Time the session began, defaults to *now()*.
- endtime(sqlalchemy.schema.Column(sqlalchemy.dialects.mysql.DATETIME)) - The time session ended.

- previous_session_id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the previous session.
- previous_session (sqlalchemy.orm.relationship) - The previous session.
- participants (sqlalchemy.orm.relationship) - Users that belong to this session.
- computers (sqlalchemy.orm.relationship) - Computers that belong to this session.

Parameters **project** (models.Project) – The project for the session.

addUser(*user*)
Add users to a session.

Parameters **user** (models.User) – User to be added into the session.

fileRoutine()
File checking routine for logging.

Throws IOError When log.txt is not available for write access.

get_last_checked()
Fetch last_checked field.

Returns last_checked field (None before models.Session.start() is called).

Return type datetime.datetime or None

start()
Start a session. Set the last_checked field to current DateTime.

class models.**User**(*name*, *company*)
A class representation of a user.

Fields:

- id(sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - ID of the user, used as primary key in database table.
- name (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Name of the user (Max 50 characters).
- email (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Email address of the user (Max 100 characters).
- title (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Title of the user in the company (Max 50 characters).
- department (sqlalchemy.schema.Column(sqlalchemy.types.String)) - Department of the user in the company (Max 100 characters).
- company_id (sqlalchemy.schema.Column(sqlalchemy.types.Integer)) - Company id of the employing company.
- company (sqlalchemy.orm.relationship) - Company relationship.

Parameters

- **name** (String) – Name of the user.
- **company** (models.Company) – The employer.

1.5 SWNP module

Created on 30.4.2012

@author: neriksso

class swnp.**Message**(*TAG*, *PREFIX*, *PAYLOAD*)
A class representation of a Message.

Messages are divided into three parts: TAG, PREFIX, PAYLOAD. Messages are encoded to json for transmission.

Parameters

- **TAG** (*String.*) – TAG of the message.
- **PREFIX** (*String.*) – PREFIX of the message.
- **PAYLOAD** (*String.*) – PAYLOAD of the message.

static from_json(*json_dict*)
Return a message from json.

Parameters **json_dict** (*json.*) – The json.

Return type swnp.Message.

static to_dict(*msg*)
Return a message in a dict.

Parameters **msg** (swnp.Message) – The message.

Return type Dict.

class swnp.**Node**(*id*, *screens*, *name=None*, *data=None*)
A class representation of a node in the network.

Parameters

- **id** (*Integer.*) – Node id
- **screens** (*Integer.*) – Amount of visible screens.
- **name** (*String.*) – The name of the node.

refresh()
Updates the timestamp.

class swnp.**SWNP**(*pgm_group*, *screens=0*, *name=None*, *id=None*, *context=None*, *error_handler=None*)
The main class of swnp.

This class has the required ZeroMQ bindings and is responsible for communicating with other instances.

> **Warning:** Only one instance per computer

Parameters

- **screens** (*Integer.*) – The number of visible screens. Defaults to 0.
- **name** (*String.*) – The name of the instance. Optional.

close()
Closes all connections and exits.

do_ping()
Send a PING message to the network.

find_node(*node_id*)
Search the node list for a specific node.

Parameters **node_id** (*Integer.*) – The id of the searched node.

Return type swnp.Node

get_buffer()
Gets the buffered messages and returns them

Return type json.

get_list()
Returns a list of all nodes

Return type list.

get_screen_list()
Returns a list of screens nodes.

Return type list.

ping_handler(*payload*)
A handler for PING messages. Sends update_screens, if necessary.

Parameters **payload** (*String.*) – The payload of a PING message.

ping_routine(*error_handler*)
A routine for sending PING messages at regular intervals.

send(*tag*, *prefix*, *message*)
Send a message to the network.

Parameters

- **tag** (*String.*) – The tag of the message; recipient.
- **prefix** (*String.*) – The prefix of the message.
- **message** (*String.*) – The payload of the message.

set_screens(*screens*)
Sets the number of screens for the instance.

Parameters **screens** (*Integer.*) – New number of screens.

shutdown()
shuts down all connections, no exit.

sub_routine(*sub_url*, *context*)
Subscriber routine for the node ID.

Parameters

- **sub_url** (*String*) – Subscribing URL.
- **context** (zmq.core.context.Context) – ZeroMQ context for message sending

sub_routine_sys(*sub_url*, *context*)
Subscriber routine for the node ID.

