Reference Manual

Generated by Doxygen 1.8.13

Contents

Chapter 1

Dimensionless V1.0

Overview:

Dimensionless is a fully distributed platform for inter device communication across the Internet.

Documentation:

`Online Dimensionless Documentation`

List of Open-Source Projects Used:

- Boost 1.64.0
- CodeLite 10.0.7
- Curl 7.54.0
- GNU IceCat 52.1.0
- Grav 1.2.4
- Libtorrent 1.1.4
- Nginx 1.13.1
- Nmap 7.50
- OpenSSL 1.1.0f
- OpenVPN 2.4.3
- PHP 7.1.6
- Tor 0.3.0.8
- wxWidgets 3.0.3
- Zlib 1.2.8
- And many more... (Such as those used for compilation or otherwise - see Building.md for more details)

License:

(CCC License)

For most intents and purposes the following license is identical to the GNU Public License version 2 (GPLv2) which can be viewed here: https://www.gnu.org/licenses/gpl-2.0.txt

The following license adds a condition that is oriented towards those that are not physically located in a location where the above license is enforceable. If the GPLv2 is currently unenforcable at your current physical location, then allow the users, be they yours or from the Internet judge your compliance and accept their help in fixing any potential issues.

Chapter 2

Building

This page contains information on how to build the application.

Introduction:

Currently the building process for the application was only tested on a Debian GNU/Linux operating system and may fail to work on other operating systems as it may lack certain elements required for them. Due to the time constraints on this project I do not see cross-platform building to be possible, but perhaps at one point in the future it may be done.

Pre-Requisites:

- Boost 1.64.0
- CodeLite 10.0.7
- Curl 7.54.0
- GNU IceCat 52.1.0
- Grav 1.2.4
- Libtorrent 1.1.4
- Nginx 1.13.1
- Nmap 7.50
- OpenSSL 1.1.0f
- OpenVPN 2.4.3
- PHP 7.1.6
- Tor 0.3.0.8
- wxWidgets 3.0.3
- Zlib 1.2.8

Chapter 3

Installation

This page contains information on how to install the application.

Introduction:

Before installing the application, please ensure you check the system requirements to ensure you can run it. Most modern computers and internet connections should be fine to run it.

Successfully Tested System Requirements:

- Operating System: Debian Stretch 9.1 (64 bit) with Backports
- Processor: i7 3770k @ 3.5 GHz
- Random Access Memory: 3x2GB DDR3 Double Channel
- Hard Drives: 1x3TB + 1x1TB + 180GB SSD (*YOU WILL NOT NEED THIS MUCH SPACE, unless you are sharing a lot of large files*)
- Graphics Card: Nvidia GTX 980 4.0 GB (*YOU WILL NOT NEED A GRAPHICS CARD THAT IS THIS POW↩ ERFUL*.)
- Monitor Resolution: 1920x1080 (native)
- Internet Connection: 10 Mbps down, 2 Mbps up
- Time to scan a list of ∼90 IPv4 address ranges (∼90000 discrete IPv4 addresses) in a local town: ∼9 minutes.

Realistic System Requirements:

- A modern processor with multiple cores and threads capable of running multi-threaded wxWidgets GUI applications (Clock speed realisitically in the range 1.0-2.0 GHz)
- A monitor with minimum resolution 1280x720 (720p)
- A decent graphics card that can display GUIs with ease
- An internet connection capable of connecting to the Internet.

- Megabytes of hard drive space (unless your website content is larger).

∗Important note on hardware requirements: I do not have the resources or time to test every possible system out there, hence I provide no guarantees the application will run without flaw, if you have any particular issues you may email me: dev@eclecticdimensions.com and hopefully I will be able to resolve your issues, if not, you may need to have a look at the source code and tailor it to your needs.

Installation:

Build the application from source and run it or download the installer and follow the on-screen prompts.

Setup Images:

TODO

Uninstallation:

Go to the location where the application was installed, run the uninstaller and follow the on-screen prompts. Alternatively run the uninstaller from the control panel (if available).

Uninstallation Images:

TODO

Chapter 4

Usage Tutorial

This page contains information on how to use the application after it has been installed.

Introduction:

Dimensionless is an application that consists of many frames whereby you, the user, can interact with it and perform the tasks you need to do.

Startup Frame:

This is the first frame you will encounter upon starting the application. You must provide your valid username and password credentials and click login or press enter to proceed further into the application.

Main Frame:

This frame is usually the first frame you will meet after logging in. It contains an application log to give you some indication of what processes the application is performing, a connected users list to allow you to interact with, and some buttons to view your personal site as well as push any updates to your connected users.

Search For Users Frame:

This frame allows you to search for users that you can send connection requests to in the hope they become connected users.

Chat Frame:

This frame allows you to chat with other connected users.

Settings Frame:

This frame allows you to view and edit your current settings.

Troubleshooting:

If anything displayed on the application looks like bad data, the application files may be corrupted - please check the application directory's User folder.

Can't login? Ensure your username and password are correct.

Something not working? Not everything is currently fully implemented.

Chapter 5

Internet Scanning

This page contains information on how the application searches for other users on the Internet.

Overview:

The following application makes use of Nmap and GeoIP to scan portions of the Internet for other users. It is unlikely to land you in trouble given the rampant scans going on by malicious malware creators and these scans simply try to obtain the HTTP title of websites that can be hosted at other IP addresses which can be thought of as virtually indistinguishable from accessing a lot of websites quickly. Naturally, given this state of affairs it would be wise to limit your scans and ensure you have as many connected users as possible to spread out the risk and parallelize your searches for faster performance.

Scan Test Results:

The following is a scanning test performed on IPv4 addresses that are supposedly located in a small European town. Approximately 90000 discrete IPv4 addresses (about 90 IPv4 address ranges) are scanned in about 540 seconds (9 minutes). A typical metropolitan city somewhere in America or the United Kingdom may have as many as $\sim$100 times (or more!) this many addresses for specific districts, therefore making it crucial to have a large portion of connected users to aid you in your searches.

Starting Nmap 7.50 (https://nmap.org) at 2017-08-17 16:50 CEST Nmap scan report for 31.178.40.188 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.41.78 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.41.225 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: QNAP

Nmap scan report for 31.178.43.101 Host is up (0.033s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 31.178.43.252 Host is up (0.032s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 31.178.44.219 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 31.178.45.57 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: ipCAM

Nmap scan report for 31.178.47.206 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.48.148 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.48.253 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.54.36 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://31.178.54.36/

Nmap scan report for 31.178.58.5 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 31.178.61.88 Host is up (0.085s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.64.62 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.64.99 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 31.178.66.109 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 31.178.69.175 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 31.178.73.216 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 31.178.74.162 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 31.178.75.83 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://31.178.75.83/

Nmap scan report for 31.178.78.93 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 31.178.78.156 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 31.178.79.66 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 31.178.80.78 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie mo znale zasobu.

Nmap scan report for 31.178.81.66 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 31.178.84.2 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 31.178.89.1 Host is up (0.20s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 31.178.90.59 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 31.178.90.218 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 62.233.130.82 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 62.233.130.86 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 62.233.130.130 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 62.233.130.140 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://62.233.130.↩
140/user.html

Nmap scan report for 62.233.130.142 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error

Nmap scan report for 62.233.131.227 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 62.233.131.234 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: File Not Found

Nmap scan report for 78.88.220.9 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Gargoyle Router Management Utility |_Requested resource was http://78.88.220.9/login.sh

Nmap scan report for 78.88.220.137 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 78.88.220.140 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 78.88.221.129 Host is up (0.018s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: not found

Nmap scan report for 78.88.221.151 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 78.88.221.153 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 78.88.221.245 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 79.184.51.214 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.184.53.161 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 79.184.53.183 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Page not found

Nmap scan report for 79.184.60.22 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Database Error

Nmap scan report for 79.184.60.32 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.184.61.100 Host is up (0.099s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.184.62.165 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 79.184.62.222 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 79.184.63.13 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.184.63.80 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.185.22.9 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.185.87.207 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://79.185.87.207/

Nmap scan report for 79.185.87.209 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.93.26 Host is up (0.099s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object Not Found

Nmap scan report for 79.185.93.33 Host is up (0.090s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.93.51 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.185.93.158 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.94.119 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 79.185.94.152 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 79.185.94.168 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.185.94.173 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.185.94.238 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 79.185.96.40 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Web Configurator

Nmap scan report for 79.185.96.152 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 79.185.96.200 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.96.212 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.96.221 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 79.185.96.237 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.96.245 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.185.140.93 Host is up (0.085s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.140.105 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.185.140.192 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 79.185.226.193 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.185.226.209 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 79.185.226.224 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: D-Link ADSL Router

Nmap scan report for 79.185.226.243 Host is up (0.098s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 79.185.227.13 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.227.33 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.185.227.34 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.185.227.73 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.185.227.162 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 79.185.227.235 Host is up (0.098s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.227.237 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.185.227.251 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.229.29 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.185.229.141 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.185.229.149 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.185.229.214 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 79.187.216.4 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.187.216.6 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.187.216.7 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.187.216.58 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie mo znale zasobu.

Nmap scan report for 79.187.216.142 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 79.187.216.218 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.187.216.254 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Error: 404

Nmap scan report for 79.187.217.14 Host is up (0.099s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.187.217.27 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://79.187.217.↩ 27/user.html

Nmap scan report for 79.187.217.90 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.187.217.236 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.187.217.238 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.187.217.239 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 79.188.62.10 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Protected Object

Nmap scan report for 79.188.62.14 Host is up (0.097s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 79.188.62.72 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.188.62.74 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.188.62.75 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.188.62.78 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.188.63.90 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Apache Tomcat/7.0.28 - Error report

Nmap scan report for 79.188.63.106 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 79.188.63.170 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.188.63.194 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 79.188.63.252 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.188.63.254 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.188.63.255 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 79.191.104.135 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.191.105.13 Host is up (0.098s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 79.191.105.21 Host is up (0.15s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.191.111.49 Host is up (0.099s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 79.191.124.214 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://79.191.124.214↩
:8001/

Nmap scan report for 79.191.124.219 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: IP camera

Nmap scan report for 79.191.135.12 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 79.191.246.101 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 79.191.246.119 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 80.55.152.146 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 80.55.152.194 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 80.55.153.34 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 80.55.153.194 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Cross Error

Nmap scan report for 80.55.153.230 Host is up (0.096s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 80.55.154.58 Host is up (0.098s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=gb2312).

Nmap scan report for 80.55.154.61 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http | http-title: |_Requested resource was /sdiportal/Full.aspx

Nmap scan report for 80.55.154.82 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 - Forbidden: Access is denied.

Nmap scan report for 80.55.154.214 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 80.55.155.50 Host is up (0.093s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: File Not Found

Nmap scan report for 80.55.155.53 Host is up (0.096s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Ko "Sarenka" Wroc

Nmap scan report for 80.55.155.146 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 80.55.155.147 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 80.55.155.148 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be displayed

Nmap scan report for 80.55.155.150 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title:

Nmap scan report for 80.55.156.30 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 80.55.156.44 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 80.55.156.46 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 80.55.156.47 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 80.55.156.62 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 80.55.156.109 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 80.55.156.130 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - B: 404

Nmap scan report for 80.55.156.131 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 80.55.156.132 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 80.55.156.133 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 80.55.156.134 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 80.55.156.146 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 80.55.156.195 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 80.55.157.150 Host is up (0.090s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 80.55.158.22 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 80.55.159.54 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 80.55.159.250 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://80.55.159.250↩
:444/user.html

Nmap scan report for 81.6.128.62 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.128.226 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.128.242 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title:

Nmap scan report for 81.6.128.250 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie znaleziono obiektu!

Nmap scan report for 81.6.129.2 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be displayed

Nmap scan report for 81.6.129.8 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.6.129.12 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.129.14 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: IDEAWEB - B 404

Nmap scan report for 81.6.129.115 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: JBoss Web/7.4.8.Final-redhat-4 - JBWEB000064: Error report

Nmap scan report for 81.6.129.116 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: JBoss Web/7.4.8.Final-redhat-4 - JBWEB000064: Error report

Nmap scan report for 81.6.129.118 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http | http-title: enova365 |_Requested resource was /Home/Error?err=NG↩
VjOWVkOTUtYTY1Ni00NDZkLWFiN2YtNmM4YTliODAwN2Zk

Nmap scan report for 81.6.129.122 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 81.6.129.123 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 81.6.129.124 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 81.6.129.125 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 81.6.133.2 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.3 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.133.4 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.133.5 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.133.11 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.133.17 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.21 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Narodowy Fundusz Zdrowia :: 404

Nmap scan report for 81.6.133.25 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.32 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.35 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Gargoyle Router Management Utility |_Requested resource was http://81.6.133.35/login.sh

Nmap scan report for 81.6.133.36 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.6.133.37 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.42 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.47 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.6.133.↩ 47/user.html

Nmap scan report for 81.6.133.50 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.133.65 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.100 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.133.104 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.133.105 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.133.106 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.107 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.108 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.111 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.112 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.113 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.133.114 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.129 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.137 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://81.6.133.137/

Nmap scan report for 81.6.133.142 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 81.6.133.144 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.145 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.6.133.146 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.147 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.133.151 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.153 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.156 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.133.161 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.162 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.6.133.167 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.6.133.168 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.133.170 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.133.172 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.6.133.173 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.179 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.133.180 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Open Webif

Nmap scan report for 81.6.133.184 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.190 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Authorization Required

Nmap scan report for 81.6.133.195 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.133.197 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 81.6.133.199 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.201 Host is up (0.082s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.133.213 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.226 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.133.228 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 81.6.133.249 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.134.195 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.6.135.113 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.2 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.12 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.13 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.17 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.22 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.30 Host is up (0.018s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.31 Host is up (0.018s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://ox.interbit.↩ com.pl/

Nmap scan report for 81.6.136.50 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: ZnajdzFoto.pl - wyszukiwarka zdj

Nmap scan report for 81.6.136.56 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.57 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.58 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.59 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: ZnajdzFoto.pl - wyszukiwarka zdj

Nmap scan report for 81.6.136.60 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.63 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.64 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.100 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.101 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Eksadata sp. z o.o. » Strona nie zosta znaleziona

Nmap scan report for 81.6.136.102 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: PASSport - Interaktywny system zarz aktywno spor...

Nmap scan report for 81.6.136.113 Host is up (0.023s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.114 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.115 Host is up (0.022s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.116 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.117 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.118 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.119 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.120 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: ERROR 404 - Not Found!

Nmap scan report for 81.6.136.121 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.122 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.123 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.124 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.136.125 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.137.194 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.137.195 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.137.196 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Site doesn't have a title (text/html; charset=UTF-8). |_↩ Requested resource was http://81.6.137.196/systemLogin/action/login

Nmap scan report for 81.6.137.198 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.137.201 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://smartbiuro24.↩ pl/user.html

Nmap scan report for 81.6.137.206 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.bizin.↩ pl/user.html

Nmap scan report for 81.6.137.242 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.137.244 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.137.246 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.139.8 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.6.139.↩ 8/user.html

Nmap scan report for 81.6.139.67 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.139.84 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.139.90 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 81.6.139.92 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 81.6.139.114 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.141.8 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 81.6.141.9 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error

Nmap scan report for 81.6.141.11 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 81.6.141.13 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 81.6.141.14 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.141.16 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.141.19 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 81.6.141.25 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 81.6.141.26 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 81.6.141.27 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.6.141.↩ 27/index.php/user.html?

Nmap scan report for 81.6.141.30 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.141.31 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 81.6.141.34 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.141.44 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: B logowania

Nmap scan report for 81.6.141.46 Host is up (0.024s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.6.141.48 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.6.141.49 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error

Nmap scan report for 81.6.141.51 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.6.141.52 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 81.6.141.53 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Cross Error

Nmap scan report for 81.6.141.55 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.141.60 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.6.141.61 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Cross Error

Nmap scan report for 81.6.141.64 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.141.69 Host is up (0.025s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.6.141.71 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.6.141.74 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.6.141.76 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.141.81 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.6.141.234 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.142.89 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.143.12 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.6.187.13 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 81.6.187.33 Host is up (0.14s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 81.6.187.51 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=UTF-8).

Nmap scan report for 81.6.187.55 Host is up (0.082s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.187.67 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.6.187.95 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.6.187.212 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 - Dost zabroniony: odmowa dost.

Nmap scan report for 81.6.190.10 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.6.191.78 Host is up (0.028s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.26.0.7 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.0.8 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.tu.kielce.↩ pl/user.html

Nmap scan report for 81.26.0.11 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.tu.kielce.↩ pl/user.html

Nmap scan report for 81.26.0.19 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 81.26.0.20 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.tu.kielce.↩ pl/user.html

Nmap scan report for 81.26.0.25 Host is up (0.093s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.0.35 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://login.tu.↩ kielce.pl/user.html

Nmap scan report for 81.26.0.38 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.0.↩ 38/user.html

Nmap scan report for 81.26.0.41 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.26.0.43 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Otwarty System Antyplagiatowy |_Requested resource was http://81.26.0.43/auth/login

Nmap scan report for 81.26.0.45 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.0.47 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.0.49 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.0.53 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.26.0.115 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 81.26.0.116 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 81.26.0.193 Host is up (0.082s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.0.242 Host is up (0.032s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.3.14 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.3.33 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.26.4.81 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://stgc.tu.kielce.↩ pl/user.html

Nmap scan report for 81.26.5.23 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.26.5.68 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.5.69 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.5.79 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.5.80 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 81.26.5.115 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: JBossWeb/2.0.1.GA - Error report

Nmap scan report for 81.26.6.3 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.4 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.5 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.6 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.8 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.12 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.14 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 502 Bad Gateway

Nmap scan report for 81.26.6.16 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.6.↩
16/user.html

Nmap scan report for 81.26.6.19 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.20 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.6.110 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.8.8 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.8.12 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.26.8.26 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.26.8.142 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.8.149 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 81.26.8.157 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
157/user.html

Nmap scan report for 81.26.8.160 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
160/user.html

Nmap scan report for 81.26.8.161 Host is up (0.029s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Digital Library of Jan Kochanowski University - ERROR!

Nmap scan report for 81.26.8.162 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
162/user.html

Nmap scan report for 81.26.8.163 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Apache Tomcat/7.0.34 - Error report

Nmap scan report for 81.26.8.164 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.8.165 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
165/user.html

Nmap scan report for 81.26.8.166 Host is up (0.031s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
166/user.html

Nmap scan report for 81.26.8.175 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
175/user.html

Nmap scan report for 81.26.8.178 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.8.↩
178/user.html

Nmap scan report for 81.26.8.181 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.8.188 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 81.26.9.2 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.9.2/user.↩ html

Nmap scan report for 81.26.9.3 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.9.3/user.↩ html

Nmap scan report for 81.26.9.4 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.9.4/user.↩ html

Nmap scan report for 81.26.9.84 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.9.138 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://radio.ujk.edu.↩ pl/login

Nmap scan report for 81.26.15.2 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.15.3 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.15.130 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.26.19.120 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.26.19.163 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 81.26.19.164 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 81.26.19.187 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 81.26.19.189 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 81.26.20.2 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.26.20.3 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Portal Dydaktyczny Katedry Informatyki — ...

Nmap scan report for 81.26.20.4 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.20.5 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Access forbidden!

Nmap scan report for 81.26.20.6 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.20.7 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.20.8 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.20.11 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Sign in GitLab |_Requested resource was http://81.↩ 26.20.11/users/sign_in

Nmap scan report for 81.26.20.32 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.23.171 Host is up (0.082s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie mo znale tej strony

Nmap scan report for 81.26.23.194 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://sipws.konecki.↩ powiat.pl/geoportal/Full.aspx

Nmap scan report for 81.26.23.195 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.26.23.238 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.26.23.251 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.23.253 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.27.5 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://mail.staszow.↩
pl/user.html

Nmap scan report for 81.26.27.65 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://sip.staszowski.↩
eu/geoportal/Full.aspx

Nmap scan report for 81.26.27.66 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.26.28.6 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: IIS 8.5 Detailed Error - 404.0 - Not Found

Nmap scan report for 81.26.28.8 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.10 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://sip.powiat.↩
kielce.pl/geoportal/Full.aspx

Nmap scan report for 81.26.28.11 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://sipws-kielce.↩
e-swietokrzyskie.pl/EGP/Full.aspx

Nmap scan report for 81.26.28.29 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.26.28.30 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.36 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie mo odnale strony

Nmap scan report for 81.26.28.47 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.54 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Apache Tomcat/7.0.28 - Error report

Nmap scan report for 81.26.28.67 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - B: 404

Nmap scan report for 81.26.28.99 Host is up (0.031s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - B: 404

Nmap scan report for 81.26.28.100 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Digital Library - Resource not found.

Nmap scan report for 81.26.28.101 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.103 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.28.106 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - B: 404

Nmap scan report for 81.26.28.130 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.131 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://pod.wseip.edu.↩pl/user.html

Nmap scan report for 81.26.28.134 Host is up (0.032s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.142 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.170 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://e.wseip.edu.↩pluser.html

Nmap scan report for 81.26.28.210 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.212 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Kielce

Nmap scan report for 81.26.28.233 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Authorization Required

Nmap scan report for 81.26.28.234 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.235 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie odnaleziono witryny

Nmap scan report for 81.26.28.236 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 81.26.28.241 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - B: 404

Nmap scan report for 81.26.28.245 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.28.246 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.29.33 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Szlak Green Velo

Nmap scan report for 81.26.29.34 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.26.29.35 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: GreenVelo | Chwilowa przerwa

Nmap scan report for 81.26.29.37 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: eWykazy - elektroniczne wykazy zawieraj informacje i dane ...

Nmap scan report for 81.26.29.38 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Status - Wrota Portal Wojew ...

Nmap scan report for 81.26.29.41 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.29.42 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.29.43 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - B: 404

Nmap scan report for 81.26.29.44 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Prace administracyjne

Nmap scan report for 81.26.29.45 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.26.29.46 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.29.47 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Urz Marsza Wojew - B...

Nmap scan report for 81.26.29.48 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Biuletyn Informacji Publicznej Wojew ...

Nmap scan report for 81.26.29.49 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Status - Regionalny System BIP

Nmap scan report for 81.26.29.50 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://piwik.↩ wrota-swietokrzyskie.pl/piwik/

Nmap scan report for 81.26.29.51 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Apache Tomcat/8.0.42 - Error report

Nmap scan report for 81.26.29.52 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.26.29.53 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://sipws.↩ e-swietokrzyskie.pl/EGP/Full.aspx

Nmap scan report for 81.26.29.54 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.26.29.55 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://asip.↩ e-swietokrzyskie.pl/ara/Form.aspx

Nmap scan report for 81.26.29.57 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Serwisy Regionalnego Programu Operacyjnego Wojew ...

Nmap scan report for 81.26.29.58 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 81.26.29.59 Host is up (0.093s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /index.php?option=com_↩ content&view=article&id=75

Nmap scan report for 81.26.29.60 Host is up (0.093s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 81.26.29.65 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html;charset=ISO-8859-1).

Nmap scan report for 81.26.29.66 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html;charset=ISO-8859-1).

Nmap scan report for 81.26.29.74 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.29.↩ 74/user.html

Nmap scan report for 81.26.29.77 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.26.29.78 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.29.98 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.26.29.101 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.29.102 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.29.176 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http | http-title: 404 - Nie odnaleziono pliku lub katalogu. |_Requested resource was http://81.26.29.176/msi/user.html

Nmap scan report for 81.26.29.232 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.26.29.242 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.30.2 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.30.3 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.30.12 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: QNAP

Nmap scan report for 81.26.30.13 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.26.30.19 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.26.30.↩ 19/user.html

Nmap scan report for 81.26.30.22 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.30.23 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 81.26.30.24 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.30.27 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.30.28 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.26.30.29 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 81.26.30.30 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.26.31.1 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.2 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.3 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.4 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.5 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.6 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 81.26.31.7 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.8 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.9 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.10 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.11 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.12 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.13 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.14 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.15 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.16 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.17 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.18 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.19 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be found

Nmap scan report for 81.26.31.20 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be found

Nmap scan report for 81.26.31.21 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.22 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.23 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.24 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.25 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.26 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.27 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.28 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.29 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.30 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.31 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.32 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.33 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.34 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.35 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.36 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.37 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.38 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.39 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.40 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.41 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.42 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.43 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.44 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.45 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.46 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.47 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.48 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.49 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.50 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.51 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.52 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.53 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.54 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.55 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.56 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.57 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.58 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.59 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.60 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.61 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.62 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.63 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.64 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.65 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.66 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.67 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.68 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.69 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.70 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.71 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.72 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.73 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.74 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.75 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.76 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.77 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.78 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.79 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.80 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.81 Host is up (0.033s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.82 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.83 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.84 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.85 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.86 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.87 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.88 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.89 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.90 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.91 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.92 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.93 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.94 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.95 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.96 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.97 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.98 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.99 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.100 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.101 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.102 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.103 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.104 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.105 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.106 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.108 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.109 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.110 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.111 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.112 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.113 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.114 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.115 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.116 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.117 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.118 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.119 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.120 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.121 Host is up (0.033s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.122 Host is up (0.031s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.123 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.124 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.125 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.126 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.127 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.128 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.129 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.130 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.131 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.132 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.133 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.134 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.135 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.136 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.137 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.138 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.139 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.140 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.141 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.142 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.143 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.144 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.145 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.146 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.147 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.148 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.149 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.150 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.151 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.152 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.153 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.154 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.155 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.156 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.157 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.158 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.159 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.160 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.161 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.162 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.163 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.164 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.165 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.166 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.167 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.168 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.169 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.170 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.171 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.172 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.173 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.174 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.175 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.176 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.177 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.178 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.179 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.180 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.181 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.182 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.183 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.184 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.185 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.186 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.187 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.188 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.189 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.190 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.191 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.192 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.193 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.194 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.195 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.196 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.197 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.198 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.199 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.200 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.201 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.202 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.203 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.204 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.205 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.206 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.207 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.208 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.209 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.210 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.211 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.212 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.213 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.214 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.215 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.216 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.217 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.218 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.219 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.220 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.221 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.222 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.223 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.224 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.225 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.226 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.227 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.228 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.229 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.230 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.231 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.232 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.233 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.234 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.235 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.237 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.238 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.239 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.240 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.241 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.242 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.243 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.244 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.245 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.246 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.247 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.248 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.250 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.251 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.252 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.26.31.253 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.4 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.5 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.6 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.7 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.10 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be found

Nmap scan report for 81.219.176.13 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error

Nmap scan report for 81.219.176.52 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie masz uprawnie do ogl tej strony

Nmap scan report for 81.219.176.61 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.62 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.121 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.122 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 81.219.176.123 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.124 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.125 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.126 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.132 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.176.137 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.153 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.161 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.162 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://81.219.176.164

Nmap scan report for 81.219.176.164 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 81.219.176.166 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.219.180.70 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://81.219.180.↩
70/user.html

Nmap scan report for 81.219.183.50 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.183.66 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.183.86 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 81.219.183.162 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 81.219.183.190 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 81.219.188.6 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.26 Host is up (0.028s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.94 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.97 Host is up (0.090s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.99 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.100 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.102 Host is up (0.090s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.103 Host is up (0.090s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.104 Host is up (0.097s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.105 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.106 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.107 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.108 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.112 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.113 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.114 Host is up (0.033s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.115 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.116 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.117 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.120 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.122 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.225 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.228 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.219.188.241 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.249 Host is up (0.029s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.188.253 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.1 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.36 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.169 Host is up (0.097s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.173 Host is up (0.090s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.184 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.194 Host is up (0.14s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.196 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.206 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.207 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.210 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.211 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.215 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.217 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.219 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.223 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.224 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.226 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.230 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.231 Host is up (0.095s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.232 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.236 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.237 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.238 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.239 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.241 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.245 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.247 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.248 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.253 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.189.254 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Mimosa A5 |_Requested resource was http://81.219.↩ 189.254/login

Nmap scan report for 81.219.208.8 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.208.36 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.208.40 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.208.187 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.208.192 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.208.253 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.219.209.10 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.16 Host is up (0.16s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.118 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.145 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.168 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.170 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.199 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.232 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.233 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.235 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.236 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.238 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.239 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.240 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.241 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.243 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.244 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.246 Host is up (0.092s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.247 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.248 Host is up (0.033s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.209.249 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.210.3 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Mimosa A5 |_Requested resource was http://81.219.↩
210.3/login

Nmap scan report for 81.219.210.4 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.210.6 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.210.10 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.219.210.16 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 81.219.210.27 Host is up (0.027s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 81.219.210.86 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 81.219.210.154 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.219.210.193 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 81.219.210.201 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 81.219.210.220 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 83.6.54.185 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 83.6.54.203 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 83.6.54.226 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 83.6.54.233 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Page not found

Nmap scan report for 83.6.54.242 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.15.48.117 Host is up (0.15s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.49.86 Host is up (0.096s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Unauthorized

Nmap scan report for 83.15.49.154 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 83.15.49.218 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.49.235 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.50.22 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 83.15.50.82 Host is up (0.082s latency).

