Chimaira Documentation

Release 0.2.1

Nick Eriksson

August 23, 2012

CONTENTS

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

Required python modules for Chimaira:

- 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

1.1 Controller module

Created on 28.5.2012

@author: neriksso

controller.**AddEvent**(*session*, *desc*)
Adds an event to the database.

Parameters

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

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.

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.`**SCANNER**(*session_id*, *project_id*, *path*)
A scanner thread for monitoring user actions (Open, Close, Create, etc..) during a session. Utilizes win32 python extensions.

Note: Not in use. Old implementation.

Deprecated since version 0.2.

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 th thread.

stop()
Stops the thread.

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

controller.**createAll**()
Create tables to the database

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

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

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.**get_or_create**(*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.**init_sync_project_dir**(*project_id*)
Initial sync of project dir and database.

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

controller.**is_project_file**(*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.

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.

1.2 Models module

Created on 23.5.2012

@author: neriksso

class models.**Action**(*name*)
A class representation of a action. Describes a file action.

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

class models.**Company**(*name*)
A class representation of a company.

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

class models.**Computer**(**kwargs)
A class representation of a computer.

Fields:

- name
- ip
- mac
- user
- wos_id

class models.**Event**(**kwargs)
A class representation of Event. A simple note with timestamp during a session.

Fields:

- desc
- time
- session

class models.**File**(**kwargs)
A class representation of a file.

Fields:

- path
- project

class models.**FileAction**(*file*, *action*, *session=None*, *computer=None*, *user=None*)
A class representation of a fileaction.

Parameters

- **file** (models.File) – The file in question.
- **action** (models.Action) – The action in question.
- **session** (models.Session) – The session in question.
- **computer** (models.Computer) – The computer in question.
- **user** (models.User) – The user performing the action.

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

Parameters **project** – The project for the session.

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

fileRoutine()
File checking routine for logging

get_last_checked()
Fetch last checked field.

Returns Last checked field

Return type Datetime

start()
Start a session. Set the last checked field.

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

Parameters

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

1.3 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*)
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**(*screens=0*, *name=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()
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.

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.

1.4 Utils

Created on 9.5.2012

@author: neriksso

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

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

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

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

utils.**SaveScreen**(*filename*)
Saves the background image of the desktop.

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

utils.**copy_file_to_project**(*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.

utils.**copy_to_temp**(*filepath*)
Copy a file to temporary folder.

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

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

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

utils.**file_to_base64**(*filepath*)
Transform a file to a binary object.

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

utils.**get_file_extension**(*path*)
Returns the file extension of a file

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

Return type String.

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

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

utils.**get_node_img**(*node*)
Searches for a nodes image. Can be used in GUI.

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

utils.**get_sendkeys**(*code*)
Returns a character for a key code.

Parameters **code** (*Integer.*) – The character code.

utils.**is_subtree**(*filename*, *parent*)
Determines, if filename is inside the parent folder.

Parameters

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

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

Parameters

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

utils.**open_file**(*filepath*)
Opens a file path.

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

utils.**opened_files_query**()
Calls the openedfilesview.

utils.**recent_files_query**()
Calls the recentfilesview.

utils.**search_file**(*filename*, *search_path*)
Search file in a given path.

Parameters

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

1.5 Wos module

Created on 8.5.2012

@author: neriksso

class wos.**AddProjectDialog**(*parent*, *title*)
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.**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**(*session_id*, *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.**EvtDialog**(*parent*)
A dialog for entering an Event / Note, captures the timestamp.

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

AddEvent(*event*)
Handles the addition of an event to the database, when "Add" button is presed.

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

onCancel(*event*)
Closes the dialog with no actions.

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

class wos.**FileDrop**(*window*, *parent*, *i*)
Filedrop handler

Parameters

- **window** (wx.Frame) – The window frame for handling FileDrop
- **parent** (wos.Wos) – Parent for handling file send.
- **i** (*Integer.*) – Id for the node.

ChangeId(*ide*)
Change the id of the instance, :param ide: New id. :type ide: Integer

OnDropFiles(*x*, *y*, *filenames*)
Handler for dropped files event.