Parameters

- **sub_url** (*String*) – Subscribing URL.
- **context** (zmq.core.context.Context) – ZeroMQ context for message sending

sync_handler(*msg*)
Handler for sync messages.

Deprecated since version 0.2.

Parameters **msg** (swnp.Message) – The message.

sys_handler(*msg*)
Handler for "SYS" messages.

Parameters **msg** (swnp.Message) – The received message.

timeout_routine()
Routine for checking node list and removing nodes with timeout.

1.6 Filesystem module

Created on 17.5.2013

filesystem.**CopyFileToProject**(*filepath*, *project_id*)
Copy file to project dir and return new filepath in project dir.

Parameters

- **filepath** (*String.*) – The file path.
- **project_id** (*Integer.*) – Project id from database.

filesystem.**CopyToTemp**(*filepath*)
Copy a file to temporary folder.

Parameters **filepath** (*String.*) – The file path.

filesystem.**CreateProjectDir**(*dir_name*)
Creates a project directory, if one does not exist in the file system

Parameters **dir_name** (*String.*) – Name of the directory

filesystem.**FileToBase64**(*filepath*)
Transform a file to a binary object.

Parameters **filepath** (*String.*) – The file path.

filesystem.**GetFileExtension**(*path*)
Returns the file extension of a file

Parameters **path** (*String*) – The file path.

Return type String.

filesystem.**GetNodeImg**(*node*)
Searches for a node's image in STORAGE.

Parameters **node** (*Integer.*) – The node id.

filesystem.**IsSubtree**(*filename*, *parent*)
Determines, if filename is inside the parent folder.

Parameters

- **filename** (*String.*) – The file path.
- **parent** (*String.*) – The parent file path.

filesystem.**OpenFile**(*filepath*)
Opens a file path.

Parameters **filepath** (*String.*) – The file path.

filesystem.**OpenedFilesQuery**()
Calls the openedfilesview.

filesystem.**RecentFilesQuery**()
Calls the recentfilesview.

filesystem.**SaveScreen**(*win*, *filepath*)
Saves the background image of the desktop.

Parameters **filename** (*String.*) – The filename for the saved image.

filesystem.**SearchFile**(*filename*, *search_path*)
Search file in a given path.

Parameters

- **filename** (*String.*) – The file name.
- **search_path** (*String.*) – The search path.

1.7 Utils module

Created on 17.5.2013

utils.**DottedIPToInt**(*dotted_ip*)
Transforms a dotted IP address to Integer.

Parameters **dotted_ip** (*String*) – The IP address.

Returns The IP address.

Return type Integer

utils.**GetLANMachines**(*lan_ip*)

Parameters **lan_ip** (*string*) – Local Area Network IP.

Returns lan machines

Return type string[]

utils.**GetLocalIPAddress**(*target*)
Used to get local Internet Protocol address.

Returns The current IP address.

Return type string

utils.**GetMacForIp**(*ip*)
Returns the mac address for an local IP address.

Parameters **ip** (*String.*) – IP address

utils.**IntToDottedIP**(*intip*)
Transforms an Integer IP address to dotted representation.

Parameters **intip** (*Integer*) – The IP

Returns The IP

Return type string

utils.**IterIsLast**(*iterable*) → generates (item, islast) pairs
Generates pairs where the first element is an item from the iterable source and the second element is a boolean flag indicating if it is the last item in the sequence.

Parameters **iterable** (*iterable*) – The iterable element.

utils.**MapNetworkShare**(*letter*, *share*)
Maps the network share to a letter.

Parameters

- **letter** (*String.*) – The letter for which to map.
- **share** (*String.*) – The network share.

1.8 Wos module

Created on 8.5.2012

@author: neriksso

class wos.**AddProjectDialog**(*parent*, *title*, *project_id=None*)
A dialog for adding a new project

Parameters

- **parent** (wx.Frame) – Parent frame.
- **title** (*String.*) – A title for the dialog.

OnAdd(*e*)
Handles the addition of a project to database, when "Add" button is pressed.

Parameters **e** (*Event.*) – GUI Event.

OnClose(*e*)
Handles "Close" button presses

Parameters **e** (*Event.*) – GUI Event.

class wos.**AudioRecorder**(*parent*)
A thread for capturing audio continuously. It keeps a buffer that can be saved to a file.

run()
Continuously record from the microphone to the buffer. If the buffer is full, the first frame will be removed and the new block appended.

save(*ide*, *path*)
Save the buffer to a file.

class wos.**CHECK_UPDATE**
Thread for checking version updates.

class wos.**CONN_ERR_TH**(*parent*)
Thread for checking connection errors.