PORT STATE SERVICE 80/tcp open http |_http-title:

Nmap scan report for 83.15.50.118 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.15.50.126 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.15.50.142 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 83.15.50.178 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.51.194 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.52.35 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.15.53.102 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 83.15.53.178 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.15.53.181 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 83.15.53.182 Host is up (0.098s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.54.166 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 83.15.55.34 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.38 Host is up (0.098s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.54 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.58 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.106 Host is up (0.097s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.107 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.109 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.110 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.55.124 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 83.15.55.238 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 83.15.60.70 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.15.60.186 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 83.15.61.76 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.15.61.181 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Runtime Error |_Requested resource was /Home/↩ Error?err=OWMwYzQ1OTEtYWE0My00ZjIwLWIxYmMtZjdlNTM5ZjE1ZGE0

Nmap scan report for 83.15.61.220 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 83.15.61.222 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 83.15.61.223 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 83.15.62.142 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 83.15.63.50 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 83.15.63.74 Host is up (0.097s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.15.63.118 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=UTF-8).

Nmap scan report for 83.15.63.170 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 83.15.63.206 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 83.18.72.186 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.18.72.246 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 83.18.73.218 Host is up (0.082s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: File Not Found

Nmap scan report for 83.18.73.238 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 83.18.74.86 Host is up (0.13s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error

Nmap scan report for 83.18.74.94 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 83.18.74.210 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.18.74.213 Host is up (0.091s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.18.74.235 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://manhattan.na.↩ pl/user.html

Nmap scan report for 83.18.75.18 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://83.18.75.18↩
:443/user.html

Nmap scan report for 83.18.75.58 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.18.75.138 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.18.75.196 Host is up (0.16s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.18.75.198 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.18.75.199 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.18.76.219 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://83.18.76.↩
219/user.html

Nmap scan report for 83.18.77.10 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.18.78.2 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.18.78.82 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 83.18.78.83 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 83.18.78.186 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.18.79.2 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Login Incorrect

Nmap scan report for 83.18.79.3 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 83.238.90.1 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 83.238.90.3 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.238.90.4 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 83.238.90.5 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Wiadomo administracyjna

Nmap scan report for 83.238.90.6 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Wiadomo administracyjna

Nmap scan report for 83.238.91.131 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 83.238.91.136 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 83.238.91.149 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 83.238.91.245 Host is up (0.032s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 88.156.66.163 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 88.156.66.202 Host is up (0.100s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 88.156.67.67 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 88.156.67.156 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 88.156.67.166 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 88.156.67.182 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Linksys PAP2 Configuration

Nmap scan report for 88.156.252.2 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 88.156.252.4 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 88.156.252.5 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 88.156.253.13 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 - Forbidden

Nmap scan report for 88.156.253.21 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 88.156.253.125 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 88.156.253.142 Host is up (0.018s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 88.156.253.161 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 88.156.253.219 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 88.156.254.174 Host is up (0.033s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: not found

Nmap scan report for 88.156.254.178 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 88.156.254.190 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 88.156.254.219 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 89.78.109.225 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 89.78.110.121 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 89.78.110.139 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 89.78.110.156 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 89.78.110.210 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 91.192.228.73 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.192.228.88 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.192.228.145 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 91.192.228.219 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 91.192.228.253 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 91.193.41.3 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Strona kt szukasz nie istnieje (404)

Nmap scan report for 91.193.41.4 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 91.193.41.9 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 91.193.41.11 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.193.41.12 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 91.193.41.16 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 91.193.41.20 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=gb2312).

Nmap scan report for 91.193.41.21 Host is up (0.097s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.193.41.22 Host is up (0.099s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 91.193.41.23 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/xml; charset=utf-8).

Nmap scan report for 91.193.41.58 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 91.193.41.60 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 91.193.41.82 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.193.41.88 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 91.193.41.251 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.193.41.↩
251/user.html

Nmap scan report for 91.193.41.252 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.193.41.252↩
:10443/user.html

Nmap scan report for 91.195.46.15 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 91.195.46.17 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.195.46.19 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Cross Error

Nmap scan report for 91.195.46.21 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 91.195.46.23 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 91.195.46.24 Host is up (0.085s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
24/user.html

Nmap scan report for 91.195.46.27 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Waiting...

Nmap scan report for 91.195.46.29 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.29↩
:443/user.html

Nmap scan report for 91.195.46.30 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 91.195.46.36 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Waiting...

Nmap scan report for 91.195.46.39 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
39/user.html

Nmap scan report for 91.195.46.40 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 91.195.46.47 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 91.195.46.51 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
51/user.html

Nmap scan report for 91.195.46.52 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.195.46.59 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
59/user.html

Nmap scan report for 91.195.46.63 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
63/user.html

Nmap scan report for 91.195.46.65 Host is up (0.031s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.195.46.66 Host is up (0.028s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 91.195.46.70 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
70/user.html

Nmap scan report for 91.195.46.80 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.195.46.87 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 91.195.46.89 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 91.195.46.98 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.195.46.113 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.195.46.145 Host is up (0.046s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
145/user.html

Nmap scan report for 91.195.46.161 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.46.↩
161/user.html

Nmap scan report for 91.195.46.178 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 91.195.47.6 Host is up (0.12s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.195.47.↩
6/user.html

Nmap scan report for 91.195.47.9 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 91.195.47.81 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 91.195.56.7 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Strony nie znaleziono

Nmap scan report for 91.195.56.8 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: netcity.pl ++ Internet Service Provider tel. 41 3029000

Nmap scan report for 91.195.56.14 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.195.56.15 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.195.57.90 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 91.195.57.104 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 91.195.57.107 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title:

Nmap scan report for 91.215.46.2 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 Unauthorized

Nmap scan report for 91.215.46.3 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.215.46.10 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://91.215.46.↩ 10/user.html

Nmap scan report for 91.215.46.12 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: An Error Occurred: Not Found

Nmap scan report for 91.215.46.43 Host is up (0.027s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 91.215.46.85 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 91.215.46.94 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found !!!

Nmap scan report for 91.215.46.108 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 91.215.46.124 Host is up (0.032s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 94.177.135.1 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 94.177.135.3 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 94.177.135.69 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://94.177.135.↩ 69/user.html

Nmap scan report for 94.177.135.129 Host is up (0.092s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.136 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 94.177.135.137 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Rocket.Chat

Nmap scan report for 94.177.135.142 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.144 Host is up (0.085s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Login Incorrect

Nmap scan report for 94.177.135.145 Host is up (0.099s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 94.177.135.147 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 94.177.135.148 Host is up (0.029s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 94.177.135.149 Host is up (0.085s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 94.177.135.154 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Gargoyle Router Management Utility

Nmap scan report for 94.177.135.162 Host is up (0.081s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.163 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 94.177.135.165 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.171 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 94.177.135.172 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.175 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.176 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 94.177.135.177 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 94.177.135.178 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Gargoyle Router Management Utility |_Requested resource was http://94.177.135.178/login.sh

Nmap scan report for 94.177.135.180 Host is up (0.042s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Gargoyle Router Management Utility |_Requested resource was http://94.177.135.180/login.sh

Nmap scan report for 94.177.135.181 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.182 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Gargoyle Router Management Utility |_Requested resource was http://94.177.135.182/login.sh

Nmap scan report for 94.177.135.185 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.193 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 94.177.135.195 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Gargoyle Router Management Utility

Nmap scan report for 94.177.135.196 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Gargoyle Router Management Utility |_Requested resource was http://94.177.135.196/login.sh

Nmap scan report for 94.177.135.198 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.200 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Gargoyle Router Management Utility

Nmap scan report for 94.177.135.203 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error

Nmap scan report for 94.177.135.204 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 94.177.135.220 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 94.177.135.223 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 94.177.135.228 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.230 Host is up (0.034s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 94.177.135.235 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://termed24.↩
eu/user.html

Nmap scan report for 94.177.135.239 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 94.177.135.243 Host is up (0.088s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 94.177.135.244 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 94.177.135.246 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 94.177.135.251 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 94.177.135.253 Host is up (0.031s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 94.177.135.254 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 178.235.112.40 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 178.235.112.42 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 178.235.112.130 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 178.235.112.175 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 178.235.113.63 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 178.235.113.87 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 178.235.113.153 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Sign in GitLab |_Requested resource was http://178.↩
235.113.153/users/sign_in

Nmap scan report for 178.235.113.167 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 178.235.113.198 Host is up (0.049s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 178.235.114.192 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 178.235.114.215 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 178.235.115.76 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 178.235.115.78 Host is up (0.035s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 178.235.115.91 Host is up (0.028s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 178.235.115.167 Host is up (0.023s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 178.235.115.213 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap scan report for 178.235.115.240 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 178.235.118.60 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 178.235.118.62 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 178.235.118.210 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 178.235.118.255 Host is up (0.030s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 178.235.119.2 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 178.235.119.119 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 178.235.119.146 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Cross Error

Nmap scan report for 178.235.119.171 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 178.235.119.203 Host is up (0.031s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 185.23.148.2 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.23.148.5 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.23.148.6 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Wiadomo administracyjna

Nmap scan report for 185.23.148.147 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 185.23.149.9 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 185.23.149.78 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 185.23.149.121 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.23.149.199 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 185.23.150.57 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Site or Page Not Found

Nmap scan report for 185.23.150.62 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.23.150.95 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 185.23.150.166 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 185.23.150.208 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 185.23.151.34 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 185.23.151.80 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 185.23.151.191 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 185.23.151.199 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 185.23.151.207 Host is up (0.056s latency).

PORT STATE SERVICE 80/tcp open http |_http-title:

Nmap scan report for 185.23.151.221 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title:

Nmap scan report for 185.23.151.246 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to /

Nmap scan report for 185.69.0.33 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 185.69.0.53 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 185.69.0.68 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://dziekanat.ws.↩ edu.pl/user.html

Nmap scan report for 185.69.0.69 Host is up (0.085s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://rekrutacja.ws.↩ edu.pl/user.html

Nmap scan report for 185.69.0.70 Host is up (0.11s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.0.75 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html).

Nmap scan report for 185.69.0.82 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://185.69.0.↩ 82/user.html

Nmap scan report for 185.69.0.84 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://185.69.0.↩ 84/user.html

Nmap scan report for 185.69.0.91 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=iso-8859-1).

Nmap scan report for 185.69.0.115 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.0.130 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie mo znale tej strony

Nmap scan report for 185.69.0.133 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.0.171 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Nie mo znale tej strony

Nmap scan report for 185.69.0.172 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be found

Nmap scan report for 185.69.0.174 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.0.196 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Wirtualny Dziekanat - B |_Requested resource was http↩
://185.69.0.196/Shared/Error?status=404

Nmap scan report for 185.69.0.211 Host is up (0.089s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.1.124 Host is up (0.086s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://185.69.1.↩
124/user.html

Nmap scan report for 185.69.1.125 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.1.139 Host is up (0.092s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.1.140 Host is up (0.087s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.69.1.161 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 185.72.32.31 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.124.13 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 193.19.124.39 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: FortiManager Error

Nmap scan report for 193.19.124.42 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://ecorp.bskielce.↩
com.pl/homenet-webapp/

Nmap scan report for 193.19.124.66 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://193.19.124.↩
66/user.html

Nmap scan report for 193.19.124.154 Host is up (0.094s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 193.19.124.170 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://193.19.124.170↩
:4430/user.html

Nmap scan report for 193.19.124.211 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 193.19.124.218 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 193.19.124.221 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Access forbidden!

Nmap scan report for 193.19.125.26 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 193.19.125.52 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page cannot be displayed

Nmap scan report for 193.19.125.80 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 193.19.125.81 Host is up (0.059s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://exchange.↩
cersanit.com.pl/owa

Nmap scan report for 193.19.125.82 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Rovese S.A. Authentication

Nmap scan report for 193.19.125.84 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.bluenet.↩
cersanit.com.pl

Nmap scan report for 193.19.125.88 Host is up (0.029s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 193.19.125.89 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 193.19.125.90 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://tms.rovese.ro/R↩
OUAOptiengineWeb/user.html

Nmap scan report for 193.19.125.91 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - File or directory not found.

Nmap scan report for 193.19.125.99 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.100 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.101 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.102 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.103 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.104 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.106 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.107 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.108 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.109 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.117 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.bluenet.↩ cersanit.com.pl

Nmap scan report for 193.19.125.131 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 193.19.125.195 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 193.19.125.203 Host is up (0.040s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://193.19.125.↩ 203/user.html

Nmap scan report for 193.19.125.250 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 193.19.125.252 Host is up (0.048s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: B: 404 Strona nie istnieje

Nmap scan report for 193.19.125.253 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 193.24.200.48 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Not Found

Nmap scan report for 193.24.202.138 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Login Incorrect

Nmap scan report for 193.24.202.143 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Unauthorized

Nmap scan report for 193.24.202.178 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Unauthorized

Nmap scan report for 193.24.202.181 Host is up (0.054s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Login Incorrect

Nmap scan report for 193.24.202.208 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 193.24.203.120 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 194.181.128.69 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.128.71 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 194.181.128.101 Host is up (0.039s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Unauthorized

Nmap scan report for 194.181.128.113 Host is up (0.064s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.128.170 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 194.181.128.171 Host is up (0.093s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 194.181.128.214 Host is up (0.018s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: The page is not found

Nmap scan report for 194.181.128.226 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 194.181.128.246 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.129.2 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.bizin.↩ pl/user.html

Nmap scan report for 194.181.129.11 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.bizin.↩ pl/user.html

Nmap scan report for 194.181.129.12 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.bizin.↩ pl/user.html

Nmap scan report for 194.181.129.20 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.129.21 Host is up (0.071s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://www.abctrack.↩ pl/user.html

Nmap scan report for 194.181.129.22 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.129.↩ 22/user.html

Nmap scan report for 194.181.129.23 Host is up (0.076s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Projekt |_Requested resource was http://194.181.↩ 129.23/systemLogin/action/login

Nmap scan report for 194.181.129.24 Host is up (0.079s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Site doesn't have a title (text/html; charset=UTF-8). |_↩ Requested resource was http://194.181.129.24/systemLogin/action/login

Nmap scan report for 194.181.129.25 Host is up (0.083s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.129.26 Host is up (0.023s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.129.27 Host is up (0.022s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://www.poczta.↩
pcet.pl/user.html

Nmap scan report for 194.181.129.28 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.129.↩
28/user.html

Nmap scan report for 194.181.129.30 Host is up (0.027s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to http://www.security-service.↩
pami.pl/user.html

Nmap scan report for 194.181.129.74 Host is up (0.029s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 194.181.129.82 Host is up (0.061s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 File Not Found

Nmap scan report for 194.181.129.129 Host is up (0.050s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.129.129↩
:443/index.php/user.html?

Nmap scan report for 194.181.129.130 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 - Dost zabroniony: odmowa dost.

Nmap scan report for 194.181.129.170 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Nie odnaleziono pliku lub katalogu.

Nmap scan report for 194.181.129.227 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.129.242 Host is up (0.041s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.130.19 Host is up (0.070s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=utf-8).

Nmap scan report for 194.181.130.129 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 194.181.130.130 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: hosting1.pl - B 404 - Error 404

Nmap scan report for 194.181.130.131 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.130.132 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.130.133 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.130.150 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Index_Page |_Requested resource was /r49435,/adv,/index.html

Nmap scan report for 194.181.130.152 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=utf-8).

Nmap scan report for 194.181.130.161 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 194.181.130.230 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: File Not Found

Nmap scan report for 194.181.130.254 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.130.254↩
:443/index.php/user.html?

Nmap scan report for 194.181.131.67 Host is up (0.084s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 194.181.131.69 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: IP camera

Nmap scan report for 194.181.131.71 Host is up (0.058s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Document Error: Not Found

Nmap scan report for 194.181.131.129 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 194.181.131.130 Host is up (0.055s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 194.181.131.131 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 194.181.131.146 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.131.148 Host is up (0.072s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.131.↩ 148/user.html

Nmap scan report for 194.181.131.190 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.131.194 Host is up (0.052s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 194.181.131.242 Host is up (0.044s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 194.181.134.2 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 - Nie odnaleziono pliku lub katalogu.

Nmap scan report for 194.181.134.104 Host is up (0.036s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.134.124 Host is up (0.038s latency).

PORT STATE SERVICE 80/tcp open http | http-title: Error 404 - Page Not Found |_Requested resource was /nocookies.html

Nmap scan report for 194.181.134.196 Host is up (0.053s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 194.181.134.197 Host is up (0.073s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Not Found

Nmap scan report for 194.181.134.202 Host is up (0.077s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 194.181.134.230 Host is up (0.026s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.134.242 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.134.246 Host is up (0.045s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.135.150 Host is up (0.078s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.135.198 Host is up (0.075s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 194.181.138.26 Host is up (0.019s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 194.181.138.29 Host is up (0.062s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 194.181.138.186 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 194.181.138.190 Host is up (0.051s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.138.230 Host is up (0.025s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 194.181.138.238 Host is up (0.047s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Object not found!

Nmap scan report for 194.181.139.1 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.2 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.3 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.4 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.5 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.6 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.7 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.8 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.9 Host is up (0.068s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 194.181.139.33 Host is up (0.074s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title (text/html; charset=iso-8859-1).

Nmap scan report for 194.181.139.34 Host is up (0.060s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.139.↩ 34/user.html

Nmap scan report for 194.181.139.40 Host is up (0.021s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 194.181.139.54 Host is up (0.043s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.139.↩ 54/user.html

Nmap scan report for 194.181.139.55 Host is up (0.037s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Error 404: Not Found

Nmap scan report for 194.181.139.56 Host is up (0.10s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://194.181.139.↩ 56/user.html

Nmap scan report for 213.77.108.4 Host is up (0.065s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Did not follow redirect to https://213.77.108.↩ 4/user.html

Nmap scan report for 213.77.108.139 Host is up (0.069s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 213.77.108.163 Host is up (0.057s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 213.77.108.185 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Site doesn't have a title.

Nmap scan report for 213.77.108.189 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: Login Incorrect

Nmap scan report for 213.77.108.191 Host is up (0.080s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 404 Not Found

Nmap scan report for 213.77.108.247 Host is up (0.063s latency).

PORT STATE SERVICE 80/tcp open http

Nmap scan report for 213.77.117.210 Host is up (0.066s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 401 - Unauthorized

Nmap scan report for 213.77.117.249 Host is up (0.067s latency).

PORT STATE SERVICE 80/tcp open http |_http-title: 403 Forbidden

Nmap done: 92354 IP addresses (92354 hosts up) scanned in 535.39 seconds

Chapter 6

Contributing

This page contains information on how to contribute to the development of this application.

How can you contribute?

- Submit patches to add features, improve and fix the source code of this application.
- Improve this technical documentation to be more complete.
- Spread the word and introduce more users to this application.

Chapter 7

Namespace Index

7.1 Namespace List

Here is a list of all namespaces with brief descriptions:

DimensionlessFrames . . . . . ??
DimensionlessFunctionality . . . . . ??
DimensionlessFunctionality::ApplicationLogic . . . . . ??
DimensionlessFunctionality::Compression . . . . . ??
DimensionlessFunctionality::Cryptography . . . . . ??
DimensionlessFunctionality::Networking . . . . . ??
DimensionlessFunctionality::Networking::Curl . . . . . ??
DimensionlessFunctionality::Networking::GeoIP . . . . . ??
DimensionlessFunctionality::Networking::Libtorrent . . . . . ??
DimensionlessFunctionality::Networking::Libtorrent::UPnP . . . . . ??
DimensionlessFunctionality::Networking::Multicast . . . . . ??
DimensionlessFunctionality::Networking::Multicast::Receiver . . . . . ??
DimensionlessFunctionality::Networking::Multicast::Sender . . . . . ??
DimensionlessFunctionality::Networking::Nmap . . . . . ??
DimensionlessFunctionality::Networking::OpenVPN . . . . . ??
DimensionlessFunctionality::Networking::PHPFPM . . . . . ??
DimensionlessFunctionality::Networking::Tor . . . . . ??
DimensionlessFunctionality::Networking::WebBrowser . . . . . ??
DimensionlessFunctionality::Networking::WebServer . . . . . ??
DimensionlessFunctionality::UtilityFunctions . . . . . ??
DimensionlessFunctionality::Validation . . . . . ??
DimensionlessFunctionality::XML . . . . . ??

Chapter 8

Hierarchical Index

8.1 Class Hierarchy

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

DimensionlessFrames::ChatFrame::ChatFrameThreadHandler . . . ??
DimensionlessFunctionality::ApplicationLogic::CurrentUserSession . . . ??
wxImagePanel::handler . . . ??
DimensionlessFrames::MainFrame::MainFrameThreadHandler . . . ??
DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler . . . ??
DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler . . . ??
DimensionlessFrames::StartupFrame::StartupFrameThreadHandler . . . ??
DimensionlessFunctionality::ApplicationLogic::User . . . ??
- DimensionlessFunctionality::ApplicationLogic::ConnectedUser . . . ??
- DimensionlessFunctionality::ApplicationLogic::CurrentUser . . . ??

wxApp
- MainApp . . . ??

wxFrame
- ChatFrameBase . . . ??
  - DimensionlessFrames::ChatFrame . . . ??
- MainFrameBase . . . ??
  - DimensionlessFrames::MainFrame . . . ??
- SearchForUsersFrameBase . . . ??
  - DimensionlessFrames::SearchForUsersFrame . . . ??
- SettingsFrameBase . . . ??
  - DimensionlessFrames::SettingsFrame . . . ??
- StartupFrameBase . . . ??
  - DimensionlessFrames::StartupFrame . . . ??

wxObject
- DimensionlessFrames::ChatUser . . . ??

wxPanel
- wxImagePanel . . . ??

wxTreeItemData
- DimensionlessFrames::ConnectedUserIndex . . . ??

Chapter 9

Class Index

9.1 Class List

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

DimensionlessFrames::ChatFrame
: This class creates a ChatFrame to chat with other ConnectedUsers . . . . . ??

ChatFrameBase . . . . . ??

DimensionlessFrames::ChatFrame::ChatFrameThreadHandler
: This struct handles events posted from our chatFrameThread . . . . . ??

DimensionlessFrames::ChatUser
: This class stores a reference to a ConnectedUser that the current user can chat with in the ChatFrame . . . . . ??

DimensionlessFunctionality::ApplicationLogic::ConnectedUser
: This class denotes a user that is connected to the static CurrentUser that is logged into the application . . . . . ??

DimensionlessFrames::ConnectedUserIndex
: This class stores a reference to a ConnectedUser that the list of ConnectedUsers in the MainFrame . . . . . ??

DimensionlessFunctionality::ApplicationLogic::CurrentUser
: This class denotes the User that is currently logged into the application . . . . . ??

DimensionlessFunctionality::ApplicationLogic::CurrentUserSession
: This singleton class stores a smart shared pointer reference to the currently logged in user and a collection of their connected users . . . . . ??

wxImagePanel::handler
: This functor allows for events on the wxImagePanel to be handled . . . . . ??

MainApp . . . . . ??

DimensionlessFrames::MainFrame
: This class creates a MainFrame that is displayed after a user logs in . . . . . ??

MainFrameBase . . . . . ??

DimensionlessFrames::MainFrame::MainFrameThreadHandler
: This struct handles events posted from our mainFrameThread . . . . . ??

DimensionlessFrames::SearchForUsersFrame
: This class creates a SearchForUsersFrame that is displayed when the currently logged in user wishes to search for other users . . . . . ??

SearchForUsersFrameBase . . . . . ??

DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler
: This struct handles events posted from our searchForUsersFrameThread . . . . . ??

DimensionlessFrames::SettingsFrame
: This class creates a SettingsFrame that is displayed when the currently logged in user wishes to view or modify their settings . . . . . ??

SettingsFrameBase . . . . . ??
DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler
This struct handles events posted from our settingsFrameThread . . . . . ??
DimensionlessFrames::StartupFrame
This class creates a StartupFrame that is the first frame displayed when the application is started ??
StartupFrameBase . . . . . ??
DimensionlessFrames::StartupFrame::StartupFrameThreadHandler
This struct handles events posted from our startupFrameThread . . . . . ??
DimensionlessFunctionality::ApplicationLogic::User
This class acts as a base class for ConnectedUser and CurrentUser, consisting of attributes and methods that should be common to both classes . . . . . ??
wxImagePanel
Class wxImagePanel . . . . . ??

Chapter 10

File Index

10.1 File List

Here is a list of all files with brief descriptions:

DimensionlessClient/ChatFrame.cpp . . . ??
DimensionlessClient/ChatFrame.h . . . ??
DimensionlessClient/Dimensionless.cpp . . . ??
DimensionlessClient/Dimensionless.h . . . ??
DimensionlessClient/Dimensionless_dimensionlessclient_bitmaps.cpp . . . ??
DimensionlessClient/DimensionlessControls.cpp . . . ??
DimensionlessClient/DimensionlessControls.h . . . ??
DimensionlessClient/DimensionlessFunctionality.cpp . . . ??
DimensionlessClient/DimensionlessFunctionality.h . . . ??
DimensionlessClient/main.cpp . . . ??
DimensionlessClient/MainFrame.cpp . . . ??
DimensionlessClient/MainFrame.h . . . ??
DimensionlessClient/SearchForUsersFrame.cpp . . . ??
DimensionlessClient/SearchForUsersFrame.h . . . ??
DimensionlessClient/SettingsFrame.cpp . . . ??
DimensionlessClient/SettingsFrame.h . . . ??
DimensionlessClient/StartupFrame.cpp . . . ??
DimensionlessClient/StartupFrame.h . . . ??
DimensionlessClient/wxcrafter_bitmaps.cpp . . . ??

Chapter 11

Namespace Documentation

11.1 DimensionlessFrames Namespace Reference

Classes

- class ChatFrame

  This class creates a ChatFrame to chat with other ConnectedUsers.
- class ChatUser

  This class stores a reference to a ConnectedUser that the current user can chat with in the ChatFrame.
- class ConnectedUserIndex

  This class stores a reference to a ConnectedUser that the list of ConnectedUsers in the MainFrame.
- class MainFrame

  This class creates a MainFrame that is displayed after a user logs in.
- class SearchForUsersFrame

  This class creates a SearchForUsersFrame that is displayed when the currently logged in user wishes to search for other users.
- class SettingsFrame

  This class creates a SettingsFrame that is displayed when the currently logged in user wishes to view or modify their settings.
- class StartupFrame

  This class creates a StartupFrame that is the first frame displayed when the application is started.

11.1.1 Detailed Description

This namespace contains all of the classes for the frames the application uses.

11.2 DimensionlessFunctionality Namespace Reference

Namespaces

- ApplicationLogic
- Compression
- Cryptography
- Networking
- UtilityFunctions
- Validation
- XML

11.2.1 Detailed Description

The DimensionlessFunctionality namespace stores all of the functionality for the application.

11.3 DimensionlessFunctionality::ApplicationLogic Namespace Reference

Classes

- class ConnectedUser

  This class denotes a user that is connected to the static CurrentUser that is logged into the application.
- class CurrentUser

  This class denotes the User that is currently logged into the application.
- class CurrentUserSession

  This singleton class stores a smart shared pointer reference to the currently logged in user and a collection of their connected users.
- class User

  This class acts as a base class for ConnectedUser and CurrentUser, consisting of attributes and methods that should be common to both classes.

Enumerations

- enum Request {
  Request::SendConnectionRequest, Request::AcceptConnectionRequest, Request::RejectConnectionRequest, Request::SendMessage,
  Request::PerformSearch, Request::GetOurRemoteIP, Request::SetOurXML, Request::SetOurTorrent,
  Request::GetConnectedUserXML, Request::GetConnectedUserTorrent, Request::Null }

  This enumeration decides the type of request that our CurrentUser wants to send to their connectedUsers.

Functions

- void RefreshConnectedUsers ()
- void SendConnectedUserRequest (const Request &request, const std::shared_ptr< ConnectedUser > &targetUser, const string &extradata)
- void SendMultithreadedConnectedUserRequest (std::atomic< unsigned int > &activeRequestHandlingUsers, const Request &request, const std::shared_ptr< ConnectedUser > &targetUser, const string &extradata)
- std::vector< std::shared_ptr< ConnectedUser > > ProcessFoundUsers (const string &foundUsers)
- bool LoginInitialization (const string &username, const string &password)
- bool VerifyLoginCredentials (const string &username, const string &password)
- bool NewUserRegistration (const string &username, const string &password)
- bool LogOutCleanup (const string &username)

11.3.1 Detailed Description

The ApplicationLogic namespace stores all of the functions and classes that manage our connected users and combine functions from other namespaces to perform actions necessary for the application to function.

11.3.2 Enumeration Type Documentation

11.3.2.1 Request

```
enum DimensionlessFunctionality::ApplicationLogic::Request  [strong]
```

This enumeration decides the type of request that our CurrentUser wants to send to their connectedUsers.

Enumerator

SendConnectionRequest	Denotes that a connection request is to be sent from us to another user.
AcceptConnectionRequest	Denotes that a connection request is accepted by us.
RejectConnectionRequest	Denotes that a connection request is rejected by us.
SendMessage	Denotes that a message is to be sent to another user from us.
PerformSearch	Denotes that an IP address search is to be performed by another user for us.
GetOurRemoteIP	Denotes that we request our global IPv4 address from another user.
SetOurXML	Denotes that another user is to store the XML we provide them with in this request.
SetOurTorrent	Denotes that another user is to store and use the torrent we provide them with in this request.
GetConnectedUserXML	Denotes that we request the XML of another user for us to store.
GetConnectedUserTorrent	Denotes that we request the torrent of another user for us to store and use.
Null	Denotes that a lack of a request type has been selected.