Parameters

- **x** (*Float.*) – X-coordinate of dropped location
- **y** (*Float.*) – Y-coordinate of dropped location
- **filenames** (*A list.*) – File paths of files to be transferred.

class wos.**PreferencesDialog**(*config*)
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.

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

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

SelEvent(*evt*)
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.

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.

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

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

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

Parameters

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

:type send_file:Function. :param handle_file:Handles files :type handle_file:Function.

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.`**Wos**(*parent*, *title*)
WOS Application Frame

Parameters

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

AddProjectReg(*project_id*)
Adds project folder to registry

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

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

Parameters

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

AddScreen(*evt*)
Adds a new imaginary screen Deprecated since version 0.2.

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

HideScreens()
Hides all screens

InitScreens()
Inits Screens

InitUI()
UI initing

LRButton(*evt*)
Sends left or right arrow button presses. Deprecated since version 0.2.

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

MessageHandler(*message*)
Message handler for received messages

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

OnAboutBox(*e*)
About dialog

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

OnCreateTables(*evt*)
Create necessary db tables

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

OnDragInit(*event*)
Drag init for recent files.

Note: For future use.

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

OnEvtBtn(*evt*)
Event Button handler

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

OnExit(*event*)
Exits program

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

OnIconify(*evt*)
Window minimize event handler

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

OnProjectSelected()
Project selected event handler

OnSession(*evt*)
Start or terminate a session

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

OnTaskBarActivate(*evt*)
Taskbar activate event handler

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

OnTaskBarClose(*evt*)
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

RemoveAllRegEntries()
Removes all related registry entries

SelProjectDialog(*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.

StartCurrentProject()
Start current project loop

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

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

alignCenterTop()
Aligns frame to Horizontal center and vertical top

createConfig()
Creates a config file

fillRecentFilesDropdown()
Fills recent files dropdown

get_icon(*icon*)
Fetches gui icons.

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

Return type wx.Image

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

loadConfig()
Loads a config file or creates one

onKeyPress(*event*)
Key Press event handler Deprecated since version 0.2.

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

shift(*evt*)
Caroussel shift function

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

showPrefs(*evt*)
Preferences dialog event handler

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

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

Parameters

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

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

USER INTERFACE

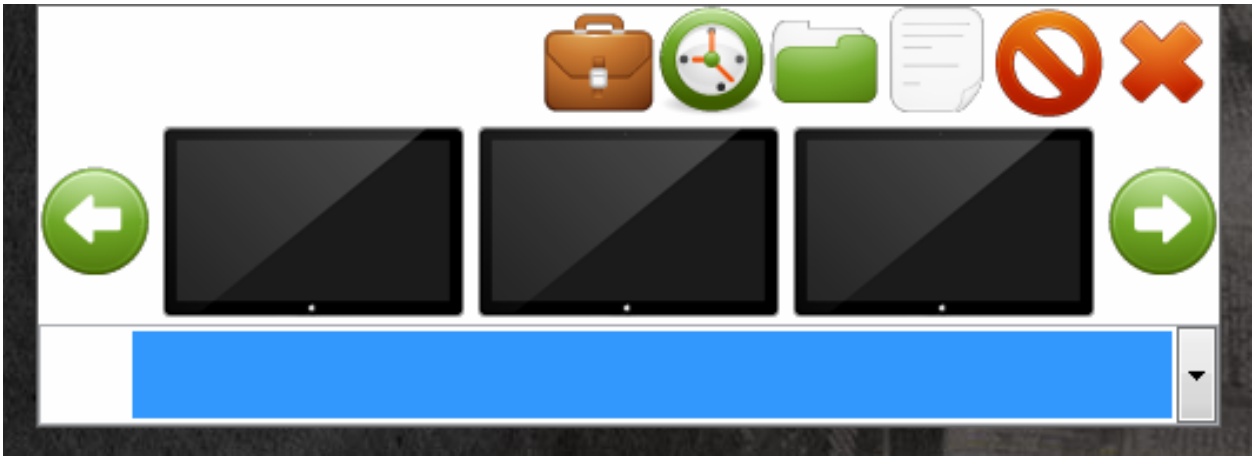

Figure 4.1: The UI of Chimaira; Several icons for different functions.

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 4.1: 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

FIVE

INDICES AND TABLES

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

PYTHON MODULE INDEX