Parameters **parent** (*wx.Frame.*) – Parent object.

run()
Starts the thread.

class wos.**CURRENT_PROJECT**(*project_id*, *swnp*)
Thread for transmitting current project selection. When user selects a project, an instance is started. When a new selection is made, by any Chimaira instance, the old instance is terminated.

Parameters

- **project_id** (*Integer.*) – Project id from the database.
- **swnp** (swnp.SWNP) – SWNP instance for sending data to the network.

run()
Starts the thread.

stop()
Stops the thread.

class wos.**CURRENT_SESSION**(*parent*, *swnp*)
Thread for transmitting current session id, when one is started by the user. When the session is ended, by any Chimaira instance, the instance is terminated.

Parameters

- **session_id** (*Integer.*) – Session id from the database.
- **swnp** (swnp.SWNP) – SWNP instance for sending data to the network.

run()
Starts the thread.

stop()
Stops the thread

class wos.**DropTarget**(*window*, *parent*, *i*)
Implements drop target functionality to receive files, bitmaps and text

OnData(*x*, *y*, *d*)
Handles drag/dropping files/text or a bitmap

class wos.**EventList**(*parent*, **args*, ***kwargs*)
A Frame which displays the possible event titles and handles the event creation.

GetIcon(*icon*)
Fetches gui icons.

Parameters **icon** (*String.*) – The icon file name.

Return type wx.Image

class wos.**GUI**(*parent*, *title*)
WOS Application Frame

Parameters

- **parent** (wx.Frame) – Parent frame.
- **title** (*String.*) – Title for the frame

AlignCenterTop()
Aligns frame to Horizontal center and vertical top

CreateConfig()
Creates a config file

GetIcon(*icon*)
Fetches gui icons.

Parameters **icon** (*String.*) – The icon file name.

Return type `wx.Image`

HandleFileSend(*file*)
Sends a file link to another node

HideScreens()
Hides all screens

InitScreens()
Inits Screens

InitUI()
UI initing

LoadConfig()
Loads a config file or creates one

MessageHandler(*message*)
Message handler for received messages

Parameters **message** (an instance of `swnp.Message`) – Received message.

OnAboutBox(*unused_event*)
About dialog

Parameters **e** (*Event.*) – GUI Event.

OnCreateTables(*evt*)
Create necessary db tables

Parameters **evt** (*Event*) – GUI event.

OnEvtBtn(*evt*)
Event Button handler

Parameters **evt** (*Event.*) – GUI Event.

OnExit(*event*)
Exits program.

Parameters **event** (*Event.*) – GUI Event

OnIconify(*unused_event*)
Window minimize event handler

Parameters **evt** (*Event.*) – GUI Event.

OnProjectSelected()
Project selected event handler

OnSession(*evt*)
Session button pressed

Parameters **evt** (*Event*) – GUI Event.

OnTaskBarActivate(*evt*)
Taskbar activate event handler

Parameters **evt** (*Event.*) – GUI Event.

OnTaskBarClose(*unused_event*)
Taskbar close event handler.

Parameters **evt** (*Event.*) – GUI Event.

OpenProjectDir(*evt*)
Opens project directory in windows explorer

Parameters **evt** (*event.*) – The GUI event.

PaintSelect(*evt*)
Paints the selection of a node.

Note: For future use.

Parameters **evt** (*Event.*) – GUI Event

SelectNode(*evt*)
Handles the selection of a node, start remote control.

Note: For future use.

Parameters **evt** (*Event.*) – GUI Event

SelectProjectDialog(*evt*)
Select project event handler

Parameters **evt** (*Event.*) – GUI Event.

SetCurrentProject(*project_id*)
Start current project loop :param project_id: The project id from database. :type project_id: Integer.

SetCurrentSession(*session_id*)
Set current session

Parameters **session_id** (*Integer.*) – a session id from database.

SetProjectObserver()
Observer for filechanges in project dir

SetScanObserver()
Observer for created files in scanned or taken with camera

Shift(*evt*)
Caroussel Shift function

Parameters **evt** (*Event.*) – GUI Event.

ShowPreferences(*evt*)
Preferences dialog event handler

Parameters **evt** (*Event.*) – GUI Event.