Definition at line 865 of file DimensionlessFunctionality.h.

```
865                             {
866                                SendConnectionRequest,
867                                AcceptConnectionRequest,
868                                RejectConnectionRequest,
869                                SendMessage,
870                                PerformSearch,
871                                GetOurRemoteIP,
872                                SetOurXML,
873                                SetOurTorrent,
874                                GetConnectedUserXML,
875                                GetConnectedUserTorrent,
876                                Null
877                              };
```

11.3.3 Function Documentation

11.3.3.1 LoginInitialization()

```
bool DimensionlessFunctionality::ApplicationLogic::LoginInitialization (
            const string & username,
            const string & password )
```

This function proceeds with initializing the necessary threads for the application to function after a given username and password have been verified.

Parameters

in	*username*	The username of the user logging in.
in	*password*	The password of the user logging in.

Returns

A boolean indicating if our login was successful or not.

Definition at line 2407 of file DimensionlessFunctionality.cpp.

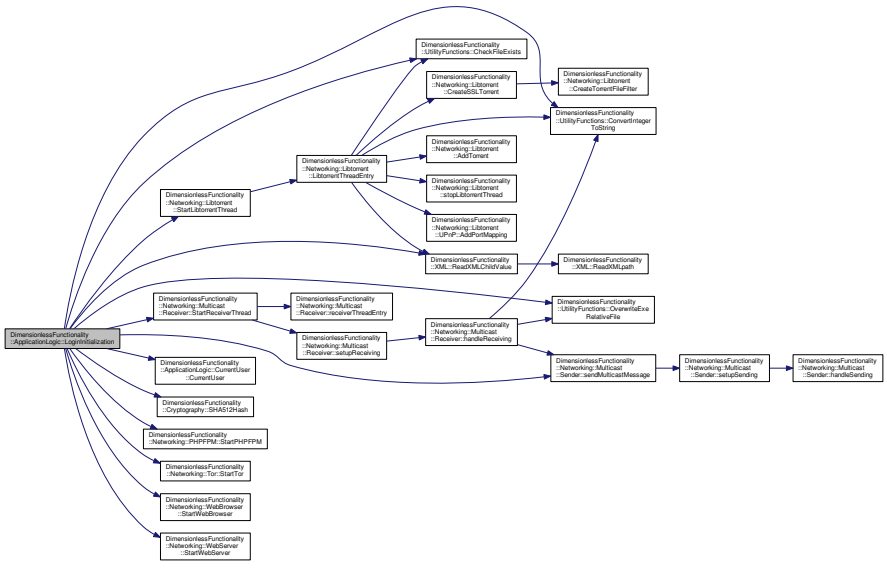
References DimensionlessFunctionality::UtilityFunctions::CheckFileExists(), DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::ApplicationLogic::CurrentUser::CurrentUser(), DimensionlessFunctionality::ApplicationLogic::CurrentUser::hashedPassword, DimensionlessFunctionality::Networking::Multicast::localUserMap, DimensionlessFunctionality::ApplicationLogic::User::online, DimensionlessFunctionality::UtilityFunctions::OverwriteExeRelativeFile(), DimensionlessFunctionality::XML::ReadXMLChildValue(), DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage(), DimensionlessFunctionality::Cryptography::SHA512Hash(), DimensionlessFunctionality::Networking::Libtorrent::StartLibtorrentThread(), DimensionlessFunctionality::Networking::PHPFPM::StartPHPFPM(), DimensionlessFunctionality::Networking::Multicast::Receiver::StartReceiverThread(), DimensionlessFunctionality::Networking::Tor::StartTor(), DimensionlessFunctionality::Networking::WebBrowser::StartWebBrowser(), DimensionlessFunctionality::Networking::WebServer::StartWebServer(), and DimensionlessFunctionality::ApplicationLogic::User::webPort.

Referenced by DimensionlessFrames::StartupFrame::OnBtnLoginClick().

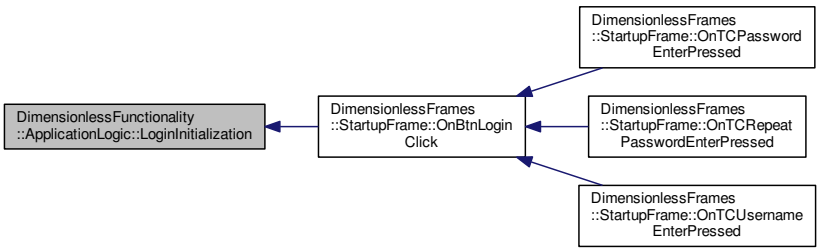
```
2408         {
2409             CurrentUserSession::currentUser = std::shared_ptr<CurrentUser>(new CurrentUser(
2410             XML::ReadXMLChildValue("./Users/"+username+"/data/"+username+".xml","User
     ","AESIV"),
2411             XML::ReadXMLChildValue("./Users/"+username+"/data/"+username+".xml","User
     ","Username"),
2412             XML::ReadXMLChildValue("./Users/"+username+"/data/"+username+".xml","User
     ","IPAddress"),
2413             XML::ReadXMLChildValue("./Users/"+username+"/data/"+username+".xml","User
     ","TorHiddenServiceAddress"),
2414             XML::ReadXMLChildValue("./Users/"+username+"/data/"+username+".xml","User
     ","FreeDNSAfraidOrgDomain")
2415             ));
2416             CurrentUserSession::currentUser->hashedPassword =
     Cryptography::SHA512Hash(password);
2417 
2418             try{
2419                 Networking::Multicast::Receiver::StartReceiverThread
     ("0.0.0.0", "239.255.0.1");
2420 
2421                 for(int i = 0; i < 3; i++){
2422                     Networking::Multicast::Sender::sendMulticastMessage
     ("239.255.0.1", "Query");
2423                 }
2424             } catch (std::exception const& ex){
2425                 std::cout << ex.what() << std::endl;
2426             }
2427             if(Networking::Multicast::localUserMap.size() > 1){
2428                 // we have company on our LAN.
2429                 // raise the port values by the number of users - 1! (Since we should be on this list too).
2430                 CurrentUserSession::currentUser->libtorrentPort +=
     Networking::Multicast::localUserMap.size() - 1;
2431                 CurrentUserSession::currentUser->webPort +=
     Networking::Multicast::localUserMap.size() - 1;
2432                 CurrentUserSession::currentUser->OpenVPNPort +=
     Networking::Multicast::localUserMap.size() - 1;
2433             }
2434             CurrentUserSession::currentUser->currentStatus = User::Status::online;
2435 
2436             if(UtilityFunctions::CheckFileExists(
     ApplicationInfo::GetExecutablePath()+"Users/"+username+"/nginx.conf")){
2437                 std::ifstream file(ApplicationInfo::GetExecutablePath()+"Users/"+username+"/nginx.conf");
2438                 std::stringstream buffer;
2439                 buffer << file.rdbuf();
2440                 std::string str = buffer.str();
2441                 file.close();
```

```
2442                boost::replace_all(str, UtilityFunctions::ConvertIntegerToString
    (1337), UtilityFunctions::ConvertIntegerToString(
    CurrentUserSession::currentUser->webPort));
2443                UtilityFunctions::OverwriteExeRelativeFile("
    Users/"+username+"/nginx.conf", str);
2444            }
2445
2446            if(UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username+"/torrc")){
2447                std::ifstream file(ApplicationInfo::GetExecutablePath()+"Users/"+username+"/torrc");
2448                std::stringstream buffer;
2449                buffer << file.rdbuf();
2450                std::string str = buffer.str();
2451                file.close();
2452                boost::replace_all(str, UtilityFunctions::ConvertIntegerToString
    (1337), UtilityFunctions::ConvertIntegerToString(
    CurrentUserSession::currentUser->webPort));
2453                UtilityFunctions::OverwriteExeRelativeFile("
    Users/"+username+"/torrc", str);
2454            }
2455
2456            Networking::Libtorrent::StartLibtorrentThread(
    username, CurrentUserSession::currentUser->hashedPassword);
2457            //openvpn?
2458            Networking::Tor::StartTor("Users/"+username);
2459            Networking::PHPFPM::StartPHPFPM("Users/"+username);
2460            Networking::WebServer::StartWebServer("Users/"+username,
    CurrentUserSession::currentUser->hashedPassword);
2461            Networking::WebBrowser::StartWebBrowser();
2462
2463            return true;
2464        }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.3.3.2 LogOutCleanup()

```
bool DimensionlessFunctionality::ApplicationLogic::LogOutCleanup (
            const string & username )
```

This function logs out a user and cleans up any threads started by the LoginInitialization function.

Parameters

in	*username*	The username of the user logging out.

Returns

A boolean indicating if our user has logged out successfully.

Definition at line 2557 of file DimensionlessFunctionality.cpp.

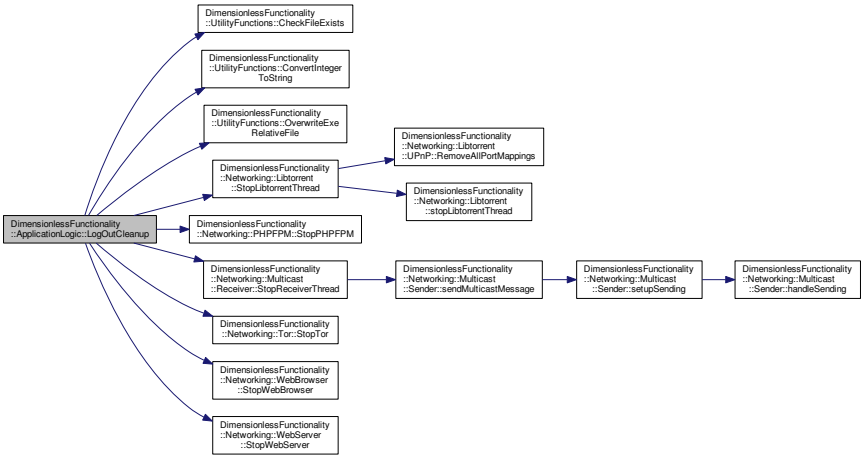
References DimensionlessFunctionality::UtilityFunctions::CheckFileExists(), DimensionlessFunctionality::Utility↩
Functions::ConvertIntegerToString(), DimensionlessFunctionality::ApplicationLogic::User::offline, Dimensionless↩
Functionality::UtilityFunctions::OverwriteExeRelativeFile(), DimensionlessFunctionality::Networking::Libtorrent↩
::StopLibtorrentThread(), DimensionlessFunctionality::Networking::PHPFPM::StopPHPFPM(), Dimensionless↩
Functionality::Networking::Multicast::Receiver::StopReceiverThread(), DimensionlessFunctionality::Networking↩
::Tor::StopTor(), DimensionlessFunctionality::Networking::WebBrowser::StopWebBrowser(), Dimensionless↩
Functionality::Networking::WebServer::StopWebServer(), and DimensionlessFunctionality::ApplicationLogic::↩
User::webPort.

Referenced by DimensionlessFrames::MainFrame::OnClose().

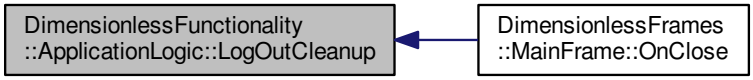
```
2558         {
2559             Networking::WebBrowser::StopWebBrowser();
2560             Networking::WebServer::StopWebServer("Users/"+username);
2561             Networking::PHPFPM::StopPHPFPM("Users/"+username);
2562             Networking::Tor::StopTor("Users/"+username);
2563             //openvpn?
2564             Networking::Libtorrent::StopLibtorrentThread();
2565             try{
2566                 Networking::Multicast::Receiver::StopReceiverThread
     ();
2567             } catch(std::exception& ex){
```

```
2568                std::cout << ex.what() << std::endl;
2569            }
2570
2571            CurrentUserSession::currentUser->currentStatus = User::Status::offline;
2572
2573            if(UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username+"/nginx.conf")){
2574                std::ifstream file(ApplicationInfo::GetExecutablePath()+"Users/"+username+"/nginx.conf");
2575                std::stringstream buffer;
2576                buffer << file.rdbuf();
2577                std::string str = buffer.str();
2578                file.close();
2579                boost::replace_all(str, UtilityFunctions::ConvertIntegerToString
    (CurrentUserSession::currentUser->webPort),
    UtilityFunctions::ConvertIntegerToString(1337));
2580                UtilityFunctions::OverwriteExeRelativeFile("
    Users/"+username+"/nginx.conf", str);
2581            }
2582
2583            if(UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username+"/torrc")){
2584                std::ifstream file(ApplicationInfo::GetExecutablePath()+"Users/"+username+"/torrc");
2585                std::stringstream buffer;
2586                buffer << file.rdbuf();
2587                std::string str = buffer.str();
2588                file.close();
2589                boost::replace_all(str, UtilityFunctions::ConvertIntegerToString
    (CurrentUserSession::currentUser->webPort),
    UtilityFunctions::ConvertIntegerToString(1337));
2590                UtilityFunctions::OverwriteExeRelativeFile("
    Users/"+username+"/torrc", str);
2591            }
2592
2593            CurrentUserSession::connectedUsers.clear();
2594            CurrentUserSession::currentUser.reset();
2595            return true;
2596        }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.3.3.3 NewUserRegistration()

```
bool DimensionlessFunctionality::ApplicationLogic::NewUserRegistration (
            const string & username,
            const string & password )
```

This function registers a new user for our application.

Parameters

in	*username*	The username of the new user to register.
in	*password*	The password for the new user to register.

Returns

A boolean indicating if our new user was registered successfully or not.

Definition at line 2483 of file DimensionlessFunctionality.cpp.

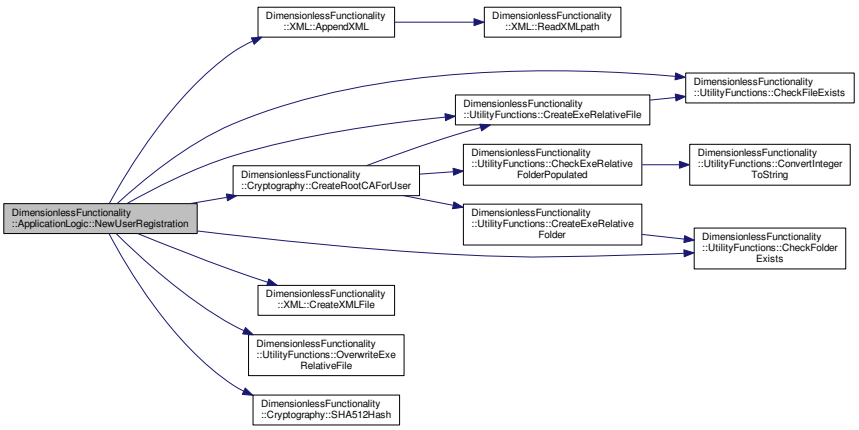
References DimensionlessFunctionality::XML::AppendXML(), DimensionlessFunctionality::UtilityFunctions::CheckFileExists(), DimensionlessFunctionality::UtilityFunctions::CheckFolderExists(), DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile(), DimensionlessFunctionality::Cryptography::CreateRootCAForUser(), DimensionlessFunctionality::XML::CreateXMLFile(), DimensionlessFunctionality::UtilityFunctions::OverwriteExeRelativeFile(), DimensionlessFunctionality::Cryptography::SHA512Hash(), and DimensionlessFunctionality::ApplicationLogic::User::username.

Referenced by DimensionlessFrames::StartupFrame::ThreadEntryCheckLogin().

```
2484        {
2485            if(!UtilityFunctions::CheckFolderExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username)){
2486                string cpcmd = "cp -R ./User-Template ./Users/"+username;
2487                std::system(cpcmd.c_str());
2488                if(UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username+"/data/bin/grav")){
2489                    UtilityFunctions::CreateExeRelativeFile("Users/"
    +username+"/data/user/accounts/"+username+".yaml",
2490                    "email: "+username+"@dimensionless\n"
2491                    +"fullname: "+username+"\n"
2492                    +"title: "+username+"\n"
2493                    +"state: enabled\n"
2494                    +"access:\n  admin:\n    login: true\n    super: true\n  site:\n    login: true\n");
2495                    string gravpasscmd = "cd ./Executables; ./php -r \"echo \\\"hashed_password: \\\"
    .password_hash(\\\""+password+"\\\",PASSWORD_DEFAULT);\" >> ../Users/"+username+"/data/user/accounts/"+username+"
    .yaml";
2496                    std::system(gravpasscmd.c_str());
2497                }
2498
2499                if(UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username+"/nginx.conf")){
2500                    std::ifstream file(ApplicationInfo::GetExecutablePath()+"Users/"+username+"/nginx.conf"
    );
2501                    std::stringstream buffer;
2502                    buffer << file.rdbuf();
2503                    std::string str = buffer.str();
2504                    file.close();
2505                    boost::replace_all(str, "<place-username-here>", username);
2506                    UtilityFunctions::OverwriteExeRelativeFile("
    Users/"+username+"/nginx.conf", str);
2507                }
2508
2509                if(UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+username+"/torrc")){
```

```
2510                    std::ifstream file(ApplicationInfo::GetExecutablePath()+"Users/"+username+"/torrc");
2511                    std::stringstream buffer;
2512                    buffer << file.rdbuf();
2513                    std::string str = buffer.str();
2514                    file.close();
2515                    boost::replace_all(str, "<insert-full-path-here>", ApplicationInfo::GetExecutablePath()
    +"Users/"+username+"/hiddenservice");
2516                    UtilityFunctions::OverwriteExeRelativeFile("
    Users/"+username+"/torrc", str);
2517                }
2518
2519                boost::random_device rd;
2520                // Removed a bunch of potentially harmful characters.
2521                std::string chars("
    abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^*()`~-_=+[{]}|;:.?");
2522                string AESIV = "";
2523                boost::variate_generator< boost::random_device&, boost::uniform_int<> >
2524                gen(rd, boost::uniform_int<>(0, chars.size()-1));
2525                for(int i = 0; i < 16; i++) {
2526                    AESIV += chars[gen()];
2527                }
2528
2529                XML::CreateXMLFile("./Users/"+username+"/data/"+username+".xml", "User");
2530                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    Username", username, {}, {});
2531                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    IPAddress", "", {}, {});
2532                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    FreeDNSAfraidOrgDomain", "", {}, {});
2533                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    TorHiddenServiceAddress", "", {}, {});
2534                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "AESIV",
     AESIV, {}, {});
2535                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    LastPageUpdatedTime", "", {}, {});
2536                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    Platform", "Desktop", {}, {});
2537                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    DefaultWebBrowser", "GNU-IceCat", {}, {});
2538                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "
    PublicSearchVisibility", "Visible", {}, {});
2539                XML::AppendXML("./Users/"+username+"/data/"+username+".xml", "User", "Tor", "
    Enabled-Hidden-Service", {}, {});
2540                Cryptography::CreateRootCAForUser(username,
    Cryptography::SHA512Hash(password));
2541                return true;
2542            } else {
2543                if(username.length() > 50){
2544                    throw std::runtime_error("Error! User: " + username.substr(0,50) + "... already exists.
     Please choose another username.");
2545                } else {
2546                    throw std::runtime_error("Error! User: " + username + " already exists. Please choose
     another username.");
2547                }
2548                return false;
2549            }
2550        }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.3.3.4 ProcessFoundUsers()

```
std::vector< std::shared_ptr< ConnectedUser > > DimensionlessFunctionality::Application↩
Logic::ProcessFoundUsers (
            const string & foundUsers )
```

This function processes a comma separated string into a vector of ConnectedUser smart pointer references.

Parameters

in	*foundUsers*	A string of users and their data residing at a particular local address (obtained via multicast message).

Returns

A vector of ConnectedUser smart pointer references.

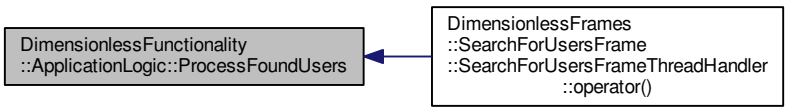
Definition at line 2357 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()().

```
2357                                                                                    {
2358             std::vector< std::shared_ptr<ConnectedUser> > returnConnectedUsers;
2359             std::vector<string> foundUsersTemp;
2360             boost::split(foundUsersTemp, foundUsers, boost::is_any_of(","));
2361             int totalUsers = atoi(foundUsersTemp[0].c_str());
2362
2363             vector<string>::const_iterator first = foundUsersTemp.begin() + totalUsers + 1;
2364             vector<string>::const_iterator last = foundUsersTemp.end();
2365             vector<string> foundUsersSubVectorTemp(first, last);
2366             string singleUserTemp = boost::algorithm::join(foundUsersSubVectorTemp, ",");
2367             int runningTotalIndexStart = 0;
2368             for(int i = 0; i < totalUsers; i++){
2369                 string singleUser = singleUserTemp.substr(runningTotalIndexStart,atoi(foundUsersTemp[(i+1)]
     .c_str()));
2370                 runningTotalIndexStart+=atoi(foundUsersTemp[(i+1)].c_str());
2371
2372                 std::vector<string> singleUserTempSplit;
2373                 boost::split(singleUserTempSplit, singleUser, boost::is_any_of(","));
2374
2375                 int sumOfLengths = 0;
2376
2377                 for (int i = 0; i < 7; i++) {
2378                     sumOfLengths += singleUserTempSplit[i].length();
2379                 }
2380
2381                 int additionalSum = 0;
2382
2383                 std::vector<string> finalSingleUserItems;
2384                 for (int i = 0; i < 7; i++) {
2385                     string item = singleUser.substr(sumOfLengths + additionalSum + 7 + i, atoi(
```

```
    singleUserTempSplit[i].c_str()));
2386                    finalSingleUserItems.push_back(item);
2387                    additionalSum += atoi(singleUserTempSplit[i].c_str());
2388                }
2389
2390                //aesiv can be anything, will be overwritten in xml on request sent.
2391                shared_ptr<ConnectedUser> tempUser = std::shared_ptr<ConnectedUser>(new ConnectedUser("
    aesiv",finalSingleUserItems[0],finalSingleUserItems[1],finalSingleUserItems[2],finalSingleUserItems[3]));
2392                tempUser->libtorrentPort = atoi(finalSingleUserItems[4].c_str());
2393                tempUser->webPort = atoi(finalSingleUserItems[5].c_str());
2394                tempUser->OpenVPNPort = atoi(finalSingleUserItems[6].c_str());
2395                returnConnectedUsers.push_back(tempUser);
2396            }
2397
2398            return returnConnectedUsers;
2399        }
```

Here is the caller graph for this function:



11.3.3.5 RefreshConnectedUsers()

```
void DimensionlessFunctionality::ApplicationLogic::RefreshConnectedUsers ( )
```

This function reloads all the ConnectedUsers from our CurrentUser's XML.

Definition at line 1952 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::XML::GetXMLElementsAndAttributes(), DimensionlessFunctionality::XML::ReadXMLChildValue(), and DimensionlessFunctionality::ApplicationLogic::User::username.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
1953        {
1954            std::vector<string> xmlConnectedUsers =
    XML::GetXMLElementsAndAttributes("./Users/"+CurrentUserSession::currentUser
    ->username+"/data/"+CurrentUserSession::currentUser->username+".xml","User","ConnectedUser",{"aesiv","
    ipaddress","torhiddenserviceaddress","freednsafraidorgdomain"});
1955            CurrentUserSession::connectedUsers.clear();
1956            for(unsigned int i = 0; i < xmlConnectedUsers.size(); i++){
1957                CurrentUserSession::connectedUsers.emplace_back(std::shared_ptr<ConnectedUser>(new
    ConnectedUser(XML::ReadXMLChildValue("./Users/"+CurrentUserSession::currentUser->username+"
    /data/"+CurrentUserSession::currentUser->username+".xml","User",xmlConnectedUsers[i+1]),
1958
          XML::ReadXMLChildValue("./Users/"+CurrentUserSession::currentUser->username+"
    /data/"+CurrentUserSession::currentUser->username+".xml","User",xmlConnectedUsers[i]),
1959
          XML::ReadXMLChildValue("./Users/"+CurrentUserSession::currentUser->username+"
    /data/"+CurrentUserSession::currentUser->username+".xml","User",xmlConnectedUsers[i+2]),
1960
          XML::ReadXMLChildValue("./Users/"+CurrentUserSession::currentUser->username+"
    /data/"+CurrentUserSession::currentUser->username+".xml","User",xmlConnectedUsers[i+3]),
1961
          XML::ReadXMLChildValue("./Users/"+CurrentUserSession::currentUser->username+"
    /data/"+CurrentUserSession::currentUser->username+".xml","User",xmlConnectedUsers[i+4])));
1962            }
1963        }
```

Here is the call graph for this function:

Here is the caller graph for this function:

11.3.3.6 SendConnectedUserRequest()

```
void DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest (
            const Request & request,
            const std::shared_ptr< ConnectedUser > & targetUser,
            const string & extradata )
```

This function sends a request synchronously to one of our ConnectedUsers through PHP.

Parameters

in	*request*	The type of request to be sent.
in	*targetUser*	A reference to the ConnectedUser the request is to be sent to.
in	*extradata*	Additional data that may be required to be sent along with our request.

Definition at line 1971 of file DimensionlessFunctionality.cpp.

References AcceptConnectionRequest, DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::Cryptography::CreatePeerCertificateForUser(), DimensionlessFunctionality::↩ Networking::Multicast::Receiver::data, GetConnectedUserTorrent, GetConnectedUserXML, GetOurRemoteIP, Null, PerformSearch, RejectConnectionRequest, SendConnectionRequest, SendMessage, SetOurTorrent, SetOurXML, and DimensionlessFunctionality::ApplicationLogic::User::username.

Referenced by DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick(), DimensionlessFrames↩ ::MainFrame::OnTcConnectedUsersContextMenuClick(), DimensionlessFrames::SearchForUsersFrame::Thread↩ EntrySendConnectionRequest(), and DimensionlessFrames::ChatFrame::ThreadEntrySendMessage().

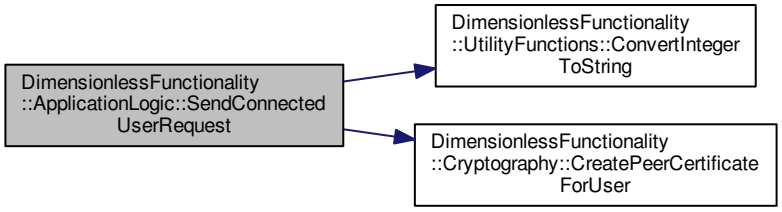
```
1972         {
1973             string serverurl;
1974             if(targetUser->regularDomain != ""){
1975                 serverurl = "http%3A%2F%2F"+targetUser->regularDomain+"%3A"+
```

```
    UtilityFunctions::ConvertIntegerToString(targetUser->webPort);
1976            } else {
1977                serverurl = "http%3A%2F%2F"+targetUser->globalIPv4Address+"%3A"+
    UtilityFunctions::ConvertIntegerToString(targetUser->webPort);
1978            }
1979            string basicauth = targetUser->globalIPv4Address+":"+targetUser->AESiv;
1980
1981            boost::random_device rd;
1982            // Removed a bunch of potentially harmful characters.
1983            std::string chars("
    abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^*()`~-_=+[{]}|;:.?");
1984            string aeskey = "";
1985            boost::variate_generator< boost::random_device&, boost::uniform_int<> >
1986            gen(rd, boost::uniform_int<>(0, chars.size()-1));
1987            for(int i = 0; i < 32; i++) {
1988                aeskey += chars[gen()];
1989            }
1990
1991            string source = CurrentUserSession::currentUser->toSourceTargetString();
1992            string target = targetUser->toSourceTargetString();
1993
1994            string requesttype = "";
1995            string data = "";
1996            switch(request){
1997                case Request::SendConnectionRequest:
1998                {
1999                    requesttype = "send-connection-request";
2000                    Cryptography::CreatePeerCertificateForUser(
    CurrentUserSession::currentUser->username,targetUser->username,targetUser->AESiv,
    CurrentUserSession::currentUser->hashedPassword);
2001                    data = CurrentUserSession::currentUser->toDataString();
2002                    break;
2003                }
2004                case Request::AcceptConnectionRequest:
2005                {
2006                    requesttype = "accept-connection-request";
2007                    Cryptography::CreatePeerCertificateForUser(
    CurrentUserSession::currentUser->username,targetUser->username,targetUser->AESiv,
    CurrentUserSession::currentUser->hashedPassword);
2008
2009                    std::ifstream torrentfile("./Users/"+CurrentUserSession::currentUser->username+"/"+
    CurrentUserSession::currentUser->username+".torrent");
2010                    std::stringstream torrentbuffer;
2011                    torrentbuffer << torrentfile.rdbuf();
2012                    std::string torrent = torrentbuffer.str();
2013                    torrentfile.close();
2014
2015                    std::ifstream privatekeyfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-privatekey.pem");
2016                    std::stringstream privatekeybuffer;
2017                    privatekeybuffer << privatekeyfile.rdbuf();
2018                    std::string peerprivatekey = privatekeybuffer.str();
2019                    privatekeyfile.close();
2020
2021                    std::ifstream certfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-cert.pem");
2022                    std::stringstream certbuffer;
2023                    certbuffer << certfile.rdbuf();
2024                    std::string peercert = certbuffer.str();
2025                    certfile.close();
2026
2027                    std::ifstream dhparamsfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/dhparams.pem");
2028                    std::stringstream dhparamsbuffer;
2029                    dhparamsbuffer << dhparamsfile.rdbuf();
2030                    std::string dhparams = dhparamsbuffer.str();
2031                    dhparamsfile.close();
2032
2033                    data = UtilityFunctions::ConvertIntegerToString
    (torrent.length())+","+
2034                            UtilityFunctions::ConvertIntegerToString
    (peerprivatekey.length())+","+
2035                            UtilityFunctions::ConvertIntegerToString
    (peercert.length())+","+
2036                            UtilityFunctions::ConvertIntegerToString
    (dhparams.length())+","+
2037                            torrent+","+
2038                            peerprivatekey+","+
2039                            peercert+","+
2040                            dhparams;
2041                    break;
2042                }
2043                case Request::RejectConnectionRequest:
2044                {
2045                    requesttype = "reject-connection-request";
2046                    break;
2047                }
```

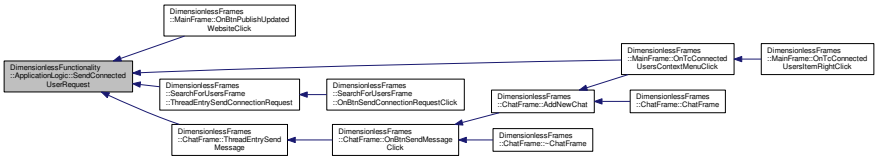
```
2048                case Request::SendMessage:
2049                {
2050                    requesttype = "send-message";
2051                    data = extradata;
2052                    break;
2053                }
2054                case Request::PerformSearch:
2055                {
2056                    requesttype = "perform-search";
2057                    data = extradata;
2058                    break;
2059                }
2060                case Request::GetOurRemoteIP:
2061                {
2062                    requesttype = "get-remote-ip";
2063                    break;
2064                }
2065                case Request::SetOurXML:
2066                {
2067                    requesttype = "set-client-xml";
2068                    data = CurrentUserSession::currentUser->toDataString(extradata);
2069                    break;
2070                }
2071                case Request::SetOurTorrent:
2072                {
2073                    requesttype = "set-client-torrent";
2074                    Cryptography::CreatePeerCertificateForUser(
    CurrentUserSession::currentUser->username,targetUser->username,targetUser->AESiv,
    CurrentUserSession::currentUser->hashedPassword);
2075
2076                    std::ifstream torrentfile("./Users/"+CurrentUserSession::currentUser->username+"/"+
    CurrentUserSession::currentUser->username+".torrent");
2077                    std::stringstream torrentbuffer;
2078                    torrentbuffer << torrentfile.rdbuf();
2079                    std::string torrent = torrentbuffer.str();
2080                    torrentfile.close();
2081
2082                    std::ifstream privatekeyfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-privatekey.pem");
2083                    std::stringstream privatekeybuffer;
2084                    privatekeybuffer << privatekeyfile.rdbuf();
2085                    std::string peerprivatekey = privatekeybuffer.str();
2086                    privatekeyfile.close();
2087
2088                    std::ifstream certfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-cert.pem");
2089                    std::stringstream certbuffer;
2090                    certbuffer << certfile.rdbuf();
2091                    std::string peercert = certbuffer.str();
2092                    certfile.close();
2093
2094                    std::ifstream dhparamsfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/dhparams.pem");
2095                    std::stringstream dhparamsbuffer;
2096                    dhparamsbuffer << dhparamsfile.rdbuf();
2097                    std::string dhparams = dhparamsbuffer.str();
2098                    dhparamsfile.close();
2099
2100                    data = UtilityFunctions::ConvertIntegerToString
    (torrent.length())+","+
2101                           UtilityFunctions::ConvertIntegerToString
    (peerprivatekey.length())+","+
2102                           UtilityFunctions::ConvertIntegerToString
    (peercert.length())+","+
2103                           UtilityFunctions::ConvertIntegerToString
    (dhparams.length())+","+
2104                           torrent+","+
2105                           peerprivatekey+","+
2106                           peercert+","+
2107                           dhparams;
2108                    break;
2109                }
2110                case Request::GetConnectedUserXML:
2111                {
2112                    requesttype = "get-server-xml";
2113                    break;
2114                }
2115                case Request::GetConnectedUserTorrent:
2116                {
2117                    requesttype = "get-server-torrent";
2118                    break;
2119                }
2120                case Request::Null:
2121                default:
2122                {
2123                    throw std::runtime_error("Error! Invalid request.");
2124                }
```

```
2125            }
2126
2127            string cmd = "./Executables/php-cgi -f ../Users/"+CurrentUserSession::currentUser->username+"
    /data/post-request-sender.php server-url="+serverurl+" basic-auth="+basicauth+" source="+source+" target="+
    target+" request-type="+requesttype+" data="+data+" aes-key="+aeskey+" > user-request-result.txt";
2128            std::system(cmd.c_str());
2129            std::ifstream file("./user-request-result.txt");
2130            std::stringstream buffer;
2131            buffer << file.rdbuf();
2132            std::string str = buffer.str();
2133            file.close();
2134            boost::filesystem::remove("./user-request-result.txt");
2135            if(!boost::algorithm::contains(str, "Success")){
2136                string serverurl;
2137                if(targetUser->onionDomain != ""){
2138                    serverurl = "http%3A%2F%2F"+targetUser->onionDomain+"%3A"+
    UtilityFunctions::ConvertIntegerToString(targetUser->webPort);
2139                } else {
2140                    throw std::runtime_error("Error! Something went wrong sending a request to "+targetUser
    ->username);
2141                }
2142                string cmd = "./Executables/php-cgi -f ../Users/"+CurrentUserSession::currentUser->username
    +"/data/post-request-sender.php server-url="+serverurl+" basic-auth="+basicauth+" source="+source+" target="
    +target+" request-type="+requesttype+" data="+data+" aes-key="+aeskey+" > user-request-result.txt";
2143                std::system(cmd.c_str());
2144                std::ifstream file("./user-request-result.txt");
2145                std::stringstream buffer;
2146                buffer << file.rdbuf();
2147                std::string str = buffer.str();
2148                file.close();
2149                boost::filesystem::remove("./user-request-result.txt");
2150                if(!boost::algorithm::contains(str, "Success")){
2151                    throw std::runtime_error("Error! Something went wrong sending a request to "+targetUser
    ->username);
2152                }
2153            }
2154        }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.3.3.7 SendMultithreadedConnectedUserRequest()

```
void DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest (
            std::atomic< unsigned int > & activeRequestHandlingUsers,
            const Request & request,
            const std::shared_ptr< ConnectedUser > & targetUser,
            const string & extradata )
```

This function sends a request asynchronously to one of our ConnectedUsers through PHP.

Parameters

in	*activeRequestHandlingUsers*	An reference to an atomic integer that stores the number of multithreaded requests currently being sent.
in	*request*	The type of request to be sent.
in	*targetUser*	A reference to the ConnectedUser the request is to be sent to.
in	*extradata*	Additional data that may be required to be sent along with our request.

Definition at line 2163 of file DimensionlessFunctionality.cpp.

References AcceptConnectionRequest, DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::Cryptography::CreatePeerCertificateForUser(), DimensionlessFunctionality::↩ Networking::Multicast::Receiver::data, GetConnectedUserTorrent, GetConnectedUserXML, GetOurRemoteIP, Null, PerformSearch, RejectConnectionRequest, SendConnectionRequest, SendMessage, SetOurTorrent, SetOurXML, and DimensionlessFunctionality::ApplicationLogic::User::username.

Referenced by DimensionlessFunctionality::Networking::Nmap::ScanIPAddressesWithPort().

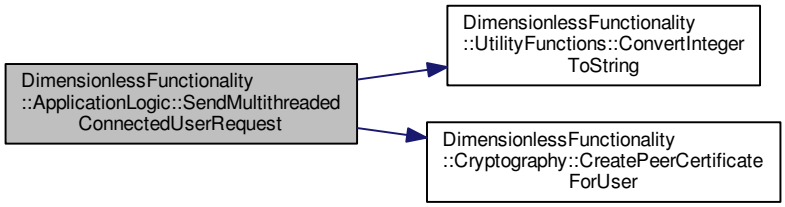
```
2164         {
2165             string serverurl;
2166             if(targetUser->regularDomain != ""){
2167                 serverurl = "http%3A%2F%2F"+targetUser->regularDomain+"%3A"+
     UtilityFunctions::ConvertIntegerToString(targetUser->webPort);
2168             } else {
2169                 serverurl = "http%3A%2F%2F"+targetUser->globalIPv4Address+"%3A"+
     UtilityFunctions::ConvertIntegerToString(targetUser->webPort);
2170             }
2171             string basicauth = targetUser->globalIPv4Address+":"+targetUser->AESiv;
2172 
2173             boost::random_device rd;
2174             // Removed a bunch of potentially harmful characters.
2175             std::string chars("
     abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^*()`~-_=+[{]}|;:.?");
2176             string aeskey = "";
2177             boost::variate_generator< boost::random_device&, boost::uniform_int<> >
2178             gen(rd, boost::uniform_int<>(0, chars.size()-1));
2179             for(int i = 0; i < 32; i++) {
2180                 aeskey += chars[gen()];
2181             }
2182 
2183             string source = CurrentUserSession::currentUser->toSourceTargetString();
2184             string target = targetUser->toSourceTargetString();
2185 
2186             string requesttype = "";
2187             string data = "";
2188             switch(request){
2189                 case Request::SendConnectionRequest:
2190                 {
2191                     requesttype = "send-connection-request";
2192                     Cryptography::CreatePeerCertificateForUser(
     CurrentUserSession::currentUser->username,targetUser->username,targetUser->AESiv,
     CurrentUserSession::currentUser->hashedPassword);
2193                     data = CurrentUserSession::currentUser->toDataString();
2194                     break;
2195                 }
2196                 case Request::AcceptConnectionRequest:
2197                 {
2198                     requesttype = "accept-connection-request";
2199                     Cryptography::CreatePeerCertificateForUser(
```

```
    CurrentUserSession::currentUser->username,targetUser->username,targetUser->AESiv,
    CurrentUserSession::currentUser->hashedPassword);
2200
2201                    std::ifstream torrentfile("./Users/"+CurrentUserSession::currentUser->username+"/"+
    CurrentUserSession::currentUser->username+".torrent");
2202                    std::stringstream torrentbuffer;
2203                    torrentbuffer << torrentfile.rdbuf();
2204                    std::string torrent = torrentbuffer.str();
2205                    torrentfile.close();
2206
2207                    std::ifstream privatekeyfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-privatekey.pem");
2208                    std::stringstream privatekeybuffer;
2209                    privatekeybuffer << privatekeyfile.rdbuf();
2210                    std::string peerprivatekey = privatekeybuffer.str();
2211                    privatekeyfile.close();
2212
2213                    std::ifstream certfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-cert.pem");
2214                    std::stringstream certbuffer;
2215                    certbuffer << certfile.rdbuf();
2216                    std::string peercert = certbuffer.str();
2217                    certfile.close();
2218
2219                    std::ifstream dhparamsfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/dhparams.pem");
2220                    std::stringstream dhparamsbuffer;
2221                    dhparamsbuffer << dhparamsfile.rdbuf();
2222                    std::string dhparams = dhparamsbuffer.str();
2223                    dhparamsfile.close();
2224
2225                    data = UtilityFunctions::ConvertIntegerToString
    (torrent.length())+","+
2226                           UtilityFunctions::ConvertIntegerToString
    (peerprivatekey.length())+","+
2227                           UtilityFunctions::ConvertIntegerToString
    (peercert.length())+","+
2228                           UtilityFunctions::ConvertIntegerToString
    (dhparams.length())+","+
2229                           torrent+","+
2230                           peerprivatekey+","+
2231                           peercert+","+
2232                           dhparams;
2233                    break;
2234                }
2235                case Request::RejectConnectionRequest:
2236                {
2237                    requesttype = "reject-connection-request";
2238                    break;
2239                }
2240                case Request::SendMessage:
2241                {
2242                    requesttype = "send-message";
2243                    data = extradata;
2244                    break;
2245                }
2246                case Request::PerformSearch:
2247                {
2248                    requesttype = "perform-search";
2249                    data = extradata;
2250                    break;
2251                }
2252                case Request::GetOurRemoteIP:
2253                {
2254                    requesttype = "get-remote-ip";
2255                    break;
2256                }
2257                case Request::SetOurXML:
2258                {
2259                    requesttype = "set-client-xml";
2260                    data = CurrentUserSession::currentUser->toDataString(extradata);
2261                    break;
2262                }
2263                case Request::SetOurTorrent:
2264                {
2265                    requesttype = "set-client-torrent";
2266                    Cryptography::CreatePeerCertificateForUser(
    CurrentUserSession::currentUser->username,targetUser->username,targetUser->AESiv,
    CurrentUserSession::currentUser->hashedPassword);
2267
2268                    std::ifstream torrentfile("./Users/"+CurrentUserSession::currentUser->username+"/"+
    CurrentUserSession::currentUser->username+".torrent");
2269                    std::stringstream torrentbuffer;
2270                    torrentbuffer << torrentfile.rdbuf();
2271                    std::string torrent = torrentbuffer.str();
2272                    torrentfile.close();
2273
```

```
2274                    std::ifstream privatekeyfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-privatekey.pem");
2275                    std::stringstream privatekeybuffer;
2276                    privatekeybuffer << privatekeyfile.rdbuf();
2277                    std::string peerprivatekey = privatekeybuffer.str();
2278                    privatekeyfile.close();
2279
2280                    std::ifstream certfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/private/"+targetUser->username+"-cert.pem");
2281                    std::stringstream certbuffer;
2282                    certbuffer << certfile.rdbuf();
2283                    std::string peercert = certbuffer.str();
2284                    certfile.close();
2285
2286                    std::ifstream dhparamsfile("./Users/"+CurrentUserSession::currentUser->username+"
    /CA/dhparams.pem");
2287                    std::stringstream dhparamsbuffer;
2288                    dhparamsbuffer << dhparamsfile.rdbuf();
2289                    std::string dhparams = dhparamsbuffer.str();
2290                    dhparamsfile.close();
2291
2292                    data = UtilityFunctions::ConvertIntegerToString
    (torrent.length())+","+
2293                            UtilityFunctions::ConvertIntegerToString
    (peerprivatekey.length())+","+
2294                            UtilityFunctions::ConvertIntegerToString
    (peercert.length())+","+
2295                            UtilityFunctions::ConvertIntegerToString
    (dhparams.length())+","+
2296                            torrent+","+
2297                            peerprivatekey+","+
2298                            peercert+","+
2299                            dhparams;
2300                    break;
2301                }
2302                case Request::GetConnectedUserXML:
2303                {
2304                    requesttype = "get-server-xml";
2305                    break;
2306                }
2307                case Request::GetConnectedUserTorrent:
2308                {
2309                    requesttype = "get-server-torrent";
2310                    break;
2311                }
2312                case Request::Null:
2313                default:
2314                {
2315                    throw std::runtime_error("Error! Invalid request.");
2316                }
2317            }
2318
2319            string cmd = "./Executables/php-cgi -f ../Users/"+CurrentUserSession::currentUser->username+"
    /data/post-request-sender.php server-url="+serverurl+" basic-auth="+basicauth+" source="+source+" target="+
    target+" request-type="+requesttype+" data="+data+" aes-key="+aeskey+" > user-request-result.txt";
2320            std::system(cmd.c_str());
2321            std::ifstream file("./user-request-result.txt");
2322            std::stringstream buffer;
2323            buffer << file.rdbuf();
2324            std::string str = buffer.str();
2325            file.close();
2326            boost::filesystem::remove("./user-request-result.txt");
2327            if(!boost::algorithm::contains(str, "Success")){
2328                string serverurl;
2329                if(targetUser->onionDomain != ""){
2330                    serverurl = "http%3A%2F%2F"+targetUser->onionDomain+"%3A"+
    UtilityFunctions::ConvertIntegerToString(targetUser->webPort);
2331                } else {
2332                    throw std::runtime_error("Error! Something went wrong sending a request to "+targetUser
    ->username);
2333                }
2334                string cmd = "./Executables/php-cgi -f ../Users/"+CurrentUserSession::currentUser->username
    +"/data/post-request-sender.php server-url="+serverurl+" basic-auth="+basicauth+" source="+source+" target="
    +target+" request-type="+requesttype+" data="+data+" aes-key="+aeskey+" > user-request-result.txt";
2335                std::system(cmd.c_str());
2336                std::ifstream file("./user-request-result.txt");
2337                std::stringstream buffer;
2338                buffer << file.rdbuf();
2339                std::string str = buffer.str();
2340                file.close();
2341                boost::filesystem::remove("./user-request-result.txt");
2342                if(!boost::algorithm::contains(str, "Success")){
2343                    throw std::runtime_error("Error! Something went wrong sending a request to "+targetUser
    ->username);
2344                }
2345            }
2346
```

```
2347             if(activeRequestHandlingUsers != 0){
2348                 activeRequestHandlingUsers--;
2349             }
2350         }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.3.3.8 VerifyLoginCredentials()

```
bool DimensionlessFunctionality::ApplicationLogic::VerifyLoginCredentials (
            const string & username,
            const string & password )
```

This function checks the username and password provided to it against what is stored on disk.

Parameters

in	*username*	The username of the user to check for.
in	*password*	The password for the user to check.

Returns

A boolean indicating if our login credentials are correct or not.

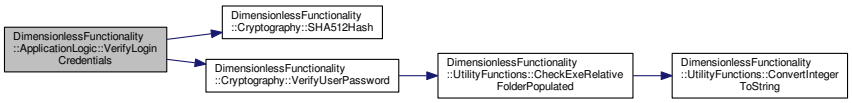
Definition at line 2472 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::Cryptography::SHA512Hash(), and DimensionlessFunctionality::↩ Cryptography::VerifyUserPassword().

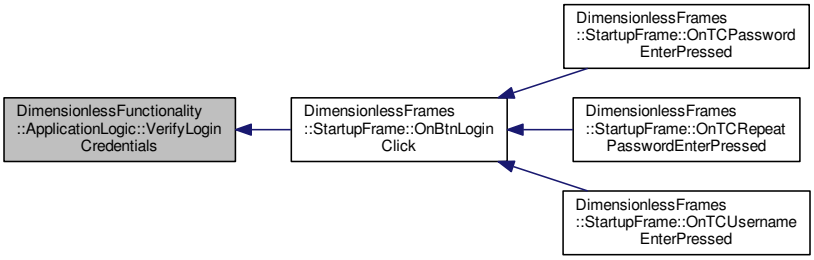
Referenced by DimensionlessFrames::StartupFrame::OnBtnLoginClick().

```
2473         {
2474             return Cryptography::VerifyUserPassword(username,
     Cryptography::SHA512Hash(password));
2475         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.4 DimensionlessFunctionality::Compression Namespace Reference

Functions

- string Compress (string dataToCompress)
- string Decompress (string dataToDecompress)

11.4.1 Detailed Description

The Compression namespace stores all of the functions that allow for compression to be performed on data.

11.4.2 Function Documentation

11.4.2.1 Compress()

```
string DimensionlessFunctionality::Compression::Compress (
            string dataToCompress )
```

This function compresses data using Zlib compression.

Parameters

in	*DataToCompress*	A string containing data to be compressed.

Returns

A string containing the compressed data.

Definition at line 241 of file DimensionlessFunctionality.cpp.

```
242         {
243             stringstream src(dataToCompress);
244             stringstream ret;
245             boost::iostreams::filtering_streambuf<boost::iostreams::input> in;
246             in.push(boost::iostreams::zlib_compressor());
247             in.push(src);
248             boost::iostreams::copy(in, ret);
249 
250             return ret.str();
251         }
```

11.4.2.2 Decompress()

```
string DimensionlessFunctionality::Compression::Decompress (
            string dataToDecompress )
```

This function decompresses Zlib compressed data.

Parameters

in	*DataToDecompress*	A string containing the Zlib compressed data to decompress.

Returns

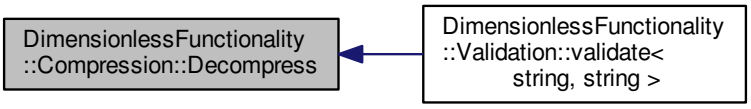
A Zlib decompressed string.

Definition at line 258 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::Validation::validate< string, string >().

```
259         {
260             stringstream src(dataToDecompress);
261             stringstream ret;
262             boost::iostreams::filtering_streambuf<boost::iostreams::input> in;
263             in.push(boost::iostreams::zlib_decompressor());
264             in.push(src);
265             boost::iostreams::copy(in, ret);
266 
267             return ret.str();
268         }
```

Here is the caller graph for this function:



11.5 DimensionlessFunctionality::Cryptography Namespace Reference

Functions

- string SHA512Hash (const string &dataToHash)
- bool VerifyUserPassword (const string &username, const string &password)
- bool CreateRootCAForUser (const string &username, const string &password)
- bool CreatePeerCertificateForUser (const string &username, const string &targetUsername, const string &aesiv, const string &password)

11.5.1 Detailed Description

The Cryptography namespace stores all of the functions that allow for encryption to be performed on data.

11.5.2 Function Documentation

11.5.2.1 CreatePeerCertificateForUser()

```
bool DimensionlessFunctionality::Cryptography::CreatePeerCertificateForUser (
            const string & username,
            const string & targetUsername,
            const string & aesiv,
            const string & password )
```

This function creates a self-signed SSL certificate and saves it along with its Diffie-Hellman parameters as separate .pem files in a given directory (for libtorrent SSL torrents).

Parameters

in	*username*	The username to store the certificate and dh_params .pem files for.
in	*targetUsername*	The username of the peer to create certificates for.
in	*aesiv*	The aesiv of the peer so they can decrypt their provided private key.
in	*password*	The password of our current user to be able to generate new certificates with our CA.

Returns

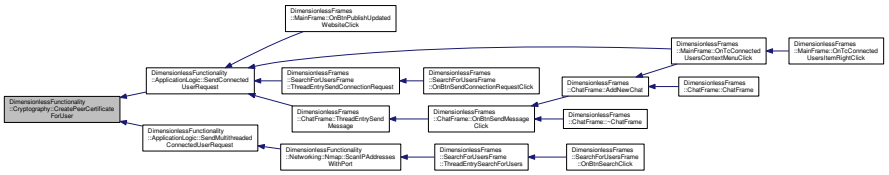
A boolean value indicating whether the peer certificate creation process succeeded.

Definition at line 395 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest(), and DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest().

```
396         {
397             string cmd = "./Executables/openssl genrsa -aes256 -passout pass:"+aesiv+" -out ./Users/"+
    username+"/CA/private/"+targetUsername+"-privatekey.pem 4096";
398             std::system(cmd.c_str());
399             cmd = "./Executables/openssl req -new -sha512 -config ./Users/"+username+"/openssl.cnf -key
     ./Users/"+username+"/CA/private/"+targetUsername+"-privatekey.pem -out ./Users/"+username+"/CA/private/"+
    targetUsername+"-req.pem -subj \"/C=ED/ST=ED/L=ED/O=Eclectic Dimensions/OU=Dimensionless/CN="+username+"\" -passin
     pass:"+aesiv;
400             std::system(cmd.c_str());
401             cmd = "./Executables/openssl ca -config ./Users/"+username+"/openssl.cnf -days 30 -batch
     -keyfile ./Users/"+username+"/CA/private/cakey.pem -passin pass:"+password+" -policy policy_anything -out ./Users/
    "+username+"/CA/private/"+targetUsername+"-cert.pem -infiles ./Users/"+username+"/CA/private/"+
    targetUsername+"-req.pem";
402             std::system(cmd.c_str());
403             return true;
404         }
```

Here is the caller graph for this function:



11.5.2.2 CreateRootCAForUser()

```
bool DimensionlessFunctionality::Cryptography::CreateRootCAForUser (
            const string & username,
            const string & password )
```

This function creates a Certificate Authority (CA) for a user.

Parameters

in	*username*	A string containing the username of the user to create a CA for.
in	*password*	A string containing the plaintext password to use to encrypt the CA's private key.

Returns

A boolean denoting whether the CA was successfully created.

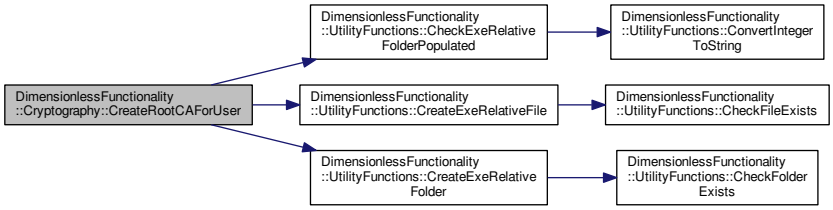
Definition at line 347 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated(), DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile(), and DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFolder().

Referenced by DimensionlessFunctionality::ApplicationLogic::NewUserRegistration().

```
348         {
349             if(!UtilityFunctions::CheckExeRelativeFolderPopulated
    ("Users/"+username+"/CA")){
350                 UtilityFunctions::CreateExeRelativeFolder("Users/"
    +username+"/CA");
351                 UtilityFunctions::CreateExeRelativeFolder("Users/"
    +username+"/CA/certs");
352                 UtilityFunctions::CreateExeRelativeFolder("Users/"
    +username+"/CA/crl");
353                 UtilityFunctions::CreateExeRelativeFolder("Users/"
    +username+"/CA/newcerts");
354                 UtilityFunctions::CreateExeRelativeFolder("Users/"
    +username+"/CA/private");
355                 UtilityFunctions::CreateExeRelativeFile("Users/"+
    username+"/CA/certs/temp.txt");
356                 UtilityFunctions::CreateExeRelativeFile("Users/"+
    username+"/CA/crl/temp.txt");
357                 UtilityFunctions::CreateExeRelativeFile("Users/"+
    username+"/CA/index.txt");
358                 UtilityFunctions::CreateExeRelativeFile("Users/"+
    username+"/CA/index.txt.attr");
359                 UtilityFunctions::CreateExeRelativeFile("Users/"+
    username+"/CA/crlnumber", "01");
360                 // necessary?
361                 //UtilityFunctions::CreateExeRelativeFile("Users/"+username+"/CA/serial","1");
362 
363                 std::ifstream file("./Executables/openssl.cnf");
364                 std::stringstream buffer;
365                 buffer << file.rdbuf();
366                 std::string str = buffer.str();
367                 file.close();
368                 boost::replace_all(str, "./demoCA", ApplicationInfo::GetExecutablePath()+"Users/"+username+
    "/CA");
369                 boost::replace_last(str, "= default", "= sha512");
370                 boost::replace_all(str, "2048", "4096");
371                 UtilityFunctions::CreateExeRelativeFile("Users/"+
    username+"/openssl.cnf", str);
372 
373                 string cmd = "./Executables/openssl genrsa -aes256 -passout pass:"+password+" -out ./Users/
    "+username+"/CA/private/cakey.pem 4096";
374                 std::system(cmd.c_str());
375                 cmd = "./Executables/openssl req -new -sha512 -config ./Users/"+username+"/openssl.cnf -key
     ./Users/"+username+"/CA/private/cakey.pem -out ./Users/"+username+"/CA/private/careq.pem -subj \"
    /C=ED/ST=ED/L=ED/O=Eclectic Dimensions/OU=Dimensionless/CN="+username+"\" -passin pass:"+password;
376                 std::system(cmd.c_str());
377                 cmd = "./Executables/openssl ca -config ./Users/"+username+"/openssl.cnf -create_serial
     -out ./Users/"+username+"/CA/cacert.pem -days 30 -batch -keyfile ./Users/"+username+"/CA/private/cakey.pem
     -passin pass:"+password+" -selfsign -extensions v3_ca -infiles ./Users/"+username+"/CA/private/careq.pem";
378                 std::system(cmd.c_str());
379                 cmd = "./Executables/openssl dhparam -outform PEM -out ./Users/"+username+"/CA/dhparams.pem
     2048";
380                 std::system(cmd.c_str());
381                 return true;
382             } else {
383                 return false;
384             }
385         }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.5.2.3 SHA512Hash()

```
string DimensionlessFunctionality::Cryptography::SHA512Hash (
            const string & dataToHash )
```

This function hashes string data using SHA512.

Parameters

in	*dataToHash*	A string containing the data to hash.

Returns

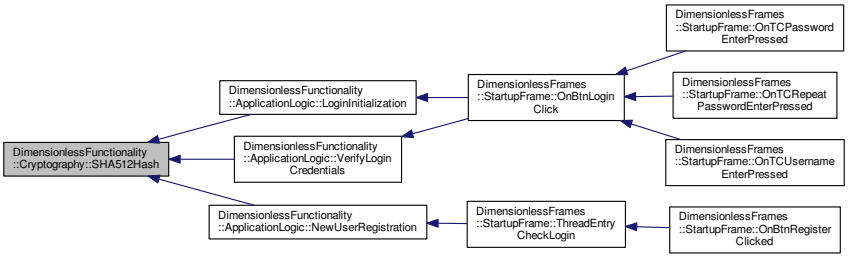
A SHA512 hashed string.