StartCurrentProject()
Start current project loop

StartCurrentSession()
Start current project loop

SwnpSend(*node*, *message*)
Sends a message to the node.

Parameters

- **node** (*String.*) – The node for which to send a message.
- **message** (*String.*) – The message.

UpdateScreens(*update*)
Called when screens need to be updated and redrawn

Parameters **update** (*Boolean.*) – Pubsub needs one param, therefore it is called update.

class wos.**INPUT_CAPTURE**(*parent*, *swnp*)
Thread for capturing input from mouse/keyboard.

Parameters

- **parent** (GUI.) – Parent instance.
- **swnp** (swnp.SWNP) – SWNP instance for sending data to the network.

OnMouseEvent(*event*)
WM_MOUSEFIRST = 0x200

WM_MOUSEMOVE = 0x200

WM_LBUTTONDOWN = 0x201

WM_LBUTTONUP = 0x202

WM_LBUTTONDBLCLK = 0x203

WM_RBUTTONDOWN = 0x204

WM_RBUTTONUP = 0x205

WM_RBUTTONDBLCLK = 0x206

WM_MBUTTONDOWN = 0x207

WM_MBUTTONUP = 0x208

WM_MBUTTONDBLCLK = 0x209

WM_MOUSEWHEEL = 0x20A

WM_MOUSEHWHEEL = 0x20E

run()
Starts the thread.

stop()
Stops the thread.

class wos.**MySplashScreen**(*parent=None*)
Create a splash screen widget.

class wos.**PreferencesDialog**(*config*, *evtlist*)
Creates and displays a preferences dialog that allows the user to change some settings.

Parameters **config** – a Config object

loadPreferences()
Load the current preferences and fills the text controls

onCancel(*event*)
Closes the dialog without modifications.

Parameters **event** (*Event.*) – GUI event.

openConfig(*event*)
Closes the dialog without modifications.

Parameters **event** (*Event.*) – GUI event.

savePreferences(*event*)
Save the preferences.

Parameters **event** (*Event.*) – GUI Event.

class wos.**ProjectSelectDialog**(*parent*)
A dialog for selecting a project.

Parameters **parent** (wx.Frame) – Parent frame.

AddEvent(*unused_event*)
Shows a modal dialog for adding a new project.

Parameters **event** (*Event.*) – GUI Event.

DelEvent(*unused_event*)
Handles the selection of a project. Starts a wos.CURRENT_PROJECT, if necessary. Shows a dialog of the selected project.

Parameters **evt** (*Event.*) – GUI Event.

EditEvent(*unused_event*)
Shows a modal dialog for adding a new project.

Parameters **event** (*Event.*) – GUI Event.

GetProjects(*company_id=1*)
Fetches all projects from the database, based on the company.

Parameters **company_id** (*Integer.*) – A company id, the owner of the projects. Defaults to 1.

SelEvent(*unused_event*)
Handles the selection of a project. Starts a wos.CURRENT_PROJECT, if necessary. Shows a dialog of the selected project.

Parameters **evt** (*Event.*) – GUI Event.

onCancel(*unused_event*)
Handles "Cancel" button presses.

Parameters **event** (*Event.*) – GUI Event.

class wos.**SEND_FILE_CONTEX_MENU_HANDLER**(*parent*, *context*, *send_file*, *handle_file*)
Thread for OS contex menu actions like file sending to other node.

Parameters

- **context** (*ZeroMQ context.*) – Context for creating sockets.
- **send_file** (*Function.*) – Sends files.
- **handle_file** (*Function.*) – Handles files

run()
Starts the thread

stop()
Stops the thread

class wos.**SysTray**(*parent*)
Taskbar Icon class

Parameters **parent** (wx.Frame) – Parent frame

CreateMenu()
Create systray menu

ShowMenu(*event*)
Show popup menu

Parameters **event** (*Event.*) – GUI event.

class wos.**UpdateDialog**(*title*, *url*, **args*, ***kwargs*)
A Dialog which notifies about a software update

class wos.**WORKER_THREAD**(*parent*)
Worker thread for non-UI jobs.

Parameters

- **context** (*ZeroMQ context.*) – Context for creating sockets.
- **send_file** (*Function.*) – Sends files.
- **handle_file** (*Function.*) – Handles files

AddProjectReg()
Adds project folder to registry

AddRegEntry(*name*, *id*)
Adds a node to registry

Parameters

- **name** (*String*) – Node name.
- **id** (*Integer.*) – Node id.