Definition at line 281 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and DimensionlessFunctionality::ApplicationLogic::VerifyLoginCredentials().

```
282         {
283             #if defined(LINUX) || defined(ANDROID)
284                 string cmd = "echo -n \""+dataToHash+"\" | ./Executables/openssl sha512 > sha512.txt";
285             #elif defined(MAC)
286                 // necessary?
287                 string cmd = "echo -n \""+dataToHash+"\" | ./Executables/openssl sha512 > sha512.txt";
288             #elif defined(WINDOWS)
289                 // Check this.
290                 string cmd = "echo \""+dataToHash+"\" | ./Executables/openssl.exe sha512 > sha512.txt";
291             #else
292                 #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
    preprocessors. Otherwise we cannot continue compilation."
293             #endif
294
295             std::system(cmd.c_str());
296             std::ifstream ifout(ApplicationInfo::GetExecutablePath()+"sha512.txt");
297             std::stringstream buffer;
298             buffer << ifout.rdbuf();
299             ifout.close();
300             string out = buffer.str();
301             boost::filesystem::remove(ApplicationInfo::GetExecutablePath()+"sha512.txt");
302
303             if(out.length() > 0){
304                 out = out.substr(0,out.length() - 1);
305                 if(out.substr(0,9) == "(stdin)= "){
306                     out = out.substr(9, out.length());
307                 }
308                 return out;
309             } else {
310                 throw std::runtime_error("Error - is an openssl executable present in the Executables
    directory at the root of this application?");
311             }
312         }
```

Here is the caller graph for this function:



11.5.2.4 VerifyUserPassword()

```
bool DimensionlessFunctionality::Cryptography::VerifyUserPassword (
            const string & username,
            const string & password )
```

This function verifies a password for a username by attempting to decrypt the particular user's private key.

Parameters

in	*username*	A string containing the username of the user to check.
in	*password*	A string containing the plaintext password to verify.

Returns

A boolean denoting whether the password provided is correct for the given username.

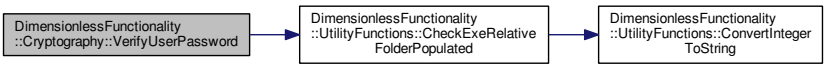
Definition at line 320 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated().

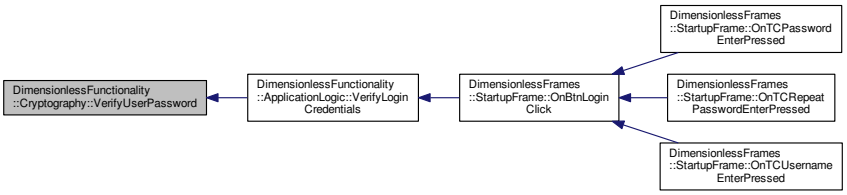
Referenced by DimensionlessFunctionality::ApplicationLogic::VerifyLoginCredentials().

```
321         {
322             if(UtilityFunctions::CheckExeRelativeFolderPopulated
     ("Users/"+username)){
323                 string cmd = "./Executables/openssl rsa -text -passin pass:"+password+" -in ./Users/"+
     username+"/CA/private/cakey.pem > login.txt";
324                 std::system(cmd.c_str());
325                 std::ifstream ifout(ApplicationInfo::GetExecutablePath()+"login.txt");
326                 std::stringstream buffer;
327                 buffer << ifout.rdbuf();
328                 ifout.close();
329                 string out = buffer.str();
330                 boost::filesystem::remove(ApplicationInfo::GetExecutablePath()+"login.txt");
331                 if(boost::algorithm::contains(out,"-----BEGIN RSA PRIVATE KEY-----")){
332                     return true;
333                 } else {
334                     return false;
335                 }
336             } else {
337                 return false;
338             }
339         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.6 DimensionlessFunctionality::Networking Namespace Reference

Namespaces

- Curl
- GeoIP
- Libtorrent
- Multicast
- Nmap
- OpenVPN
- PHPFPM
- Tor
- WebBrowser
- WebServer

11.6.1 Detailed Description

The Networking namespace stores all of the functions that allow for data to be sent, received and processed by us and various other connected devices.

11.7 DimensionlessFunctionality::Networking::Curl Namespace Reference

Functions

- string GetWebsiteTitle (const string &ipOrURL)
- string GetTorWebsiteTitle (const string &onionURL)
- string UpdateFreeDNSAfraidOrgDomain (const string &freeDNSAfraidOrgURL)

11.7.1 Detailed Description

The Curl namespace stores all of the functions that communicate with Internet hosted websites.

11.7.2 Function Documentation

11.7.2.1 GetTorWebsiteTitle()

```
string DimensionlessFunctionality::Networking::Curl::GetTorWebsiteTitle (
            const string & onionURL )
```

This function retrieves the title of an Tor hosted website accessible by a Uniform Resource Locator (URL) with a .onion top level domain name or suffix.

Parameters

in	*onionURL*	The onion URL of the website to access.

Returns

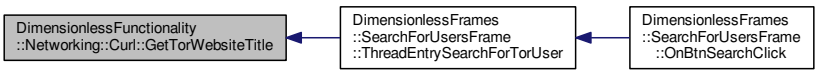
A string value containing the retrieved website title (if any).

Definition at line 442 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForTorUser().

```
443             {
444                 string cmd = "./Executables/curl --socks5-hostname 127.0.0.1:9050 " + onionURL + " >
        ./curl.txt";
445                 std::system(cmd.c_str());
446                 std::ifstream file("./curl.txt");
447                 std::stringstream buffer;
448                 buffer << file.rdbuf();
449                 std::string str = buffer.str();
450                 file.close();
451                 boost::filesystem::remove("./curl.txt");
452                 boost::replace_first(str, "<html><title>", "");
453                 boost::replace_last(str, "</html></title>", "");
454                 return str;
455             }
```

Here is the caller graph for this function:



11.7.2.2 GetWebsiteTitle()

```
string DimensionlessFunctionality::Networking::Curl::GetWebsiteTitle (
            const string & ipOrURL )
```

This function retrieves the title of a website accessible by a Uniform Resource Locator (URL).

Parameters

in	*ipOrURL*	The IP address or URL of the website to access.

Returns

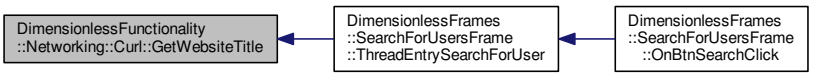
A string value containing the retrieved website title (if any).

Definition at line 422 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUser().

```
423             {
424                 string cmd = "./Executables/curl " + ipOrURL + " > ./curl.txt";
425                 std::system(cmd.c_str());
426                 std::ifstream file("./curl.txt");
427                 std::stringstream buffer;
428                 buffer << file.rdbuf();
429                 std::string str = buffer.str();
430                 file.close();
431                 boost::filesystem::remove("./curl.txt");
432                 boost::replace_first(str, "<html><title>", "");
433                 boost::replace_last(str, "</html></title>", "");
434                 return str;
435             }
```

Here is the caller graph for this function:



11.7.2.3 UpdateFreeDNSAfraidOrgDomain()

```
string DimensionlessFunctionality::Networking::Curl::UpdateFreeDNSAfraidOrgDomain (
            const string & freeDNSAfraidOrgURL )
```

This function updates our IP address with the FreeDNS service.

Parameters

in	*freeDNSAfraidOrgURL*	The FreeDNS URL (containing API key) to send a HTTP request to.

Returns

A string value containing the result of updating our IP address with the FreeDNS service.

Definition at line 462 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick().

```
463             {
464                 string cmd = "./Executables/curl/bin/curl "+freeDNSAfraidOrgURL+" > ./curl.txt";
465                 std::system(cmd.c_str());
466                 std::ifstream file("./curl.txt");
467                 std::stringstream buffer;
468                 buffer << file.rdbuf();
469                 std::string str = buffer.str();
470                 file.close();
471                 boost::filesystem::remove("./curl.txt");
472                 return str;
473             }
```

Here is the caller graph for this function:

11.8 DimensionlessFunctionality::Networking::GeoIP Namespace Reference

Functions

- void UpdateIPv4AddressList ()
- std::map< string, std::vector< string > > GetCityCountryList ()
- std::vector< string > GetIPv4AddressesForCityXorCountry (const string &cityxorcountry)
- std::vector< string > GetIPv4AddressesForCountryAndCity (const string &country, const string &city)

11.8.1 Detailed Description

The GeoIP namespace stores all of the functions that access and retrieve the IP addresse databases from Maxmind's free GeoLite2 product (http://dev.maxmind.com/geoip/geoip2/geolite2/).

11.8.2 Function Documentation

11.8.2.1 GetCityCountryList()

```
std::map< string, std::vector< string > > DimensionlessFunctionality::Networking::GeoIP::↩
GetCityCountryList ( )
```

This function obtains a map of all countries and cities that our GeoIP databases provide IP addresses for.

Returns

A map consisting of countries and a vector of their respective cities retrieved from our GeoIP databases.

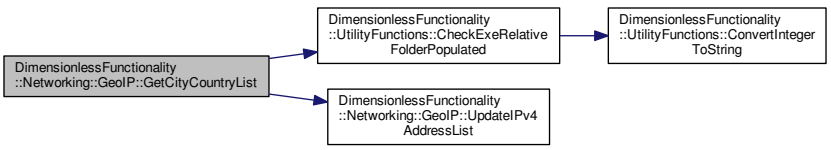
Definition at line 494 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated(), and UpdateIPv4↩AddressList().

Referenced by DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrame().

```
495             {
496                 if(!UtilityFunctions::CheckExeRelativeFolderPopulated
    ("GeoLite")){
497                     UpdateIPv4AddressList();
498                 }
499                 std::map< string, std::vector<string> > CountryCityMap;
500                 std::ifstream cityfile("./GeoLite/GeoLite2-City-Locations-en.csv");
501                 bool firstline = true;
502                 for (std::string line; std::getline(cityfile, line);)
503                 {
504                     if(firstline){
505                         firstline = false;
506                         continue;
507                     }
508                     std::vector<string> tempWords;
509                     boost::replace_all(line, "\"","");
510                     boost::split(tempWords, line, boost::is_any_of(","));
511                     if(tempWords[5] == " " || tempWords[5] == ""){
512                         tempWords[5] = "Unknown Country"; //missing something in line.
513                     }
514                     CountryCityMap[tempWords[5]].push_back(tempWords[0]);
515                     CountryCityMap[tempWords[5]].push_back(tempWords[10]);
516                 }
517                 cityfile.close();
518                 return CountryCityMap;
519             }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.8.2.2 GetIPv4AddressesForCityXorCountry()

```
std::vector< string > DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCity↩
XorCountry (
            const string & cityxorcountry )
```

This function obtains a string vector of IPv4 address for a particular city xor country.

Returns

A string vector of IPv4 addresses for the desired city xor country.

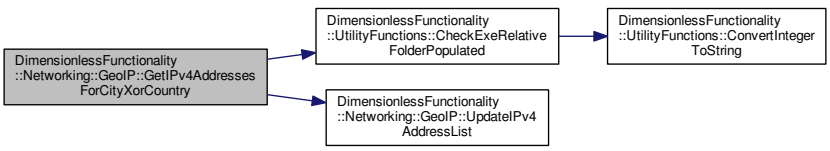
Definition at line 525 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated(), and UpdateIPv4↩AddressList().

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick().

```
526             {
527                 if(!UtilityFunctions::CheckExeRelativeFolderPopulated
    ("GeoLite")){
528                     UpdateIPv4AddressList();
529                 }
530                 std::vector<string> geonameIDs;
531                 std::ifstream cityfile("./GeoLite/GeoLite2-City-Locations-en.csv");
532                 for (std::string line; std::getline(cityfile, line);)
533                 {
534                     if(boost::algorithm::contains(line, cityxorcountry)){
535                         std::vector<string> tempWords;
536                         boost::split(tempWords, line, boost::is_any_of(","));
537                         geonameIDs.push_back(tempWords[0]);
538                     }
539                 }
540                 cityfile.close();
541 
542                 std::vector<string> IPv4Addresses;
543                 for(unsigned int i = 0; i < geonameIDs.size(); i++){
544                     std::ifstream ipv4file("./GeoLite/GeoLite2-City-Blocks-IPv4.csv");
545                     for (std::string line; std::getline(ipv4file, line);)
546                     {
547                         if(boost::algorithm::contains(line, geonameIDs[i])){
548                             std::vector<string> tempWords;
549                             boost::split(tempWords, line, boost::is_any_of(","));
550                             IPv4Addresses.push_back(tempWords[0]);
551                         }
552                     }
553                     ipv4file.close();
554                 }
555 
556                 return IPv4Addresses;
557             }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.8.2.3 GetIPv4AddressesForCountryAndCity()

```
std::vector< string > DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesFor↩
CountryAndCity (
            const string & country,
            const string & city )
```

This function obtains a string vector of IPv4 address for a particular country and city.

Returns

A string vector of IPv4 addresses for the desired country and city.

Definition at line 563 of file DimensionlessFunctionality.cpp.

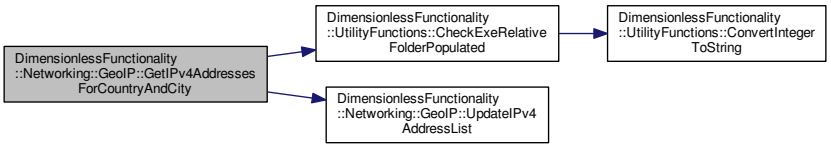
References DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated(), and UpdateIPv4↩AddressList().

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick().

```
564             {
565                 if(!UtilityFunctions::CheckExeRelativeFolderPopulated
    ("GeoLite")){
566                     UpdateIPv4AddressList();
567                 }
568                 std::vector<string> geonameIDs;
569                 std::ifstream cityfile("./GeoLite/GeoLite2-City-Locations-en.csv");
570                 for (std::string line; std::getline(cityfile, line);)
571                 {
572                     if(country.size() > 0){
573                         if(boost::algorithm::contains(line, city) && boost::algorithm::contains(line,
    country)){
574                             std::vector<string> tempWords;
575                             boost::split(tempWords, line, boost::is_any_of(","));
576                             geonameIDs.push_back(tempWords[0]);
577                         }
578                     } else {
579                         if(boost::algorithm::contains(line, city)){
580                             std::vector<string> tempWords;
581                             boost::split(tempWords, line, boost::is_any_of(","));
582                             geonameIDs.push_back(tempWords[0]);
583                         }
584                     }
585                 }
586                 cityfile.close();
587 
588                 std::vector<string> IPv4Addresses;
589                 for(unsigned int i = 0; i < geonameIDs.size(); i++){
590                     std::ifstream ipv4file("./GeoLite/GeoLite2-City-Blocks-IPv4.csv");
591                     for (std::string line; std::getline(ipv4file, line);)
```

```
592                     {
593                         if(boost::algorithm::contains(line, geonameIDs[i])){
594                             std::vector<string> tempWords;
595                             boost::split(tempWords, line, boost::is_any_of(","));
596                             IPv4Addresses.push_back(tempWords[0]);
597                         }
598                     }
599                     ipv4file.close();
600                 }
601 
602                 return IPv4Addresses;
603             }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.8.2.4 UpdateIPv4AddressList()

```
void DimensionlessFunctionality::Networking::GeoIP::UpdateIPv4AddressList ( )
```

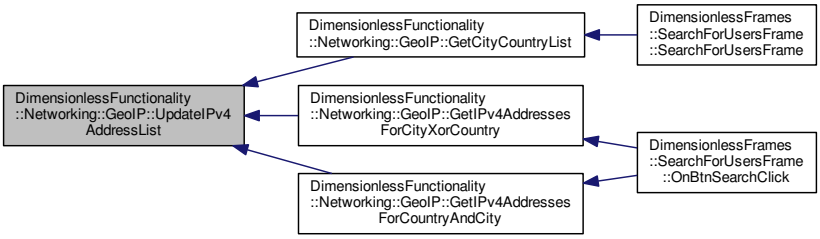
This function obtains the latest city database files from Maxmind's free GeoLite2 product (http://dev.↩ maxmind.com/geoip/geoip2/geolite2/).

Definition at line 484 of file DimensionlessFunctionality.cpp.

Referenced by GetCityCountryList(), GetIPv4AddressesForCityXorCountry(), and GetIPv4AddressesForCountry↩ AndCity().

```
485             {
486                 string cmd = "cd ./Executables; ./php -f geolite-update.php";
487                 std::system(cmd.c_str());
488             }
```

Here is the caller graph for this function:



11.9 DimensionlessFunctionality::Networking::Libtorrent Namespace Reference

Namespaces

- UPnP

Functions

- vector< string > ScanForTorrentFiles (const string &scanDirectoryRoot)
- bool CreateTorrentFileFilter (std::string const &file)
- void CreateTorrent (const string &username, const string &persistentAuthComment)
- void CreateSSLTorrent (const string &username, const string &persistentAuthComment, const string &root↩ CertificatePath)
- void AddTorrent (const string &torrentPath, const string &peerCertificatePath, const string &privateKeyPath, const string &dhParamsPath, const string &ourAESIV)
- void RemoveTorrent (const string &targetUsername, const bool &deleteFiles)
- void LibtorrentThreadEntry (const string &targetUsername, const string &targetPassword)
- void StartLibtorrentThread (const string &targetUsername, const string &targetPassword)
- void StopLibtorrentThread ()
- static std::atomic< bool > stopLibtorrentThread (false)

  This atomic boolean variable decides whether it is time to stop the libtorrent thread.

Variables

- static std::thread libtorrentThread

  This variable stores out libtorrent thread instance.
- static std::unique_ptr< session > currentSession

  This smart unique pointer stores a reference to a libtorrent session.
- static std::unique_ptr< settings_pack > currentSettings

  This smart unique pointer stores a reference to our current libtorrent session's settings.

11.9.1 Detailed Description

The Libtorrent namespace stores all of the functions that manipulate the bittorrent protocol based backend to the application.

11.9.2 Function Documentation

11.9.2.1 AddTorrent()

```
void DimensionlessFunctionality::Networking::Libtorrent::AddTorrent (
            const string & torrentPath,
            const string & peerCertificatePath,
            const string & privateKeyPath,
            const string & dhParamsPath,
            const string & ourAESIV )
```

This function asynchronously adds a torrent for our libtorrent thread to process and download.

Parameters

in	*torrentPath*	The path to the torrent file to add.
in	*peerCertificatePath*	The path to a certificate that will allow our libtorrent thread to download an ssl torrent.
in	*privateKeyPath*	The path to a certificate's private key that will allow our libtorrent thread to download an ssl torrent.
in	*dhParamsPath*	The path to the DH parameters that will allow our libtorrent thread to download an ssl torrent.
in	*ourAESIV*	The AES IV for our local user.

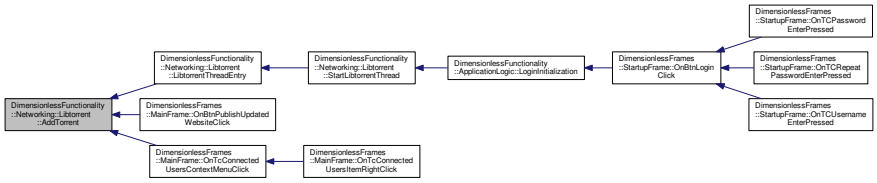
Definition at line 773 of file DimensionlessFunctionality.cpp.

References currentSession.

Referenced by LibtorrentThreadEntry(), DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
774             {
775                 libtorrent::add_torrent_params p;
776                 libtorrent::error_code ec;
777 
778                 p.ti = boost::make_shared<libtorrent::torrent_info>(torrentPath, boost::ref(ec), 0);
779                 p.save_path = "./Users/";
780 
781                 // Only for seeders like the original torrent author. Not really necessary even then but
      maybe some minor performance speedups?
782                 //p.flags |= libtorrent::add_torrent_params::flag_seed_mode;
783                 //p.seed_mode = true;
784 
785                 if(ec){
786                     throw std::runtime_error("libtorrent: failed to add torrent");
787                 } else {
788                     unsigned int oldSize = currentSession->get_torrents().size();
789                     currentSession->async_add_torrent(p);
790 
791                     while(currentSession->get_torrents().size() <= oldSize){
792                         // wait.
793                     }
794 
795                     currentSession->get_torrents()[currentSession->get_torrents
      ().size()-1].set_ssl_certificate(peerCertificatePath, privateKeyPath, dhParamsPath, ourAESIV);
796                 }
797             }
```

Here is the caller graph for this function:



11.9.2.2 CreateSSLTorrent()

```
void DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent (
            const string & username,
            const string & persistentAuthComment,
            const string & rootCertificatePath )
```

This function creates a SSL torrent and saves it.

Parameters

in	*username*	The username for which to create a torrent for.
in	*persistentAuthComment*	A torrent comment that for our purposes should be the AES IV of the user whose username is also passed into this function.
in	*rootCertificatePath*	The path to the CA certificate to be embedded in the torrent.

Definition at line 731 of file DimensionlessFunctionality.cpp.

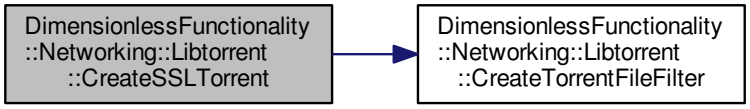
References DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::connectedUsers, and CreateTorrentFileFilter().

Referenced by LibtorrentThreadEntry(), and DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick().

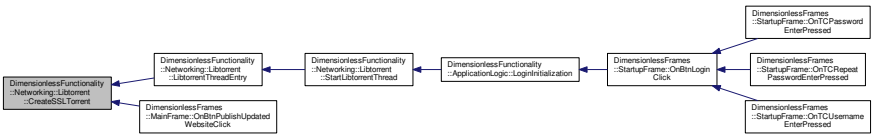
```
732             {
733                 // remove old torrent file.
734                 boost::filesystem::remove("./Users/"+username+"/"+username+".torrent");
735 
736                 libtorrent::file_storage fs;
737 
738                 // recursively adds files in directories (or single files themselves)
739                 add_files(fs, "./Users/"+username+"/", CreateTorrentFileFilter);
740 
741                 libtorrent::create_torrent t(fs);
742 
743                 for(std::shared_ptr<ApplicationLogic::ConnectedUser>& user :
      ApplicationLogic::CurrentUserSession::connectedUsers){
744                     t.add_node(std::pair<std::string,int>(user->globalIPv4Address, user->libtorrentPort));
745                 }
746                 t.set_creator(username.c_str());
747                 t.set_comment(persistentAuthComment.c_str());
748                 // private torrent flag - currently unenforceable though, also turns off DHT and LSD.
749                 //t.set_priv(true);
750 
751                 // read and set the root CA certificate for our torrent (This same certificate can be used
       to seed the torrent as a peer).
752                 std::ifstream file(rootCertificatePath);
753                 std::stringstream buffer;
```

```
754                 buffer << file.rdbuf();
755                 std::string rootCertificate = buffer.str();
756                 t.set_root_cert(rootCertificate);
757 
758                 // reads the files and calculates the hashes
759                 set_piece_hashes(t, "./Users/");
760 
761                 ofstream out("./Users/"+username+"/"+username+".torrent", std::ios_base::binary);
762                 bencode(std::ostream_iterator<char>(out), t.generate());
763             }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.9.2.3 CreateTorrent()

```
void DimensionlessFunctionality::Networking::Libtorrent::CreateTorrent (
            const string & username,
            const string & persistentAuthComment )
```

This function creates a torrent for a user and saves it (Non-ssl variant - this is currently not used in the application and do not use as it lessens the control over who can download your torrent).

Parameters

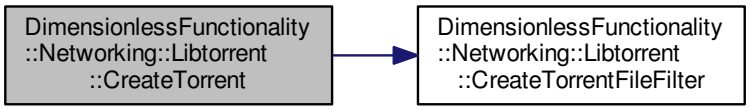
in	*username*	The username for which to create a torrent for.
in	*persistentAuthComment*	A torrent comment that for our purposes should be the AES IV of the user whose username is also passed into this function.

Definition at line 698 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::connectedUsers, and CreateTorrentFileFilter().

```
699             {
700                 // remove old torrent file.
701                 boost::filesystem::remove("./Users/"+username+"/"+username+".torrent");
702 
703                 libtorrent::file_storage fs;
704 
705                 // recursively adds files in directories (or single files themselves)
706                 add_files(fs, "./Users/"+username+"/", CreateTorrentFileFilter);
707 
708                 libtorrent::create_torrent t(fs);
709 
710                 for(std::shared_ptr<ApplicationLogic::ConnectedUser>& user :
    ApplicationLogic::CurrentUserSession::connectedUsers){
711                     t.add_node(std::pair<std::string,int>(user->globalIPv4Address, user->libtorrentPort));
712                 }
713                 t.set_creator(username.c_str());
714                 t.set_comment(persistentAuthComment.c_str());
715                 // private torrent flag - currently unenforceable though, also turns off DHT and LSD.
716                 //t.set_priv(true);
717 
718                 // reads the files and calculates the hashes
719                 set_piece_hashes(t, "./Users/"+username+"/");
720 
721                 ofstream out("./Users/"+username+"/"+username+".torrent", std::ios_base::binary);
722                 bencode(std::ostream_iterator<char>(out), t.generate());
723             }
```

Here is the call graph for this function:



11.9.2.4 CreateTorrentFileFilter()

```
bool DimensionlessFunctionality::Networking::Libtorrent::CreateTorrentFileFilter (
            std::string const & file )
```

This function prevents certain files from being added to a torrent that is being created.

Parameters

in	*file*	A string that contains the current file to filter.

Returns

A boolean value indicating whether the file should be included in a torrent.

Definition at line 683 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::currentUser.

Referenced by CreateSSLTorrent(), and CreateTorrent().

```
684             {
685                 // CA, hiddenservice directories have too many sensitive files, we send what is necessary -
     nothing more, nothing less.
686                 if(boost::algorithm::contains(file,ApplicationLogic::CurrentUserSession::currentUser->
    username+".xml") || boost::algorithm::contains(file,"/CA/") || boost::algorithm::contains(file,"/hiddenservice/")
    ){
687                     return false;
688                 } else {
689                     return true;
690                 }
691             }
```

Here is the caller graph for this function:

11.9.2.5 LibtorrentThreadEntry()

```
void DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry (
            const string & targetUsername,
            const string & targetPassword )
```

This function is the entry for our application's libtorrent thread and runs for the duration of the application's running time or until the current user logs out.

Parameters

in	*targetUsername*	The username for our currently logged in user.
in	*targetPassword*	The password for our currently logged in user.

Definition at line 824 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::Networking::Libtorrent::UPnP::AddPortMapping(), AddTorrent(), DimensionlessFunctionality::UtilityFunctions::CheckFileExists(), DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), CreateSSLTorrent(), currentSession, currentSettings, DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::currentUser, DimensionlessFunctionality::XML::ReadXMLChildValue(), and stopLibtorrentThread().