RemoveAllRegEntries()
Removes all related registry entries

parseConfig(*config*)
Handles config file settings

CHAPTER

TWO

BUGS

Bug	Description	Status
Sample bug	Description for sample	Open / Closed / Will not be fixed

CHAPTER

THREE

FEATURES

Feature	Description
Project	User can add, edit and select a project
Session	User can start, end and continue sessions
Event	User can tag an interesting event during a session
File Monitoring	Users' file actions are monitored during a session. It includes opening files.

CHAPTER

FOUR

LICENSE

European Union Public Licence

22. 1.1

EUPL © the European Community 2007

This European Union Public Licence (the "EUPL") applies to the Work or Software (as defined below) which is provided under the terms of this Licence. Any use of the Work, other than as authorised under this Licence is prohibited (to the extent such use is covered by a right of the copyright holder of the Work).

The Original Work is provided under the terms of this Licence when the Licensor (as defined below) has placed the following notice immediately following the copyright notice for the Original Work:

> Licensed under the EUPL V.1.1

or has expressed by any other mean his willingness to license under the EUPL.

4.1 1. Definitions

In this Licence, the following terms have the following meaning:

- **The Licence:** This Licence.
- **The Original Work or the Software:** The software distributed and/or communicated by the Licensor under this Licence, available as Source Code and also as Executable Code as the case may be.
- **Derivative Works:** The works or software that could be created by the Licensee, based upon the Original Work or modifications thereof. This Licence does not define the extent of modification or dependence on the Original Work required in order to classify a work as a Derivative Work; this extent is determined by copyright law applicable in the country mentioned in Article 15.
- **The Work:** The Original Work and/or its Derivative Works.
- **The Source Code:** The human-readable form of the Work which is the most convenient for people to study and modify.
- **The Executable Code:** Any code which has generally been compiled and which is meant to be interpreted by a computer as a program.
- **The Licensor:** The natural or legal person that distributes and/or communicates the Work under the Licence.
- **Contributor(s):** Any natural or legal person who modifies the Work under the Licence, or otherwise contributes to the creation of a Derivative Work.
- **The Licensee or "You":** Any natural or legal person who makes any usage of the Software under the terms of the Licence.

- **Distribution and/or Communication:** Any act of selling, giving, lending, renting, distributing, communicating, transmitting, or otherwise making available, on-line or off-line, copies of the Work or providing access to its essential functionalities at the disposal of any other natural or legal person.

4.2 2. Scope of the rights granted by the Licence

The Licensor hereby grants You a world-wide, royalty-free, non-exclusive, sublicensable licence to do the following, for the duration of copyright vested in the Original Work:

- use the Work in any circumstance and for all usage,
- reproduce the Work,
- modify the Original Work, and make Derivative Works based upon the Work,
- communicate to the public, including the right to make available or display the Work or copies thereof to the public and perform publicly, as the case may be, the Work,
- distribute the Work or copies thereof,
- lend and rent the Work or copies thereof,
- sub-license rights in the Work or copies thereof.

Those rights can be exercised on any media, supports and formats, whether now known or later invented, as far as the applicable law permits so.

In the countries where moral rights apply, the Licensor waives his right to exercise his moral right to the extent allowed by law in order to make effective the licence of the economic rights here above listed.

The Licensor grants to the Licensee royalty-free, non exclusive usage rights to any patents held by the Licensor, to the extent necessary to make use of the rights granted on the Work under this Licence.

4.3 3. Communication of the Source Code

The Licensor may provide the Work either in its Source Code form, or as Executable Code. If the Work is provided as Executable Code, the Licensor provides in addition a machine-readable copy of the Source Code of the Work along with each copy of the Work that the Licensor distributes or indicates, in a notice following the copyright notice attached to the Work, a repository where the Source Code is easily and freely accessible for as long as the Licensor continues to distribute and/or communicate the Work.

4.4 4. Limitations on copyright

Nothing in this Licence is intended to deprive the Licensee of the benefits from any exception or limitation to the exclusive rights of the rights owners in the Original Work or Software, of the exhaustion of those rights or of other applicable limitations thereto.