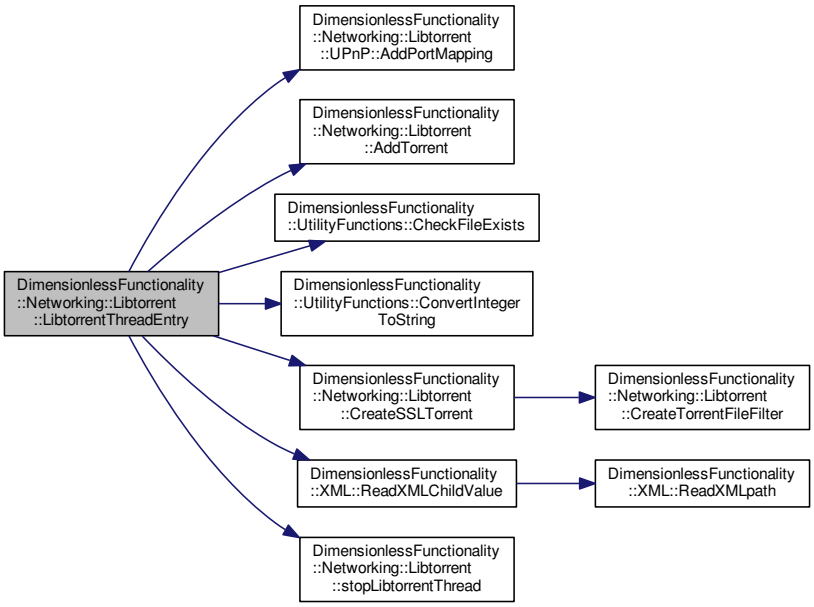
Referenced by StartLibtorrentThread().

```
825             {
826                 currentSettings = std::unique_ptr<libtorrent::settings_pack>(new
    libtorrent::settings_pack);
827                 currentSettings->set_str(settings_pack::dht_bootstrap_nodes, "");
828                 currentSettings->set_int(settings_pack::alert_mask,
    libtorrent::alert::all_categories);
829                 currentSettings->set_str(settings_pack::listen_interfaces, "0.0.0.0:"+
    UtilityFunctions::ConvertIntegerToString(
    ApplicationLogic::CurrentUserSession::currentUser->libtorrentPort));
830                 currentSettings->set_str(settings_pack::user_agent, "Dimensionless");
831
```

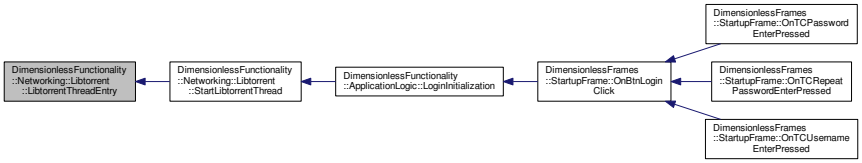
```
832                 // If you are going to be using private torrents then these settings should speed up
     execution.
833                 //currentSettings.set_bool(settings_pack::enable_lsd, false);
834                 //currentSettings.set_bool(settings_pack::enable_dht, false);
835                 // This is supposedly going to get depreciated in the next libtorrent version,
     listen_interfaces should take care of this?
836                 //currentSettings.set_int(settings_pack::ssl_listen,
     ApplicationLogic::CurrentUserSession::currentUser->libtorrentPort);
837
838                 currentSession = std::unique_ptr<libtorrent::session>(new libtorrent::session
    (*currentSettings));
839                 // libtorrent port should be automatically forwarded on session start, here's for
     everything else.
840                 UPnP::AddPortMapping(libtorrent::session_handle::udp,
    ApplicationLogic::CurrentUserSession::currentUser->webPort, ApplicationLogic::CurrentUserSession::currentUser->webPort);
841                 UPnP::AddPortMapping(libtorrent::session_handle::tcp,
    ApplicationLogic::CurrentUserSession::currentUser->webPort, ApplicationLogic::CurrentUserSession::currentUser->webPort);
842                 UPnP::AddPortMapping(libtorrent::session_handle::udp,
    ApplicationLogic::CurrentUserSession::currentUser->OpenVPNPort, ApplicationLogic::CurrentUserSession::currentUser->
    OpenVPNPort);
843                 UPnP::AddPortMapping(libtorrent::session_handle::tcp,
    ApplicationLogic::CurrentUserSession::currentUser->OpenVPNPort, ApplicationLogic::CurrentUserSession::currentUser->
    OpenVPNPort);
844
845                 if(!UtilityFunctions::CheckFileExists(
    ApplicationInfo::GetExecutablePath()+"Users/"+targetUsername+"/"+targetUsername+".torrent")){
846
    DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent
    (targetUsername,XML::ReadXMLChildValue("./Users/"+targetUsername+"/data/"+
    targetUsername+".xml","User","AESIV"),"./Users/"+targetUsername+"/CA/cacert.pem");
847                 }
848                 AddTorrent("./Users/"+targetUsername+"/"+targetUsername+".torrent","./Users/"+
    targetUsername+"/CA/cacert.pem","./Users/"+targetUsername+"/CA/private/cakey.pem","./Users/"+targetUsername+"
    /CA/dhparams.pem",targetPassword);
849
850                 while(!stopLibtorrentThread){
851                     try{
852                         std::vector<libtorrent::alert*> alerts;
853                         currentSession->pop_alerts(&alerts);
854
855                         for (libtorrent::alert const* a : alerts) {
856                             // if we receive the finished alert or an error, we're done
857                             if (libtorrent::alert_cast<libtorrent::add_torrent_alert>(a)) {
858                                 std::cout << "Torrent added - downloading now." << std::endl;
859                             }
860                             if (libtorrent::alert_cast<libtorrent::portmap_alert>(a) ||
    libtorrent::alert_cast<libtorrent::portmap_log_alert>(a) || libtorrent::alert_cast<libtorrent::portmap_error_alert>(a))
861                                 //resort to tor or openvpn, or some other NAT penetration techniques?
862                             } else {
863                                 //std::cout << a->message() << std::endl;
864                             }
865                             if (libtorrent::alert_cast<libtorrent::torrent_finished_alert>(a)) {
866                                 std::cout << "TORRENT_FINISHED: " << a->message() << std::endl;
867                             }
868                             if(auto st = libtorrent::alert_cast<libtorrent::state_update_alert>(a)) {
869                                 if (st->status.empty()) continue;
870
871                                 // The first torrent should be ours.
872                                 libtorrent::torrent_status const& s = st->status[0];
873                                 std::cout << "\r";
874                                 switch(s.state) {
875                                     case libtorrent::torrent_status::checking_files: std::cout << "checking
    "; break;
876                                     case libtorrent::torrent_status::downloading_metadata: std::cout << "dl
     metadata"; break;
877                                     case libtorrent::torrent_status::downloading: std::cout << "downloading
    "; break;
878                                     case libtorrent::torrent_status::finished: std::cout << "finished";
    break;
879                                     case libtorrent::torrent_status::seeding: std::cout << "seeding"; break
    ;
880                                     case libtorrent::torrent_status::allocating: std::cout << "allocating";
     break;
881                                     case libtorrent::torrent_status::checking_resume_data: std::cout << "
    checking resume"; break;
882                                     default: std::cout << "<>"; break;
883                                 }
884                                 std::cout << " "
885                                     << (s.download_payload_rate / 1000) << " kB/s "
886                                     << (s.total_done / 1000) << " kB ("
887                                     << (s.progress_ppm / 10000) << "%) downloaded\x1b[K";
888                                 std::cout.flush();
889                                 if(s.progress_ppm / 10000 == 100) {
890                                     std::cout << "finished!" << std::endl;
891                                     //StopLibtorrentThread();
892                                 }
893                             }
```

```
894                             if(libtorrent::alert_cast<libtorrent::torrent_alert>(a)){
895                                 std::cout << "TORRENT_ALERT: " << a->message() << std::endl;
896                             }
897                             if (libtorrent::alert_cast<libtorrent::torrent_error_alert>(a)) {
898                                 throw std::runtime_error("libtorrent: torrent error - " + a->message());
899                             }
900                             if(libtorrent::alert_cast<libtorrent::peer_alert>(a) || libtorrent::alert_cast<
    libtorrent::peer_log_alert>(a)){
901                                 std::cout << a->message() << std::endl;
902                             }
903                         }
904                         currentSession->post_torrent_updates();
905                         std::this_thread::sleep_for(std::chrono::milliseconds(1000));
906                     } catch (...) {
907                         // do nothing.
908                     }
909                 }
910                 std::cout << "Shutting down libtorrent thread." << std::endl;
911             }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.9.2.6 RemoveTorrent()

```
void DimensionlessFunctionality::Networking::Libtorrent::RemoveTorrent (
            const string & targetUsername,
            const bool & deleteFiles )
```

This function removes a torrent from those that our libtorrent thread is currently working with (if any).

Parameters

in	*targetUsername*	The username of the user whose torrent is to be removed.
in	*deleteFiles*	Decides if we should delete the files that libtorrent has downloaded for the torrent to be removed.

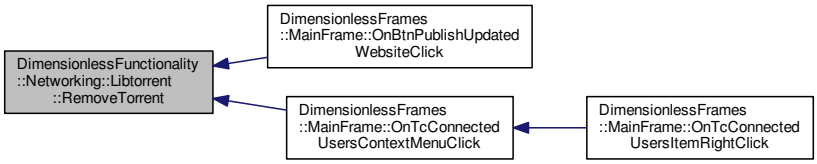
Definition at line 804 of file DimensionlessFunctionality.cpp.

References currentSession.

Referenced by DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
805             {
806                 for(libtorrent::torrent_handle& h : currentSession->get_torrents()){
807                     libtorrent::torrent_status s = h.status();
808                     if(boost::algorithm::contains(s.name, targetUsername)){
809                         if(deleteFiles){
810                             currentSession->remove_torrent(h ,
    libtorrent::session::delete_files);
811                         } else {
812                             currentSession->remove_torrent(h);
813                         }
814                         break;
815                     }
816                 }
817             }
```

Here is the caller graph for this function:



11.9.2.7 ScanForTorrentFiles()

```
vector< string > DimensionlessFunctionality::Networking::Libtorrent::ScanForTorrentFiles (
            const string & scanDirectoryRoot )
```

This function scans a directory for .torrent files and returns a string vector of paths to such files.

Parameters

in	*scanDirectoryRoot*	The root directory to scan for torrent files recursively in.

Returns

A string vector of paths to discovered torrent files.

Definition at line 661 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
662             {
663                 if(!boost::filesystem::exists(scanDirectoryRoot) || !boost::filesystem::is_directory(
    scanDirectoryRoot)) return{};
664 
665                 vector<string> ret;
666                 boost::filesystem::recursive_directory_iterator it(scanDirectoryRoot);
667                 boost::filesystem::recursive_directory_iterator endit;
668 
669                 while(it != endit)
670                 {
671                     if(boost::filesystem::is_regular_file(*it) && it->path().extension() == ".torrent") ret
    .push_back(it->path().string());
672                     ++it;
673                 }
674 
675                 return ret;
676             }
```

Here is the caller graph for this function:

11.9.2.8 StartLibtorrentThread()

```
void DimensionlessFunctionality::Networking::Libtorrent::StartLibtorrentThread (
            const string & targetUsername,
            const string & targetPassword )
```

This function starts our application's libtorrent thread.

Parameters

in	*targetUsername*	The username for our currently logged in user.
in	*targetPassword*	The password for our currently logged in user.

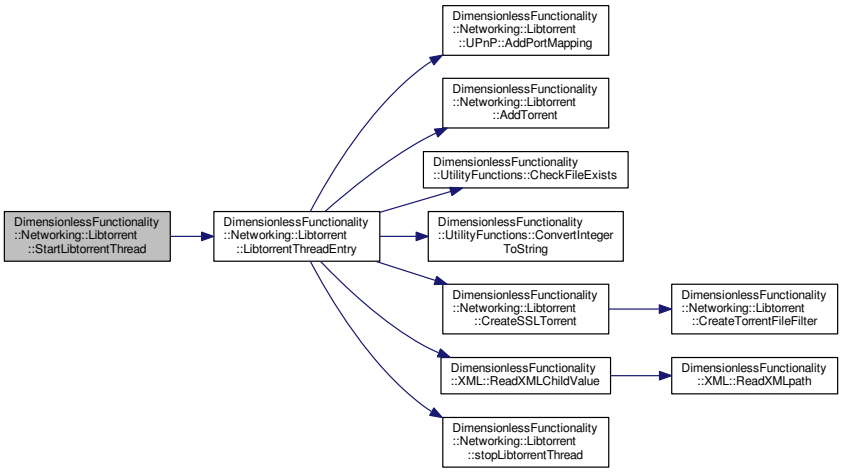
Definition at line 918 of file DimensionlessFunctionality.cpp.

References libtorrentThread, and LibtorrentThreadEntry().

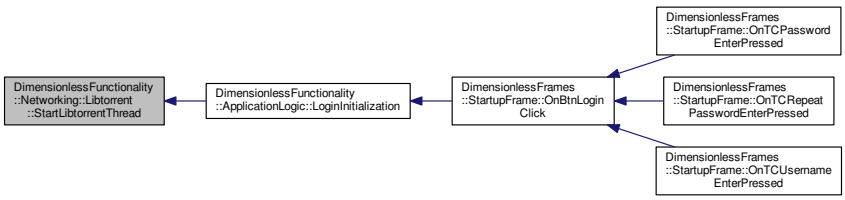
Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization().

```
919             {
920                 std::thread::id nothread;
921 
922                 if(libtorrentThread.get_id() == nothread){
923                     libtorrentThread = std::thread(
    LibtorrentThreadEntry, targetUsername, targetPassword);
924                 } else {
925                     throw std::runtime_error("Libtorrent thread already exists.");
926                 }
927             }
```

Here is the call graph for this function:



Here is the caller graph for this function:



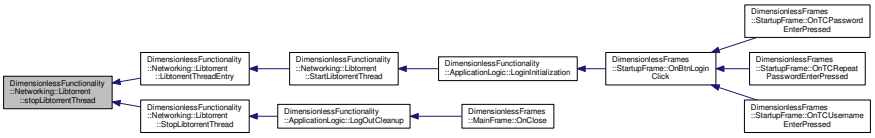
11.9.2.9 stopLibtorrentThread()

```
static std::atomic<bool> DimensionlessFunctionality::Networking::Libtorrent::stopLibtorrent↩
Thread (
            false  )  [static]
```

This atomic boolean variable decides whether it is time to stop the libtorrent thread.

Referenced by LibtorrentThreadEntry(), and StopLibtorrentThread().

Here is the caller graph for this function:



11.9.2.10 StopLibtorrentThread()

```
void DimensionlessFunctionality::Networking::Libtorrent::StopLibtorrentThread ( )
```

This function stops our application's libtorrent thread.

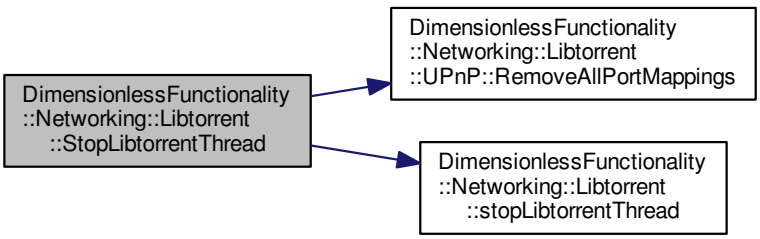
Definition at line 932 of file DimensionlessFunctionality.cpp.

References currentSession, currentSettings, libtorrentThread, DimensionlessFunctionality::Networking::↩ Libtorrent::UPnP::RemoveAllPortMappings(), and stopLibtorrentThread().

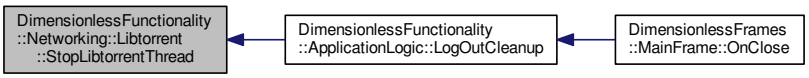
Referenced by DimensionlessFunctionality::ApplicationLogic::LogOutCleanup().

```
933             {
934                 std::thread::id nothread;
935 
936                 if(libtorrentThread.get_id() != nothread){
937                     // Remove all port mappings in UPnP and NAT-PMP first.
938                     UPnP::RemoveAllPortMappings();
939                     currentSession->abort();
940 
941                     // Stop the libtorrent thread.
942                     stopLibtorrentThread = true;
943                     libtorrentThread.join();
944                     libtorrentThread.~thread();
945 
946                     // Reset static variables to their default values.
947                     stopLibtorrentThread = false;
948                     currentSession.reset();
949                     currentSettings.reset();
950                 } else {
951                     throw std::runtime_error("No libtorrent thread exists to be stopped.");
952                 }
953             }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.9.3 Variable Documentation

11.9.3.1 currentSession

```
std::unique_ptr<session> DimensionlessFunctionality::Networking::Libtorrent::currentSession
[static]
```

This smart unique pointer stores a reference to a libtorrent session.

Definition at line 284 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::Networking::Libtorrent::UPnP::AddPortMapping(), AddTorrent(), LibtorrentThreadEntry(), DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick(), DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemoveAllPortMappings(), DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemovePortMapping(), RemoveTorrent(), and StopLibtorrentThread().

11.9.3.2 currentSettings

```
std::unique_ptr<settings_pack> DimensionlessFunctionality::Networking::Libtorrent::currentSettings [static]
```

This smart unique pointer stores a reference to our current libtorrent session's settings.

Definition at line 287 of file DimensionlessFunctionality.h.

Referenced by LibtorrentThreadEntry(), and StopLibtorrentThread().

11.9.3.3 libtorrentThread

```
std::thread DimensionlessFunctionality::Networking::Libtorrent::libtorrentThread [static]
```

This variable stores out libtorrent thread instance.

Definition at line 281 of file DimensionlessFunctionality.h.

Referenced by StartLibtorrentThread(), and StopLibtorrentThread().

11.10 DimensionlessFunctionality::Networking::Libtorrent::UPnP Namespace Reference

Functions

- bool AddPortMapping (libtorrent::session_handle::protocol_type protocolType, int externalPort, int internal↩ Port)
- bool RemovePortMapping (int index)
- bool RemoveAllPortMappings ()

Variables

- static std::vector< int > portMappings

  This integer vector stores a collection of reference to port mappings created by the AddPortMapping function.

11.10.1 Detailed Description

The UPnP (Universal Plug and Play) namespace stores all of the functions necessary to create and delete port forwarding mappings on routers that support it.

11.10.2 Function Documentation

11.10.2.1 AddPortMapping()

```
bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::AddPortMapping (
            libtorrent::session_handle::protocol_type protocolType,
            int externalPort,
            int internalPort )
```

This function adds a UPnP port mapping to any visible routers that support it. Ensure the Libtorrent thread is started first before calling this function.

Parameters

in	*protocolType*	The type of protocol the port mapping fowards (UDP or TCP).
in	*externalPort*	The external, Internet facing port that should be forwarded on the router.
in	*internalPort*	The internal, Local Area Network (LAN) facing port that should be forwarded on the router.

Returns

A boolean value indicating whether the port mapping was successfully created.

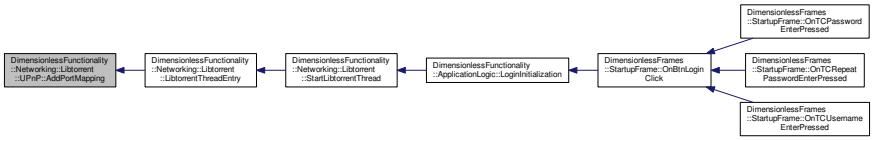
Definition at line 624 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::Networking::Libtorrent::currentSession, and portMappings.

Referenced by DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry().

```
625                 {
626                     int mapping = currentSession->add_port_mapping(protocolType, externalPort
    , internalPort);
627                     portMappings.push_back(mapping);
628                     return true;
629                 }
```

Here is the caller graph for this function:



11.10.2.2 RemoveAllPortMappings()

```
bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemoveAllPortMappings ( )
```

This function removes all UPnP port mappings from any visible routers that support it. Ensure the Libtorrent thread is started first before calling this function and that mappings have already been added by the AddPortMapping function.

Returns

A boolean value indicating whether the port mappings were successfully removed.

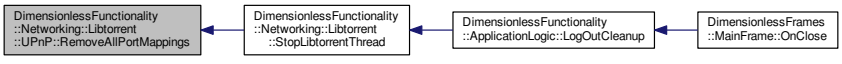
Definition at line 646 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::Networking::Libtorrent::currentSession, and portMappings.

Referenced by DimensionlessFunctionality::Networking::Libtorrent::StopLibtorrentThread().

```
647                 {
648                     for(int i : portMappings){
649                         currentSession->delete_port_mapping(i);
650                     }
651                     portMappings.clear();
652                     return true;
653                 }
```

Here is the caller graph for this function:



11.10.2.3 RemovePortMapping()

```
bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemovePortMapping (
            int index )
```

This function removes a UPnP port mapping from any visible routers that support it. Ensure the Libtorrent thread is started first before calling this function and that mappings have already been added by the AddPortMapping function.

Parameters

in	*index*	The index of the port mapping to remove.

Returns

A boolean value indicating whether the port mapping was successfully removed.

Definition at line 636 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::Networking::Libtorrent::currentSession, and portMappings.

```
637                 {
638                     currentSession->delete_port_mapping(
      portMappings[index]);
639                     return true;
640                 }
```

11.10.3 Variable Documentation

11.10.3.1 portMappings

```
std::vector<int> DimensionlessFunctionality::Networking::Libtorrent::UPnP::portMappings  [static]
```

This integer vector stores a collection of reference to port mappings created by the AddPortMapping function.

Definition at line 265 of file DimensionlessFunctionality.h.

Referenced by AddPortMapping(), RemoveAllPortMappings(), and RemovePortMapping().

11.11 DimensionlessFunctionality::Networking::Multicast Namespace Reference

Namespaces

- Receiver
- Sender

Variables

- static std::map< boost::asio::ip::address, string > localUserMap

  This map of IP addresses and strings stores a reference of the multicast users discovered on our LAN.

11.11.1 Detailed Description

The Multicast namespace stores all of the functions that discover and keep track of other LAN users of this application.

11.11.2 Variable Documentation

11.11.2.1 localUserMap

```
std::map<boost::asio::ip::address, string> DimensionlessFunctionality::Networking::Multicast↩
::localUserMap  [static]
```

This map of IP addresses and strings stores a reference of the multicast users discovered on our LAN.

Definition at line 323 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving(), DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), and DimensionlessFunctionality::Networking::Multicast::Receiver::StopReceiverThread().

11.12 DimensionlessFunctionality::Networking::Multicast::Receiver Namespace Reference

Functions

- void setupReceiving (const string &listenAddress, const string &multicastAddress)
- void handleReceiving (const boost::system::error_code &error, size_t bytesReceived)
- void receiverThreadEntry ()
- void StartReceiverThread (const string &listenAddress, const string &multicastAddress)
- void StopReceiverThread ()

Variables

- static std::thread receiverThread

  This variable stores out multicast receiver thread instance.
- static std::unique_ptr< boost::asio::io_service > receiverIOService

  This smart unique pointer stores a reference to a receiver io_service instance.
- static boost::asio::ip::address currentMulticastAddress

  This variable stores the current multicast address group that our receiver thread is listening to.
- static boost::asio::ip::udp::endpoint senderEndpoint

  This variable stores the sender of the last received multicast message.
- static std::unique_ptr< boost::asio::ip::udp::socket > receiverSocket

  This smart unique pointer stores a reference to a socket that listens for multicast messages.
- static std::vector< char > data = std::vector<char>(1024)

  This char vector stores the last received multicast message.

11.12.1 Detailed Description

The Receiver namespace stores all of the functions related to listening for multicast messages over our LAN.

11.12.2 Function Documentation

11.12.2.1 handleReceiving()

```
void DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving (
            const boost::system::error_code & error,
            size_t bytesReceived )
```

This function is called after our socket has received a multicast message.

Parameters

in	*error*	A boost system error_code that stores any information if an error occurred during message receiving.
in	*bytesReceived*	The number of bytes received from the last multicast message.

Definition at line 1048 of file DimensionlessFunctionality.cpp.

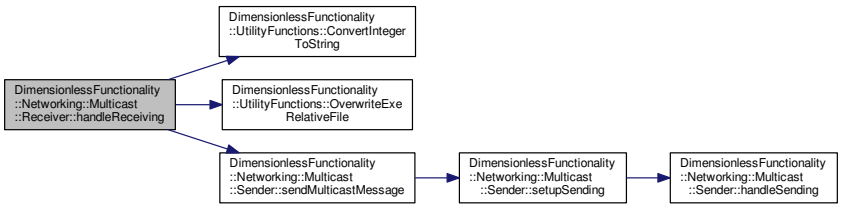
References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), currentMulticastAddress, DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::currentUser, data, DimensionlessFunctionality::Networking::Multicast::localUserMap, DimensionlessFunctionality::UtilityFunctions::OverwriteExeRelativeFile(), receiverSocket, senderEndpoint, and DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage().

Referenced by setupReceiving().

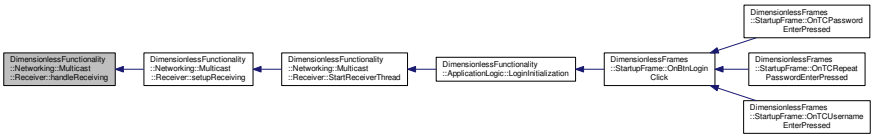
```
1049                {
1050                    if (!error) {
1051                        string receivedData(data.data());
1052                        std::cout << receivedData;
1053                        std::cout << std::endl;
1054                        std::cout << senderEndpoint.address().to_string() << std::endl;
1055                        data.clear();
1056                        data = std::vector<char>(1024);
1057
1058                        if(receivedData == "Query"){
1059                            if(ApplicationLogic::CurrentUserSession::currentUser != nullptr){
1060
    DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage
    (currentMulticastAddress.to_string(),
    ApplicationLogic::CurrentUserSession::currentUser->toMulticastString());
1061                            }
1062                        } else if(receivedData == "Offline") {
1063                            localUserMap.erase(senderEndpoint.address());
1064                        } else {
1065                            localUserMap[senderEndpoint.address()] = receivedData
    ;
1066                        }
1067
1068                        string localusers = "<html><title>";
1069                        std::vector<string> temp;
1070                        temp.push_back(UtilityFunctions::ConvertIntegerToString
    (localUserMap.size()));
1071                        for(auto const& item : localUserMap){
1072                            temp.push_back(
    UtilityFunctions::ConvertIntegerToString(item.second.length()));
1073                        }
1074                        for(auto const& item : localUserMap){
1075                            temp.push_back(item.second);
1076                        }
1077                        localusers += boost::algorithm::join(temp,",");
1078                        localusers += "</title></html>";
1079
1080                        UtilityFunctions::OverwriteExeRelativeFile
```

```
      ("Users/"+ApplicationLogic::CurrentUserSession::currentUser->username+"/data/user.html", localusers);
1081
1082                         receiverSocket->async_receive_from(
1083                             boost::asio::buffer(data, 1024), senderEndpoint,
1084                             boost::bind(&handleReceiving,
1085                                 boost::asio::placeholders::error,
1086                                 boost::asio::placeholders::bytes_transferred));
1087                     } else {
1088                         throw std::runtime_error("Multicast receiver thread has encountered an error.");
1089                     }
1090                 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.12.2.2 receiverThreadEntry()

`void DimensionlessFunctionality::Networking::Multicast::Receiver::receiverThreadEntry ( )`

This function is called from our multicast receiver thread to start listening to messages.

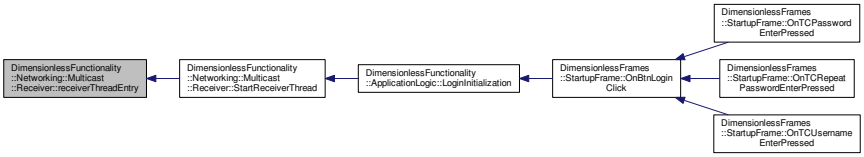
Definition at line 1095 of file DimensionlessFunctionality.cpp.

References receiverIOService.

Referenced by StartReceiverThread().

```
1096                 {
1097                     receiverIOService->run();
1098                 }
```

Here is the caller graph for this function:



11.12.2.3 setupReceiving()

```
void DimensionlessFunctionality::Networking::Multicast::Receiver::setupReceiving (
            const string & listenAddress,
            const string & multicastAddress )
```

This function sets up our multicast message recieving socket.

Parameters

in	*listenAddress*	An address on which our socket is to start listening on (typically localhost).
in	*multicastAddress*	The address of the multicast group to listen to.

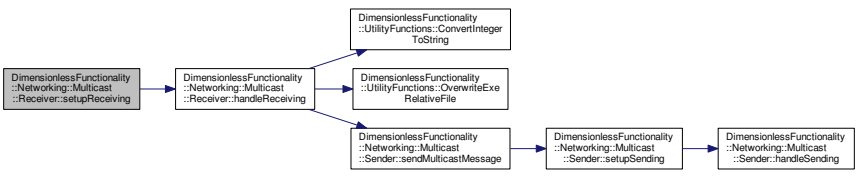
Definition at line 1020 of file DimensionlessFunctionality.cpp.

References currentMulticastAddress, data, handleReceiving(), receiverSocket, and senderEndpoint.

Referenced by StartReceiverThread().

```
1021                {
1022                    try{
1023                        currentMulticastAddress =
      boost::asio::ip::address::from_string(multicastAddress);
1024                        boost::asio::ip::udp::endpoint listen_endpoint(
      boost::asio::ip::address::from_string(listenAddress), 30001);
1025                        if(receiverSocket->is_open()){
1026                            receiverSocket->close();
1027                        }
1028                        receiverSocket->open(listen_endpoint.protocol());
1029                        receiverSocket->set_option(
      boost::asio::ip::udp::socket::reuse_address(true));
1030                        receiverSocket->bind(listen_endpoint);
1031
1032                        receiverSocket->set_option(boost::asio::ip::multicast::join_group(
      boost::asio::ip::address::from_string(multicastAddress)));
1033                        receiverSocket->async_receive_from(
1034                            boost::asio::buffer(data, 1024), senderEndpoint,
1035                            boost::bind(&handleReceiving,
1036                                boost::asio::placeholders::error,
1037                                boost::asio::placeholders::bytes_transferred));
1038                    } catch (...) {
1039                        throw std::runtime_error("Multicast receiver thread has encountered an error.");
1040                    }
1041                }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.12.2.4 StartReceiverThread()

```
void DimensionlessFunctionality::Networking::Multicast::Receiver::StartReceiverThread (
            const string & listenAddress,
            const string & multicastAddress )
```

This function is called to start our receiver thread for multicast messages.

Parameters

in	*listenAddress*	An address on which our socket is to start listening on (typically localhost).
in	*multicastAddress*	The address of the multicast group to listen to.

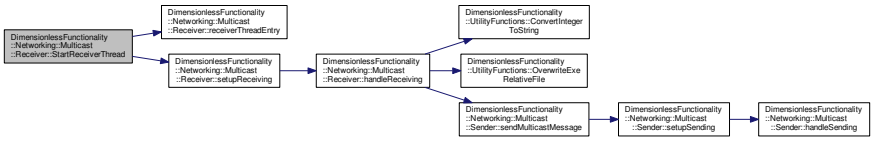
Definition at line 1105 of file DimensionlessFunctionality.cpp.