4.5 5. Obligations of the Licensee

The grant of the rights mentioned above is subject to some restrictions and obligations imposed on the Licensee. Those obligations are the following:

Attribution right: the Licensee shall keep intact all copyright, patent or trademarks notices and all notices that refer to the Licence and to the disclaimer of warranties. The Licensee must include a copy of such notices and a copy of the Licence with every copy of the Work he/she distributes and/or communicates. The Licensee must cause any Derivative Work to carry prominent notices stating that the Work has been modified and the date of modification.

Copyleft clause: If the Licensee distributes and/or communicates copies of the Original Works or Derivative Works based upon the Original Work, this Distribution and/or Communication will be done under the terms of this Licence or of a later version of this Licence unless the Original Work is expressly distributed only under this version of the Licence. The Licensee (becoming Licensor) cannot offer or impose any additional terms or conditions on the Work or Derivative Work that alter or restrict the terms of the Licence.

Compatibility clause: If the Licensee Distributes and/or Communicates Derivative Works or copies thereof based upon both the Original Work and another work licensed under a Compatible Licence, this Distribution and/or Communication can be done under the terms of this Compatible Licence. For the sake of this clause, "Compatible Licence" refers to the licences listed in the appendix attached to this Licence. Should the Licensee's obligations under the Compatible Licence conflict with his/her obligations under this Licence, the obligations of the Compatible Licence shall prevail.

Provision of Source Code: When distributing and/or communicating copies of the Work, the Licensee will provide a machine-readable copy of the Source Code or indicate a repository where this Source will be easily and freely available for as long as the Licensee continues to distribute and/or communicate the Work.

Legal Protection: This Licence does not grant permission to use the trade names, trademarks, service marks, or names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the copyright notice.

4.6 6. Chain of Authorship

The original Licensor warrants that the copyright in the Original Work granted hereunder is owned by him/her or licensed to him/her and that he/she has the power and authority to grant the Licence.

Each Contributor warrants that the copyright in the modifications he/she brings to the Work are owned by him/her or licensed to him/her and that he/she has the power and authority to grant the Licence.

Each time You accept the Licence, the original Licensor and subsequent Contributors grant You a licence to their contributions to the Work, under the terms of this Licence.

4.7 7. Disclaimer of Warranty

The Work is a work in progress, which is continuously improved by numerous contributors. It is not a finished work and may therefore contain defects or "bugs" inherent to this type of software development.

For the above reason, the Work is provided under the Licence on an "as is" basis and without warranties of any kind concerning the Work, including without limitation merchantability, fitness for a particular purpose, absence of defects or errors, accuracy, non-infringement of intellectual property rights other than copyright as stated in Article 6 of this Licence.

This disclaimer of warranty is an essential part of the Licence and a condition for the grant of any rights to the Work.

4.8 8. Disclaimer of Liability

Except in the cases of wilful misconduct or damages directly caused to natural persons, the Licensor will in no event be liable for any direct or indirect, material or moral, damages of any kind, arising out of the Licence or of the use of

the Work, including without limitation, damages for loss of goodwill, work stoppage, computer failure or malfunction, loss of data or any commercial damage, even if the Licensor has been advised of the possibility of such damage. However, the Licensor will be liable under statutory product liability laws as far such laws apply to the Work.

4.9 9. Additional agreements

While distributing the Original Work or Derivative Works, You may choose to conclude an additional agreement to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or services consistent with this Licence. However, in accepting such obligations, You may act only on your own behalf and on your sole responsibility, not on behalf of the original Licensor or any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against such Contributor by the fact You have accepted any such warranty or additional liability.

4.10 10. Acceptance of the Licence

The provisions of this Licence can be accepted by clicking on an icon "I agree" placed under the bottom of a window displaying the text of this Licence or by affirming consent in any other similar way, in accordance with the rules of applicable law. Clicking on that icon indicates your clear and irrevocable acceptance of this Licence and all of its terms and conditions.

Similarly, you irrevocably accept this Licence and all of its terms and conditions by exercising any rights granted to You by Article 2 of this Licence, such as the use of the Work, the creation by You of a Derivative Work or the Distribution and/or Communication by You of the Work or copies thereof.

4.11 11. Information to the public

In case of any Distribution and/or Communication of the Work by means of electronic communication by You (for example, by offering to download the Work from a remote location) the distribution channel or media (for example, a website) must at least provide to the public the information requested by the applicable law regarding the Licensor, the Licence and the way it may be accessible, concluded, stored and reproduced by the Licensee.

4.12 12. Termination of the Licence

The Licence and the rights granted hereunder will terminate automatically upon any breach by the Licensee of the terms of the Licence.