References receiverIOService, receiverSocket, receiverThread, receiverThreadEntry(), and setupReceiving().

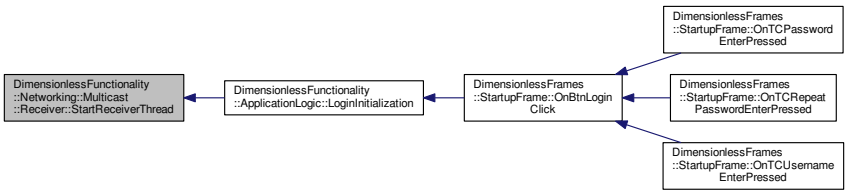
Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization().

```
1106                 {
1107                     std::thread::id nothread;
1108 
1109                     if(receiverThread.get_id() == nothread){
1110                         receiverIOService = std::unique_ptr<boost::asio::io_service>(new
      boost::asio::io_service());
1111                         receiverSocket = std::unique_ptr<boost::asio::ip::udp::socket>(new
      boost::asio::ip::udp::socket(*receiverIOService));
1112                         setupReceiving(listenAddress, multicastAddress);
1113                         receiverThread = std::thread(
      receiverThreadEntry);
1114                     } else {
1115                         throw std::runtime_error("Multicast receiver thread already exists.");
1116                     }
1117                 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.12.2.5 StopReceiverThread()

```
void DimensionlessFunctionality::Networking::Multicast::Receiver::StopReceiverThread ( )
```

This function is called to stop our receiver thread that is listening to multicast messages.

Definition at line 1122 of file DimensionlessFunctionality.cpp.

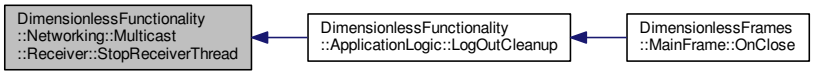
References currentMulticastAddress, DimensionlessFunctionality::Networking::Multicast::localUserMap, receiverIOService, receiverSocket, receiverThread, and DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage().

Referenced by DimensionlessFunctionality::ApplicationLogic::LogOutCleanup().

```
1123                 {
1124                     std::thread::id nothread;
1125 
1126                     if(receiverThread.get_id() != nothread){
1127                         receiverIOService->stop();
1128                         receiverSocket->close();
1129                         receiverSocket.reset();
1130                         for(int i = 0; i < 3; i++){
1131                             Sender::sendMulticastMessage(
    currentMulticastAddress.to_string(), "Offline");
1132                         }
1133                         localUserMap.clear();
1134                         receiverThread.join();
1135                         receiverThread.~thread();
1136                         receiverIOService.reset();
1137                     } else {
1138                         throw std::runtime_error("No multicast receiver thread exists to be stopped.");
1139                     }
1140                 }
```

Here is the call graph for this function:

Here is the caller graph for this function:



11.12.3 Variable Documentation

11.12.3.1 currentMulticastAddress

```
boost::asio::ip::address DimensionlessFunctionality::Networking::Multicast::Receiver::current↩
MulticastAddress  [static]
```

This variable stores the current multicast address group that our receiver thread is listening to.

Definition at line 352 of file DimensionlessFunctionality.h.

Referenced by handleReceiving(), setupReceiving(), and StopReceiverThread().

11.12.3.2 data

```
std::vector<char> DimensionlessFunctionality::Networking::Multicast::Receiver::data = std↩
::vector<char>(1024)  [static]
```

This char vector stores the last received multicast message.

Definition at line 361 of file DimensionlessFunctionality.h.

Referenced by handleReceiving(), DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest(), DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest(), setupReceiving(), and DimensionlessFunctionality::Validation::validate< string, string >().

11.12.3.3 receiverIOService

```
std::unique_ptr<boost::asio::io_service> DimensionlessFunctionality::Networking::Multicast::↩
Receiver::receiverIOService  [static]
```

This smart unique pointer stores a reference to a receiver io_service instance.

Definition at line 349 of file DimensionlessFunctionality.h.

Referenced by receiverThreadEntry(), StartReceiverThread(), and StopReceiverThread().

11.12.3.4 receiverSocket

```
std::unique_ptr<boost::asio::ip::udp::socket> DimensionlessFunctionality::Networking::Multicast↩
::Receiver::receiverSocket  [static]
```

This smart unique pointer stores a reference to a socket that listens for multicast messages.

Definition at line 358 of file DimensionlessFunctionality.h.

Referenced by handleReceiving(), setupReceiving(), StartReceiverThread(), and StopReceiverThread().

11.12.3.5 receiverThread

```
std::thread DimensionlessFunctionality::Networking::Multicast::Receiver::receiverThread  [static]
```

This variable stores out multicast receiver thread instance.

Definition at line 346 of file DimensionlessFunctionality.h.

Referenced by StartReceiverThread(), and StopReceiverThread().

11.12.3.6 senderEndpoint

```
boost::asio::ip::udp::endpoint DimensionlessFunctionality::Networking::Multicast::Receiver↩
::senderEndpoint  [static]
```

This variable stores the sender of the last received multicast message.

Definition at line 355 of file DimensionlessFunctionality.h.

Referenced by handleReceiving(), and setupReceiving().

11.13 DimensionlessFunctionality::Networking::Multicast::Sender Namespace Reference

Functions

- void setupSending (boost::asio::io_service &senderIOService, const string &multicastAddress, const string &message)
- void handleSending (const boost::system::error_code &error)
- void sendMulticastMessage (const string &multicastAddress, const string &message)

11.13.1 Detailed Description

The Sender namespace stores all of the functions related to sending multicast messages over our LAN.

11.13.2 Function Documentation

11.13.2.1 handleSending()

```
void DimensionlessFunctionality::Networking::Multicast::Sender::handleSending (
            const boost::system::error_code & error )
```

This function is called after our socket has sent a multicast message.

Parameters

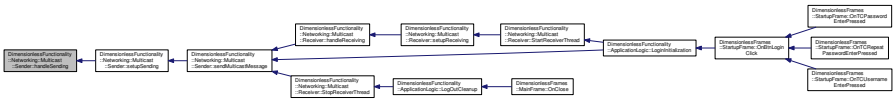
in	*error*	A boost system error_code that stores any information if an error occurred during message sending.

Definition at line 990 of file DimensionlessFunctionality.cpp.

Referenced by setupSending().

```
991                 {
992                     if (error) {
993                         throw std::runtime_error("Multicast sender thread has encountered an error.");
994                     }
995                 }
```

Here is the caller graph for this function:



11.13.2.2 sendMulticastMessage()

```
void DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage (
            const string & multicastAddress,
            const string & message )
```

This function is called by our application to send a multicast message.

Parameters

in	*multicastAddress*	The multicast address to send a multicast message to.
in	*message*	The message to send over multicast.

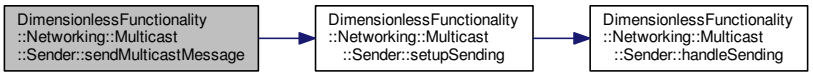
Definition at line 1002 of file DimensionlessFunctionality.cpp.

References setupSending().

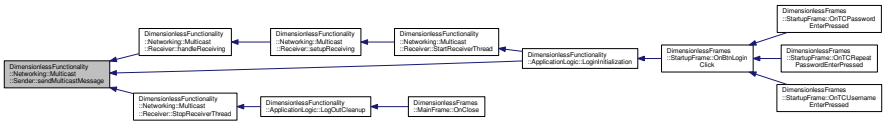
Referenced by DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving(), DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), and DimensionlessFunctionality::Networking::Multicast::Receiver::StopReceiverThread().

```
1003                 {
1004                     boost::asio::io_service senderIOService;
1005                     setupSending(senderIOService, multicastAddress, message);
1006                     senderIOService.run();
1007                 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.13.2.3 setupSending()

```
void DimensionlessFunctionality::Networking::Multicast::Sender::setupSending (
            boost::asio::io_service & senderIOService,
            const string & multicastAddress,
            const string & message )
```

This function sets up and performs the multicast sending operation.

Parameters

in	*senderIOService*	A boost asio io_service instance for our multicast sending socket.
in	*multicastAddress*	The multicast address to send a multicast message to.
in	*message*	The message to send over multicast.

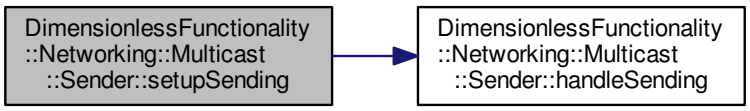
Definition at line 972 of file DimensionlessFunctionality.cpp.

References handleSending().

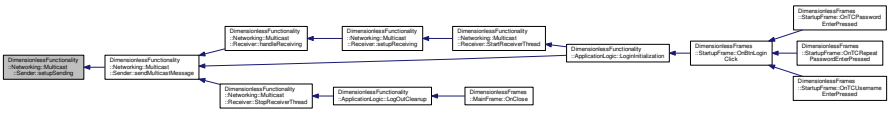
Referenced by sendMulticastMessage().

```
973                 {
974                     try {
975                         boost::asio::ip::udp::endpoint send_endpoint(boost::asio::ip::address::from_string(
      multicastAddress), 30001);
976                         boost::asio::ip::udp::socket senderSocket(senderIOService, send_endpoint.protocol()
      );
977                         senderSocket.async_send_to(boost::asio::buffer(message), send_endpoint,
978                                                    boost::bind(&Sender::handleSending,
979                                                    boost::asio::placeholders::error));
980                         //std::this_thread::sleep_for(std::chrono::seconds(1));
981                     } catch (...) {
982                         throw std::runtime_error("Multicast sender thread has encountered an error.");
983                     }
984                 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.14 DimensionlessFunctionality::Networking::Nmap Namespace Reference

Functions

- vector< string > ScanIPAddressesWithPort (const string &username, const vector< string > &IPv4↩ Addresses, const int &port)

11.14.1 Detailed Description

The Nmap namespace stores all of the functions necessary to scan IP addresses.

11.14.2 Function Documentation

11.14.2.1 ScanIPAddressesWithPort()

```
vector< string > DimensionlessFunctionality::Networking::Nmap::ScanIPAddressesWithPort (
            const string & username,
            const vector< string > & IPv4Addresses,
            const int & port )
```

This function scans a vector of IP addresses at a certain port and for a title that contains a specific username.

Parameters

in	*username*	The username to search for in the HTTP title of the IP addresses scanned.
in	*IPv4Addresses*	The collection of IPv4 addresses to scan.
in	*port*	The port number to scan for each IPv4 address provided.

Returns

A string vector of potential candidates for the username specified.

Definition at line 1156 of file DimensionlessFunctionality.cpp.

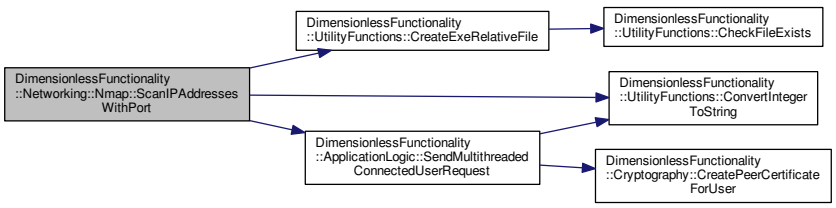
References DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::connectedUsers, DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile(), DimensionlessFunctionality::ApplicationLogic::User::desktop, DimensionlessFunctionality::ApplicationLogic::User::mobile, DimensionlessFunctionality::ApplicationLogic::User::online, DimensionlessFunctionality::ApplicationLogic::PerformSearch, and DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest().

Referenced by DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUsers().

```
1157            {
1158                std::vector<string> potentialUsers;
1159
1160                unsigned int onlineUserCount = 0;
1161                unsigned int mobileUserCount = 0;
1162
1163                for(std::shared_ptr<ApplicationLogic::ConnectedUser>& user :
    ApplicationLogic::CurrentUserSession::connectedUsers){
1164                    if(user->currentStatus == ApplicationLogic::ConnectedUser::Status::online){
1165                        onlineUserCount++;
1166                    }
1167                    if(user->currentPlatform == ApplicationLogic::ConnectedUser::Platform::mobile){
1168                        mobileUserCount++;
1169                    }
1170                }
1171
1172                std::vector<std::thread> threadpool;
1173                std::atomic<unsigned int> activeRequestHandlingUsers(0);
1174                unsigned int currentTotalAddresses = IPv4Addresses.size();
1175
1176                if(onlineUserCount > 0){
1177                    //unsigned int leftOverAddresses = IPv4Addresses.size() % onlineUserCount;
1178                    unsigned int addressesPerOnlineUsers = IPv4Addresses.size() / onlineUserCount;
1179
1180                    // distribute searchable IPv4 addresses across friends.
1181                    for(std::shared_ptr<ApplicationLogic::ConnectedUser>& user :
    ApplicationLogic::CurrentUserSession::connectedUsers){
1182                        if(user->currentStatus == ApplicationLogic::ConnectedUser::Status::online){
1183                            switch(user->currentPlatform){
1184                                case ApplicationLogic::ConnectedUser::Platform::desktop:
1185                                    {
1186                                        // give 1 weight
1187                                        vector<string>::const_iterator first = IPv4Addresses.begin() + (
    IPv4Addresses.size() - currentTotalAddresses);
1188                                        vector<string>::const_iterator last = IPv4Addresses.begin() + (
    IPv4Addresses.size() - currentTotalAddresses) + addressesPerOnlineUsers;
1189                                        vector<string> temp(first, last);
1190                                        string ips = boost::algorithm::join(temp,",");
1191                                        activeRequestHandlingUsers++;
1192                                        threadpool.push_back(std::thread(
    ApplicationLogic::SendMultithreadedConnectedUserRequest
    , std::ref(activeRequestHandlingUsers), ApplicationLogic::Request::PerformSearch, std::ref(user), std::ref(
    ips)));
1193                                        //threadpool.end().detach();
1194                                        currentTotalAddresses -= addressesPerOnlineUsers;
1195                                        break;
1196                                    }
1197                                case ApplicationLogic::ConnectedUser::Platform::mobile:
1198                                    {
1199                                        // given 1/2 weight up to limit
1200                                        //expect ~1000+ bytes per scan so some kbs per scan but perhaps
     limit to 64 scannable ip ranges.
1201                                        vector<string>::const_iterator first = IPv4Addresses.begin() + (
    IPv4Addresses.size() - currentTotalAddresses);
1202                                        vector<string>::const_iterator last = IPv4Addresses.begin() + (
    IPv4Addresses.size() - currentTotalAddresses) + addressesPerOnlineUsers/2;
1203                                        vector<string> temp(first, last);
1204                                        string ips = boost::algorithm::join(temp,",");
1205                                        activeRequestHandlingUsers++;
1206                                        threadpool.push_back(std::thread(
    ApplicationLogic::SendMultithreadedConnectedUserRequest
    , std::ref(activeRequestHandlingUsers), ApplicationLogic::Request::PerformSearch, std::ref(user), std::ref(
    ips)));
1207                                        //threadpool.end().detach();
```

```
1208                                        currentTotalAddresses -= addressesPerOnlineUsers/2;
1209                                        break;
1210                                    }
1211                                default:
1212                                    {
1213                                        throw std::runtime_error("Error! Invalid platform for a connected
 user.");
1214                                        break;
1215                                    }
1216                            }
1217                        }
1218                    }
1219                }
1220
1221                while(activeRequestHandlingUsers > 0){
1222                    // wait.
1223                }
1224
1225                for(unsigned int i = 0; i < threadpool.size(); i++){
1226                    threadpool[i].~thread();
1227                }
1228
1229                vector<string>::const_iterator first = IPv4Addresses.begin() + (IPv4Addresses.size() -
currentTotalAddresses);
1230                vector<string>::const_iterator last = IPv4Addresses.end();
1231                vector<string> temp(first, last);
1232                // only search left ips and make sure to wait for all previous requests to leave a response
1233                UtilityFunctions::CreateExeRelativeFile("ips.txt",
boost::algorithm::join(temp,"\n"));
1234                string cmd = "./Executables/nmap/nmap -Pn -T4 -n --defeat-rst-ratelimit --open -p "+
UtilityFunctions::ConvertIntegerToString(port)+" --min-parallelism
 100 --script=+http-title --script-args='url=/user.html' -iL ips.txt > search-result.txt";
1235                std::system(cmd.c_str());
1236                boost::filesystem::remove("./ips.txt");
1237
1238                std::deque<string> previousLines;
1239                std::ifstream resultfile("./search-result.txt");
1240                for (std::string line; std::getline(resultfile, line);)
1241                {
1242                    if(boost::algorithm::contains(line, "http-title") && boost::algorithm::contains(line,
username)){
1243                        if(previousLines.size() == 5){
1244                            if(boost::algorithm::contains(previousLines.back(),"(") &&
boost::algorithm::contains(previousLines.back(),")") ){
1245                                std::size_t leftbracket = previousLines.back().find_last_of("(");
1246                                std::size_t rightbracket = previousLines.back().find_last_of(")");
1247                                string IPv4Address = previousLines.back().substr(leftbracket+1,rightbracket
-leftbracket-1);
1248                                potentialUsers.push_back(IPv4Address);
1249                                potentialUsers.push_back(line.substr(14));
1250                            } else if(boost::algorithm::contains(previousLines.back()," ")) {
1251                                std::size_t lastspace = previousLines.back().find_last_of(" ");
1252                                string IPv4Address = previousLines.back().substr(lastspace+1,previousLines.
back().size()-lastspace);
1253                                potentialUsers.push_back(IPv4Address);
1254                                potentialUsers.push_back(line.substr(14));
1255                            }
1256                        }
1257                    }
1258                    previousLines.push_front(line);
1259                    if(previousLines.size() > 5){
1260                        previousLines.pop_back();
1261                    }
1262                }
1263                resultfile.close();
1264                boost::filesystem::remove("./search-result.txt");
1265                return potentialUsers;
1266            }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.15 DimensionlessFunctionality::Networking::OpenVPN Namespace Reference

Functions

- void StartOpenVPNClient (const string &vpnConfigurationDirectory)
- void StartOpenVPNServer (const string &vpnConfigurationDirectory)
- void StopOpenVPNClient (const string &vpnConfigurationDirectory)
- void StopOpenVPNServer (const string &vpnConfigurationDirectory)

11.15.1 Detailed Description

The OpenVPN namespace stores all of the functions necessary to start a VPN server or connect to other users' LAN and circumvent any port forwarding problems.

11.15.2 Function Documentation

11.15.2.1 StartOpenVPNClient()

```
void DimensionlessFunctionality::Networking::OpenVPN::StartOpenVPNClient (
            const string & vpnConfigurationDirectory )
```

This function starts our OpenVPN client instance.

Parameters

in	*vpnConfigurationDirectory*	The path to an OpenVPN client configuration file.

Definition at line 1278 of file DimensionlessFunctionality.cpp.

```
1279             {
1280                 string cmd = "";
1281                 std::system(cmd.c_str());
1282             }
```

11.15.2.2 StartOpenVPNServer()

```
void DimensionlessFunctionality::Networking::OpenVPN::StartOpenVPNServer (
             const string & vpnConfigurationDirectory )
```

This function starts our OpenVPN server instance.

Parameters

in	*vpnConfigurationDirectory*	The path to an OpenVPN server configuration file.

Definition at line 1288 of file DimensionlessFunctionality.cpp.

```
1289             {
1290                 string cmd = "";
1291                 std::system(cmd.c_str());
1292             }
```

11.15.2.3 StopOpenVPNClient()

```
void DimensionlessFunctionality::Networking::OpenVPN::StopOpenVPNClient (
             const string & vpnConfigurationDirectory )
```

This function stops our OpenVPN client instance.

Parameters

in	*vpnConfigurationDirectory*	The path to an OpenVPN client configuration file.

Definition at line 1298 of file DimensionlessFunctionality.cpp.

```
1299             {
1300                 string cmd = "";
1301                 std::system(cmd.c_str());
1302             }
```

11.15.2.4 StopOpenVPNServer()

```
void DimensionlessFunctionality::Networking::OpenVPN::StopOpenVPNServer (
            const string & vpnConfigurationDirectory )
```

This function stops our OpenVPN server instance.

Parameters

in	*vpnConfigurationDirectory*	The path to an OpenVPN server configuration file.

Definition at line 1308 of file DimensionlessFunctionality.cpp.

```
1309             {
1310                 string cmd = "";
1311                 std::system(cmd.c_str());
1312             }
```

11.16 DimensionlessFunctionality::Networking::PHPFPM Namespace Reference

Functions

- void StartPHPFPM (const string &webDirectory)
- void StopPHPFPM (const string &webDirectory)

11.16.1 Detailed Description

The PHPFPM namespace stores all of the functions that control php-fpm as the backend to our web server.

11.16.2 Function Documentation

11.16.2.1 StartPHPFPM()

```
void DimensionlessFunctionality::Networking::PHPFPM::StartPHPFPM (
            const string & webDirectory )
```

This function starts php-fpm.

Parameters

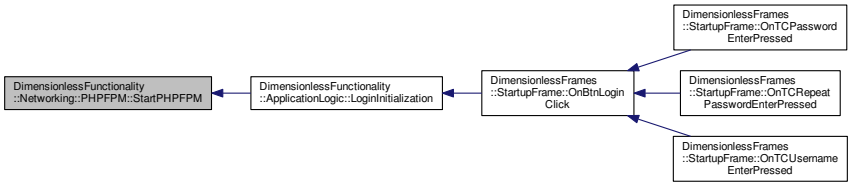
in	*webDirectory*	The directory where php-fpm configuration files are stored.

Definition at line 1324 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization().

```
1325            {
1326                #if defined(LINUX) ^ defined(MAC)
1327                    string cmd = "./Executables/php-fpm -p ./"+webDirectory+" -c ./"+webDirectory+"/php.ini
      -y ./"+webDirectory+"/php-fpm.conf";
1328                    std::system(cmd.c_str());
1329                #elif defined(WINDOWS)
1330                    string cmd = "./Executables/php-fpm.exe -p ./"+webDirectory+" -c ./"+webDirectory+"
     /php.ini -y ./"+webDirectory+"/php-fpm.conf";
1331                    std::system(cmd.c_str());
1332                #else
1333                    #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
      preprocessors. Otherwise we cannot continue compilation."
1334                #endif
1335            }
```

Here is the caller graph for this function:



11.16.2.2 StopPHPFPM()

```
void DimensionlessFunctionality::Networking::PHPFPM::StopPHPFPM (
            const string & webDirectory )
```

This function stops php-fpm.

Parameters

in	*webDirectory*	The directory where php-fpm configuration files are stored.

Definition at line 1341 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LogOutCleanup().

```
1342            {
1343                #if defined(LINUX) ^ defined(MAC)
1344                    std::ifstream file("./"+webDirectory+"/php5-fpm.pid");
1345                    std::stringstream buffer;
1346                    buffer << file.rdbuf();
1347                    file.close();
1348                    std::string str = buffer.str();
1349                    string cmd = "kill -15 "+str;
1350                    std::system(cmd.c_str());
1351                #elif defined(WINDOWS)
1352                    // Missing.
1353                #else
1354                    #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
      preprocessors. Otherwise we cannot continue compilation."
1355                #endif
1356            }
```

Here is the caller graph for this function:

11.17 DimensionlessFunctionality::Networking::Tor Namespace Reference

Functions

- void StartTor (const string &webDirectory)
- void StopTor (const string &webDirectory)

11.17.1 Detailed Description

The Tor namespace stores all of the functions that control Tor.

11.17.2 Function Documentation

11.17.2.1 StartTor()

```
void DimensionlessFunctionality::Networking::Tor::StartTor (
            const string & webDirectory )
```

This function starts Tor.

Parameters

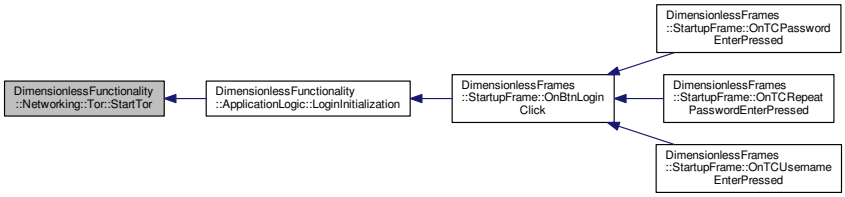
in	*webDirectory*	The directory where Tor configuration files are stored.

Definition at line 1368 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization().

```
1369             {
1370                 #if defined(LINUX) ^ defined(MAC)
1371                     string cmd = "./Executables/tor -f ./"+webDirectory+"/torrc";
1372                     std::system(cmd.c_str());
1373                 #elif defined(WINDOWS)
1374                     string cmd = "./Executables/tor.exe -f ./"+webDirectory+"/torrc";
1375                     std::system(cmd.c_str());
1376                 #else
1377                     #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
       preprocessors. Otherwise we cannot continue compilation."
1378                 #endif
1379             }
```

Here is the caller graph for this function:



11.17.2.2 StopTor()

```
void DimensionlessFunctionality::Networking::Tor::StopTor (
            const string & webDirectory )
```

This function stops Tor.

Parameters

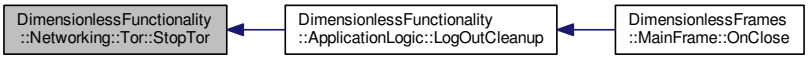
in	*webDirectory*	The directory where Tor configuration files are stored.

Definition at line 1385 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LogOutCleanup().

```
1386             {
1387                 #if defined(LINUX) ^ defined(MAC)
1388                     string cmd = "kill -15 $(pgrep --exact tor)";
1389                     std::system(cmd.c_str());
1390                 #elif defined(WINDOWS)
1391                     string cmd = "Taskkill /IM tor.exe /F";
1392                     std::system(cmd.c_str());
1393                 #else
1394                     #error "Compilation Error: Please declare either LINUX, ANDROID, MAC or WINDOWS in your
     preprocessors. Otherwise we cannot continue compilation."
1395                 #endif
1396             }
```

Here is the caller graph for this function:



11.18 DimensionlessFunctionality::Networking::WebBrowser Namespace Reference

Functions

- void StartWebBrowser (const string &url)
- void StopWebBrowser ()

11.18.1 Detailed Description

The WebBrowser namespace stores all of the functions to control our web browser (GNU IceCat or System Default).

11.18.2 Function Documentation

11.18.2.1 StartWebBrowser()

```
void DimensionlessFunctionality::Networking::WebBrowser::StartWebBrowser (
            const string & url )
```

This function starts our web browser.

Parameters

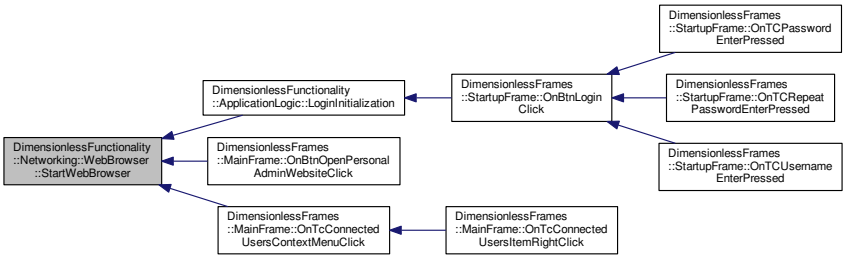
in	*url*	The URL our web browser loads when started up.

Definition at line 1408 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), DimensionlessFrames::MainFrame::OnBtnOpenPersonalAdminWebsiteClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
1409             {
1410                 #if defined(LINUX) ^ defined(MAC)
1411                     boost::filesystem::create_directory("./Executables/icecat/temp-profile");
1412                     // CHANGE THE URL HERE TO HTTPS IF WE ARE FORCING OUR SSL CERTIFICATE ON NGINX.
1413                     string cmd = "./Executables/icecat/icecat -profile ./Executables/icecat/temp-profile
    -no-remote -new-instance --private-window "+url+" &";
1414                     std::system(cmd.c_str());
1415                 #elif defined(WINDOWS)
1416                     //TODO /start or something
1417                     //std::system("./Executables/cef/cefclient --no-sandbox
    --url=http://localhost:1337/admin &");
1418                 #else
1419                     #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
    preprocessors. Otherwise we cannot continue compilation."
1420                 #endif
1421             }
```

Here is the caller graph for this function:



11.18.2.2 StopWebBrowser()

```
void DimensionlessFunctionality::Networking::WebBrowser::StopWebBrowser ( )
```

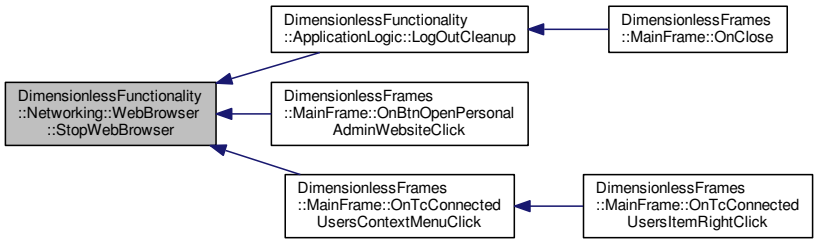
This function stops our web browser.

Definition at line 1426 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LogOutCleanup(), DimensionlessFrames::MainFrame::OnBtnOpenPersonalAdminWebsiteClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
1427             {
1428                 #if defined(LINUX) ^ defined(MAC)
1429                     boost::filesystem::remove_all("./Executables/icecat/temp-profile");
1430                     std::system("killall icecat");
1431                 #elif defined(WINDOWS)
1432                     //todo
1433                 #else
1434                     #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
       preprocessors. Otherwise we cannot continue compilation."
1435                 #endif
1436             }
```

Here is the caller graph for this function:



11.19 DimensionlessFunctionality::Networking::WebServer Namespace Reference

Functions

- void StartWebServer (const string &webDirectory, const string &password)
- void StopWebServer (const string &webDirectory)

11.19.1 Detailed Description

The WebServer namespace stores all of the functions to control our web server (Nginx).