Such a termination will not terminate the licences of any person who has received the Work from the Licensee under the Licence, provided such persons remain in full compliance with the Licence.

4.13 13. Miscellaneous

Without prejudice of Article 9 above, the Licence represents the complete agreement between the Parties as to the Work licensed hereunder.

If any provision of the Licence is invalid or unenforceable under applicable law, this will not affect the validity or enforceability of the Licence as a whole. Such provision will be construed and/or reformed so as necessary to make it valid and enforceable.

The European Commission may publish other linguistic versions and/or new versions of this Licence, so far this is required and reasonable, without reducing the scope of the rights granted by the Licence. New versions of the Licence will be published with a unique version number.

All linguistic versions of this Licence, approved by the European Commission, have identical value. Parties can take advantage of the linguistic version of their choice.

4.14 14. Jurisdiction

Any litigation resulting from the interpretation of this License, arising between the European Commission, as a Licensor, and any Licensee, will be subject to the jurisdiction of the Court of Justice of the European Communities, as laid down in article 238 of the Treaty establishing the European Community.

Any litigation arising between Parties, other than the European Commission, and resulting from the interpretation of this License, will be subject to the exclusive jurisdiction of the competent court where the Licensor resides or conducts its primary business.

4.15 15. Applicable Law

This Licence shall be governed by the law of the European Union country where the Licensor resides or has his registered office. This licence shall be governed by the Belgian law if: - a litigation arises between the European Commission, as a Licensor, and any Licensee; - the Licensor, other than the European Commission, has no residence or registered office inside a European Union country.

4.16 Appendix

“Compatible Licences” according to article 5 EUPL are:

- GNU General Public License (GNU GPL) v. 2
- Open Software License (OSL) v. 2.1, v. 3.0
- Common Public License v. 1.0
- Eclipse Public License v. 1.0
- Cecill v. 2.0

CHAPTER

FIVE

USER INTERFACE

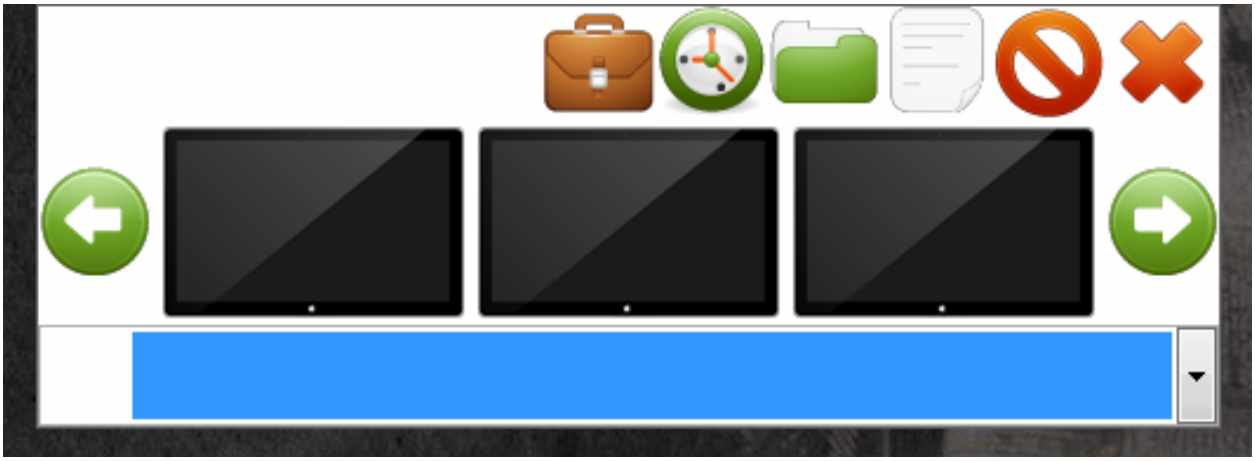

Figure 5.1: The UI of Chimaira; Several icons for different functions.

The screen icons identify different screens nodes. The user can drop files on to these icons, causing the dropped files to be opened in the specific node. The arrows control the carousel of nodes, and are visble only if more than three nodes are connected. The drop-down list in holds recently viewed files in the selected project.

Table 5.1: Icons explained

Icon	**Description**
Briefcase	Select a project
Clock	Start / End a session
Folder	Open project directory
Note	Create an Event note
Circle	Hide the application
Cross	Exit the application

CHAPTER

SIX

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

a

c

f

m

s

u

w