11.19.2 Function Documentation

11.19.2.1 StartWebServer()

```
void DimensionlessFunctionality::Networking::WebServer::StartWebServer (
            const string & webDirectory,
            const string & password )
```

This function starts our web server.

Parameters

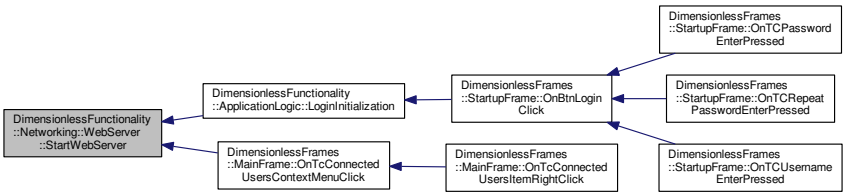
in	*webDirectory*	The directory where the website files for hosting are located.
in	*password*	If we want our web server to server our website with SSL, the password to the private key is required.

Definition at line 1449 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), and DimensionlessFrames::↩ MainFrame::OnTcConnectedUsersContextMenuClick().

```
1450             {
1451                 //WebServerProcess = new bp::child("nginx -c .", bp::std_out > WebServerBuffer,
     WebServerIOService);
1452                 //WebServerIOService.run();
1453                 //std::system("speedtest");
1454                 /* ONLY STORE OUR HASHED PASSWORD (TEMPORARILY - FOR THE RUNNING DURATION OF THE
     APPLICATION)
1455                  * IF WE WANT SSL ON OUR SITE - WILL STOP SCANS FROM WORKING AND MAY SHOW AN ANNOYING SSL
     ERROR
1456                  * MESSAGE ON BROWSERS THAT ARE NOT NATIVELY BUNDLED WITH THIS APPLICATION.
1457                  * YOU HAVE BEEN WARNED! (MUST ALSO UNCOMMENT SOME LINES IN StopWebServer AND THE SSL LINES
     IN nginx.conf)
1458                  */
1459                 /*std::ofstream outFile (ApplicationInfo::GetExecutablePath()+webDirectory+"/temp");
1460                 outFile << password;
1461                 outFile.close();*/
1462                 #if defined(LINUX) ^ defined(MAC)
1463                     string cmd = "./Executables/nginx -p ./"+webDirectory+" -c nginx.conf";
1464                     std::system(cmd.c_str());
1465                 #elif defined(WINDOWS)
1466                     string cmd = "./Executables/nginx.exe -p ./"+webDirectory+" -c nginx.conf";
1467                     std::system(cmd.c_str());
1468                 #else
1469                     #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
     preprocessors. Otherwise we cannot continue compilation."
1470                 #endif
1471             }
```

Here is the caller graph for this function:



11.19.2.2 StopWebServer()

```
void DimensionlessFunctionality::Networking::WebServer::StopWebServer (
            const string & webDirectory )
```

This function stops our web server.

Parameters

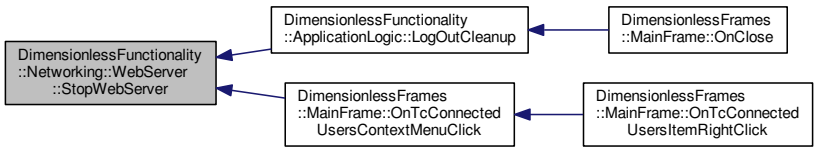
in	*webDirectory*	The directory where the website files for hosting are located.

Definition at line 1477 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LogOutCleanup(), and DimensionlessFrames::↩ MainFrame::OnTcConnectedUsersContextMenuClick().

```
1478             {
1479                 //WebServerProcess = new bp::child("nginx -c .", bp::std_out > WebServerBuffer,
     WebServerIOService);
1480                 //WebServerIOService.run();
1481                 #if defined(LINUX) ^ defined(MAC)
1482                     string cmd = "./Executables/nginx -p ./"+webDirectory+" -c nginx.conf -s stop";
1483                     std::system(cmd.c_str());
1484                     //string rmcmd = "rm -f ./"+webDirectory+"/temp";
1485                     //std::system(rmcmd.c_str());
1486                 #elif defined(WINDOWS)
1487                     string cmd = "./Executables/nginx.exe -p ./"+webDirectory+" -c nginx.conf -s stop";
1488                     std::system(cmd.c_str());
1489                     //string rmcmd = "rm -f ./"+webDirectory+"/temp";
1490                     //std::system(rmcmd.c_str());
1491                 #else
1492                     #error "Compilation Error: Please declare either LINUX, MAC or WINDOWS in your
     preprocessors. Otherwise we cannot continue compilation."
1493                 #endif
1494             }
```

Here is the caller graph for this function:



11.20 DimensionlessFunctionality::UtilityFunctions Namespace Reference

Functions

- string ConvertIntegerToString (int integerToConvert)
- bool CheckFileExists (const string &absoluteFilePath)
- bool CheckFolderExists (const string &absoluteFolderPath)
- bool CreateExeRelativeFile (const string &exeRelativeFilePath)
- bool CreateExeRelativeFile (const string &exeRelativeFilePath, const string &fileContent)
- bool OverwriteExeRelativeFile (const string &exeRelativeFilePath, const string &fileContent)
- bool CreateExeRelativeFolder (const string &exeRelativeFolderPath)
- bool CheckExeRelativeFolderPopulated (const string &exeRelativeFolderPath)

11.20.1 Detailed Description

The UtilityFunctions namespace stores all of the functions that perform operations required within the application and are not necessarily provided by C++.

11.20.2 Function Documentation

11.20.2.1 CheckExeRelativeFolderPopulated()

```
bool DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated (
            const string & exeRelativeFolderPath )
```

This function checks that a folder at a relative path is populated with files and or folders.

Parameters

in	*exeRelativeFilePath*	A string that represents the relative path of a folder to check.

Returns

A boolean denoting if the folder is populated.

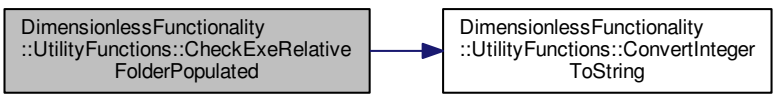
Definition at line 144 of file DimensionlessFunctionality.cpp.

References ConvertIntegerToString().

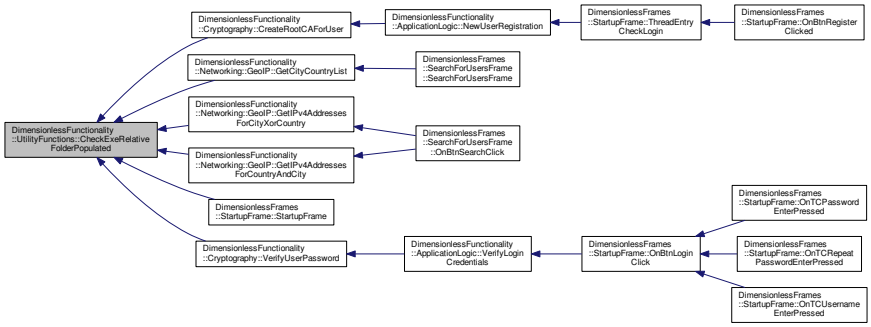
Referenced by DimensionlessFunctionality::Cryptography::CreateRootCAForUser(), DimensionlessFunctionality::Networking::GeoIP::GetCityCountryList(), DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCityXorCountry(), DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCountryAndCity(), DimensionlessFrames::StartupFrame::StartupFrame(), and DimensionlessFunctionality::Cryptography::VerifyUserPassword().

```
145         {
146             if(boost::filesystem::is_directory(ApplicationInfo::GetExecutablePath()+exeRelativeFolderPath))
    {
147                 boost::filesystem::directory_iterator endIterator;
148                 boost::filesystem::directory_iterator currentIterator(ApplicationInfo::GetExecutablePath()+
    exeRelativeFolderPath);
149                 if(currentIterator == endIterator){
150                     return false;
151                 } else {
152                     return true;
153                 }
154             } else {
155                 return false;
156             }
157         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.20.2.2 CheckFileExists()

```
bool DimensionlessFunctionality::UtilityFunctions::CheckFileExists (
            const string & absoluteFilePath )
```

This function checks a file exists at an absolute path.

Parameters

in	*absoluteFilePath*	A string that represents the absolute path of a file to check.

Returns

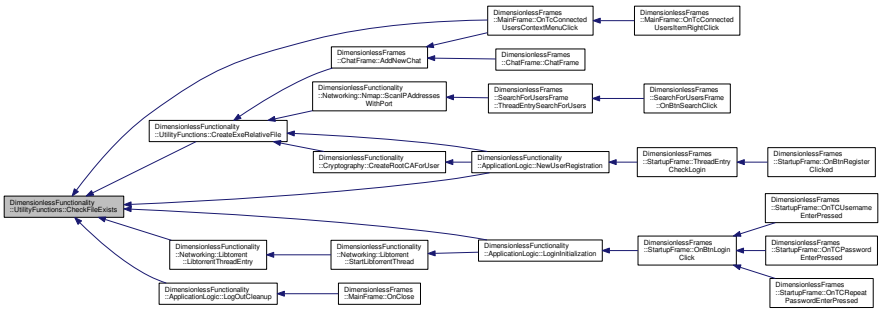
A boolean denoting if the file exists.

Definition at line 62 of file DimensionlessFunctionality.cpp.

Referenced by CreateExeRelativeFile(), DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThread↩ Entry(), DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), DimensionlessFunctionality::↩ ApplicationLogic::LogOutCleanup(), DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
63         {
64             return boost::filesystem::exists(absoluteFilePath);
65         }
```

Here is the caller graph for this function:



11.20.2.3 CheckFolderExists()

```
bool DimensionlessFunctionality::UtilityFunctions::CheckFolderExists (
            const string & absoluteFolderPath )
```

This function checks a folder exists at an absolute path.

Parameters

in	*absoluteFilePath*	A string that represents the absolute path of a folder to check.

Returns

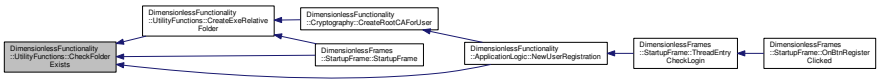
A boolean denoting if the folder exists.

Definition at line 72 of file DimensionlessFunctionality.cpp.

Referenced by CreateExeRelativeFolder(), DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and DimensionlessFrames::StartupFrame::StartupFrame().

```
73         {
74             return boost::filesystem::is_directory(absoluteFolderPath);
75         }
```

Here is the caller graph for this function:



11.20.2.4 ConvertIntegerToString()

```
string DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString (
            int integerToConvert )
```

This function converts an integer value into a string.

Parameters

in	*integerToConvert*	An integer value that is to be converted into a string.

Returns

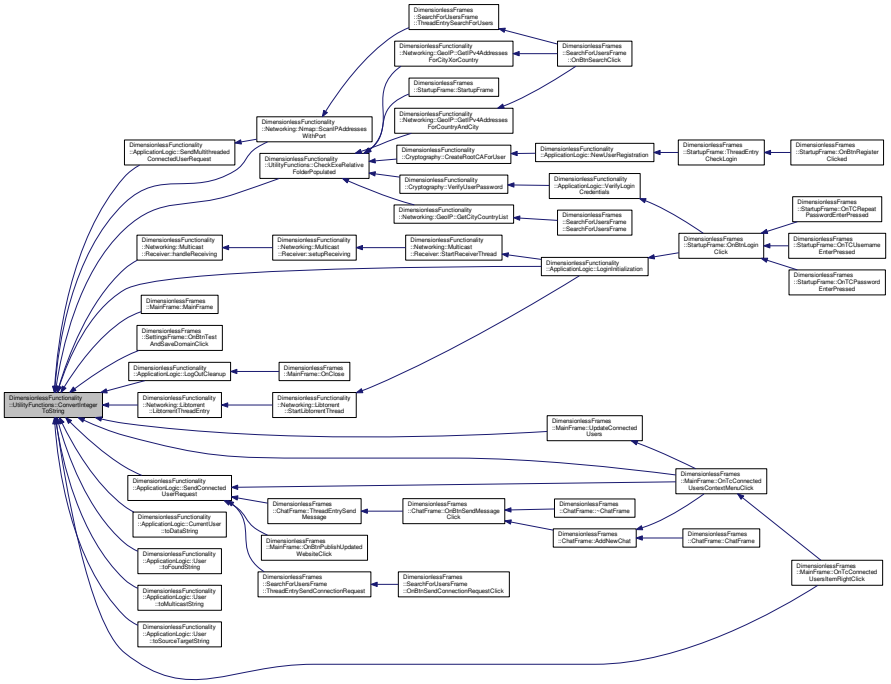
A string representation of the integerToConvert passed.

Definition at line 47 of file DimensionlessFunctionality.cpp.

Referenced by CheckExeRelativeFolderPopulated(), DimensionlessFunctionality::Networking::Multicast↩ ::Receiver::handleReceiving(), DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry(), DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), DimensionlessFunctionality::ApplicationLogic↩ ::LogOutCleanup(), DimensionlessFrames::MainFrame::MainFrame(), DimensionlessFrames::SettingsFrame↩ ::OnBtnTestAndSaveDomainClick(), DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenu↩ Click(), DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick(), DimensionlessFunctionality::↩ Networking::Nmap::ScanIPAddressesWithPort(), DimensionlessFunctionality::ApplicationLogic::SendConnected↩ UserRequest(), DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest(), DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString(), DimensionlessFunctionality::↩ ApplicationLogic::User::toFoundString(), DimensionlessFunctionality::ApplicationLogic::User::toMulticastString(), DimensionlessFunctionality::ApplicationLogic::User::toSourceTargetString(), and DimensionlessFrames::Main↩ Frame::UpdateConnectedUsers().

```
48         {
49             /* Below is a work around for the MinGW-esque compiler we're using on Windows.
50                Some other versions have a built in function for this operation.
51                We basically place the integer in a stream which can be output into a single string. */
52             stringstream convert;
53             convert << integerToConvert;
54             return convert.str();
55         }
```

Here is the caller graph for this function:



11.20.2.5 CreateExeRelativeFile() [1/2]

```
bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile (
            const string & exeRelativeFilePath )
```

This function creates a file at a relative path, if it already exists it does nothing.

Parameters

in	*exeRelativeFilePath*	A string that represents the relative path of a file to create.

Returns

A boolean denoting if the file was created successfully.

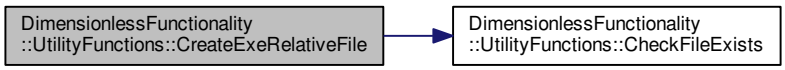
Definition at line 82 of file DimensionlessFunctionality.cpp.

References CheckFileExists().

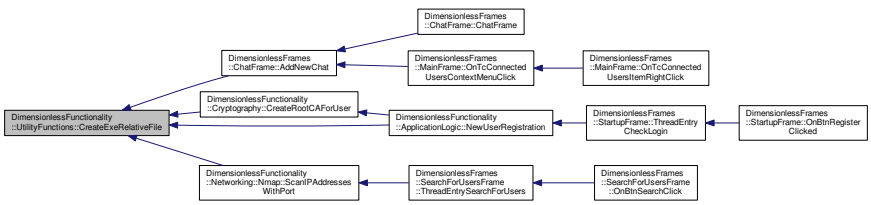
Referenced by DimensionlessFrames::ChatFrame::AddNewChat(), DimensionlessFunctionality::Cryptography::↩ CreateRootCAForUser(), DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and Dimensionless↩ Functionality::Networking::Nmap::ScanIPAddressesWithPort().

```
83         {
84             if(!CheckFileExists(ApplicationInfo::GetExecutablePath()+exeRelativeFilePath)){
85                 std::ofstream outFile (ApplicationInfo::GetExecutablePath()+exeRelativeFilePath);
86                 outFile.close();
87                 return true;
88             } else {
89                 return false;
90             }
91         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.20.2.6 CreateExeRelativeFile() [2/2]

```
bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile (
            const string & exeRelativeFilePath,
            const string & fileContent )
```

This function creates a file at a relative path, if it already exists it does nothing.

Parameters

in	*exeRelativeFilePath*	A string that represents the relative path of a file to create.
in	*fileContent*	A string that contains the data to write to the target file.

Returns

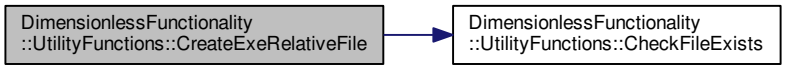
A boolean denoting if the file was created successfully.

Definition at line 99 of file DimensionlessFunctionality.cpp.

References CheckFileExists().

```
100         {
101             if(!CheckFileExists(ApplicationInfo::GetExecutablePath()+exeRelativeFilePath)){
102                 std::ofstream outFile (ApplicationInfo::GetExecutablePath()+exeRelativeFilePath);
103                 outFile << fileContent;
104                 outFile.close();
105                 return true;
106             } else {
107                 return false;
108             }
109         }
```

Here is the call graph for this function:



11.20.2.7 CreateExeRelativeFolder()

```
bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFolder (
            const string & exeRelativeFolderPath )
```

This function creates a folder at a relative path, if it already exists it does nothing.

Parameters

in	*exeRelativeFolderPath*	A string that represents the relative path of a folder to create.

Returns

A boolean denoting if the folder was created successfully.

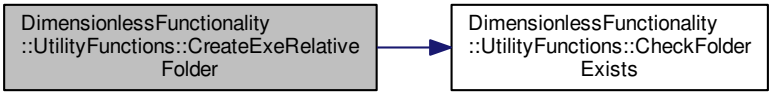
Definition at line 130 of file DimensionlessFunctionality.cpp.

References CheckFolderExists().

Referenced by DimensionlessFunctionality::Cryptography::CreateRootCAForUser(), and DimensionlessFrames::↩ StartupFrame::StartupFrame().

```
131         {
132             if(!CheckFolderExists(ApplicationInfo::GetExecutablePath()+
      exeRelativeFolderPath)){
133                 return boost::filesystem::create_directory(ApplicationInfo::GetExecutablePath()+
      exeRelativeFolderPath);
134             } else {
135                 return false;
136             }
137         }
```

Here is the call graph for this function:



Here is the caller graph for this function:

11.20.2.8 OverwriteExeRelativeFile()

```
bool DimensionlessFunctionality::UtilityFunctions::OverwriteExeRelativeFile (
            const string & exeRelativeFilePath,
            const string & fileContent )
```

This function creates or overwrites a file at a relative path.

Parameters

in	*exeRelativeFilePath*	A string that represents the relative path of a file to create or overwrite.
in	*fileContent*	A string that contains the data to write to the target file.

Returns

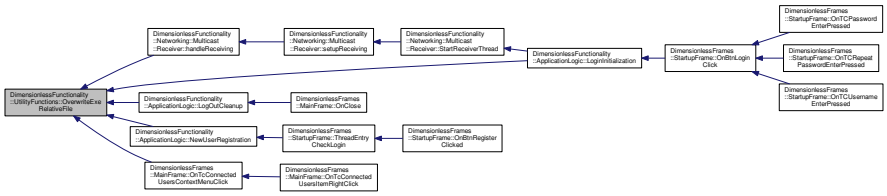
A boolean denoting if the file was created or overwritten successfully.

Definition at line 117 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving(), DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), DimensionlessFunctionality::ApplicationLogic::LogOutCleanup(), DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

```
118         {
119             std::ofstream outFile (ApplicationInfo::GetExecutablePath()+exeRelativeFilePath);
120             outFile << fileContent;
121             outFile.close();
122             return true;
123         }
```

Here is the caller graph for this function:



11.21 DimensionlessFunctionality::Validation Namespace Reference

Enumerations

- enum ValidationType { ValidationType::Null, ValidationType::DataType, ValidationType::Lookup, ValidationType::Format, ValidationType::Length, ValidationType::Presence }

  This enumeration decides the type of validation to be performed.
- enum FormatType { FormatType::Null, FormatType::ZlibCompressed }

  This enumeration decides the format of the data to be validated for the format check.

Functions

- template<typename Obtained , typename CheckableParam >
  bool validate (ValidationType typeOfValidation, Obtained obtainedValue, CheckableParam valueToValidateAgainst, FormatType format=FormatType::Null)
- template<>
  bool validate< string, int > (ValidationType typeOfValidation, string obtainedValue, int valueToValidateAgainst, FormatType format)
- template<>
  bool validate< string, string > (ValidationType typeOfValidation, string obtainedValue, string valueToValidateAgainst, FormatType format)

11.21.1 Detailed Description

The Validation namespace stores all of the functions that allow for validation to be performed on data.

11.21.2 Enumeration Type Documentation

11.21.2.1 FormatType

```
enum DimensionlessFunctionality::Validation::FormatType  [strong]
```

This enumeration decides the format of the data to be validated for the format check.

Enumerator

Null	Denotes that a lack of a format check has been selected.
ZlibCompressed	Denotes that a zlib compression check is to be performed.

Definition at line 472 of file DimensionlessFunctionality.h.

```
472                                 {
473                                   Null,
474                                   ZlibCompressed,
475                                 };
```

11.21.2.2 ValidationType

```
enum DimensionlessFunctionality::Validation::ValidationType  [strong]
```

This enumeration decides the type of validation to be performed.

Enumerator

Null	Denotes that a lack of a validation technique has been selected.
DataType	Denotes that a data type validation check is to be performed.
Lookup	Denotes that a lookup validation check is to be performed.
Format	Denotes that a format validation check is to be performed.
Length	Denotes that a length validation check is to be performed.
Presence	Denotes that a presence validation check is to be performed.

Definition at line 462 of file DimensionlessFunctionality.h.

```
462                                 {
463                                   Null,
```

```
464                                     DataType,
465                                     Lookup,
466                                     Format,
467                                     Length,
468                                     Presence
469                                   };
```

11.21.3 Function Documentation

11.21.3.1 validate()

```
template<typename Obtained , typename CheckableParam >
bool DimensionlessFunctionality::Validation::validate (
            ValidationType typeOfValidation,
            Obtained obtainedValue,
            CheckableParam valueToValidateAgainst,
            FormatType format = FormatType::Null )  [inline]
```

Allow for data of any type to be validated with this template.

Parameters

in	*typeofvalidation*	An enumeration value that decides the type of validation to perform.
in	*obtainedvalue*	A variable to be used in conjunction with the valuetovalidateagainst variable to validate.
in	*valuetovalidateagainst*	An variable to be used in conjunction with the obtainedvalue variable to validate.
in	*format*	An enumeration value that decides the format type to check if a format check is being performed.

Returns

A boolean value indicating if the validation was successful.

Definition at line 486 of file DimensionlessFunctionality.h.

```
486
                                                    {
487                 switch(typeOfValidation){
488
489                     case ValidationType::DataType:
490                         {
491                             // Retrieves the type of the variables and checks the types are identical.
492                             if(typeid(obtainedValue) == typeid(valueToValidateAgainst)){
493
494                                 return true;
495
496                             } else {
497
498                                 return false;
499
500                             }
501
502                             break;
503
504                         }
505                     case ValidationType::Presence:
506                         {
507                             // Most empty C++ variables will return a value of 1 here hence greater than 1
      implies data is present.
```

```
508                             if(sizeof(obtainedValue) > 1){
509
510                                 return true;
511
512                             } else {
513
514                                 return false;
515
516                             }
517
518                             break;
519
520                         }
521                     default:
522                         {
523
524                             return false;
525
526                             break;
527                         }
528
529                     }
530
531                 return false;
532         }
```

11.21.3.2 validate< string, int >()

```
template<>
bool DimensionlessFunctionality::Validation::validate< string, int > (
            ValidationType typeOfValidation,
            string obtainedValue,
            int valueToValidateAgainst,
            FormatType format )  [inline]
```

Allow string data and integer data to be validated using this specialized template.

Parameters

in	*typeofvalidation*	An enumeration value that decides the type of validation to perform.
in	*obtainedvalue*	A string to be used in conjunction with the valuetovalidateagainst variable to validate.
in	*valuetovalidateagainst*	An integer to be used in conjunction with the obtainedvalue variable to validate.
in	*format*	An enumeration value that decides the format type to check if a format check is being performed.

Returns

A boolean value indicating if the validation was successful.

Definition at line 543 of file DimensionlessFunctionality.h.

```
543
                                    {
544             switch(typeOfValidation){
545
546                 case ValidationType::Length:
547                     {
548                         // Since the first data type is a string, we can check its length against the
      integer passed to be interpreted as the length.
549                         string datatocheck = obtainedValue;
```

```
550
551                         unsigned int lengthtocheck = valueToValidateAgainst;
552
553                         if( datatocheck.length() <= lengthtocheck ){
554
555                             return true;
556
557                         } else {
558
559                             return false;
560
561                         }
562
563                         break;
564
565                     }
566                 default:
567                     {
568
569                         return false;
570
571                         break;
572
573                     }
574
575                 return false;
576
577             }
578
579             return false;
580         }
```

11.21.3.3 validate< string, string >()

```
template<>
bool DimensionlessFunctionality::Validation::validate< string, string > (
             ValidationType typeOfValidation,
             string obtainedValue,
             string valueToValidateAgainst,
             FormatType format )  [inline]
```

Allow two pieces of string data to be validated with this specialized template.

Parameters

in	*typeofvalidation*	An enumeration value that decides the type of validation to perform.
in	*obtainedvalue*	A string to be used in conjunction with the valuetovalidateagainst variable to validate.
in	*valuetovalidateagainst*	An string to be used in conjunction with the obtainedvalue variable to validate.
in	*format*	An enumeration value that decides the format type to check if a format check is being performed.

Returns

A boolean value indicating if the validation was successful.

Definition at line 591 of file DimensionlessFunctionality.h.

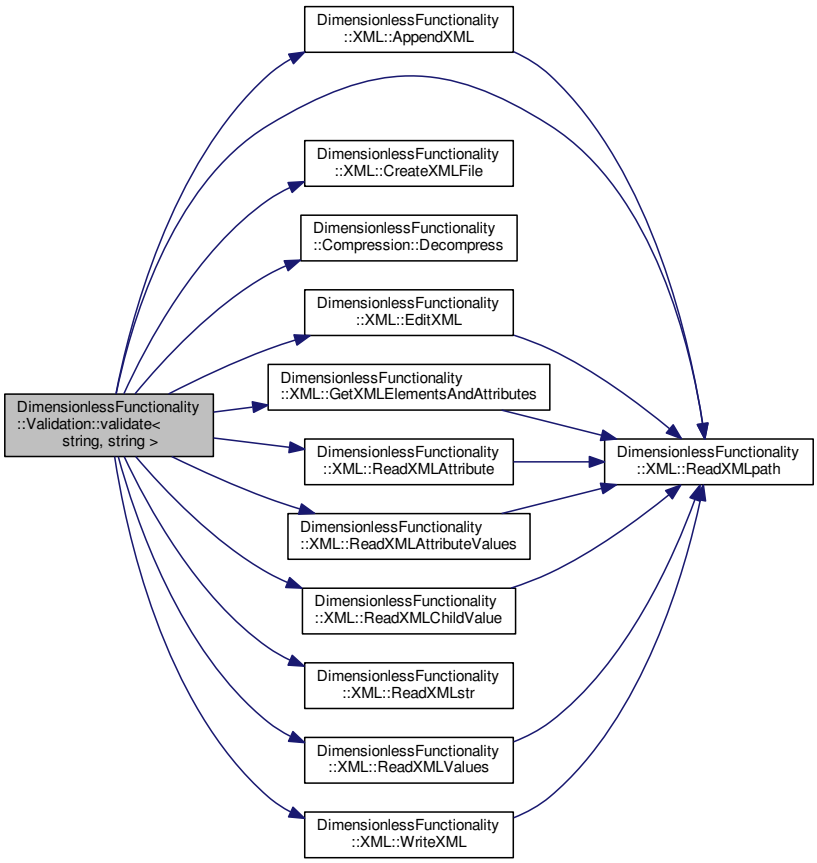
References DimensionlessFunctionality::XML::AppendXML(), DimensionlessFunctionality::XML::CreateXMLFile(), DimensionlessFunctionality::Networking::Multicast::Receiver::data, DimensionlessFunctionality::Compression↩ ::Decompress(), DimensionlessFunctionality::XML::EditXML(), DimensionlessFunctionality::XML::GetXML↩

ElementsAndAttributes(), DimensionlessFunctionality::XML::ReadXMLAttribute(), DimensionlessFunctionality::XML::ReadXMLAttributeValues(), DimensionlessFunctionality::XML::ReadXMLChildValue(), DimensionlessFunctionality::XML::ReadXMLpath(), DimensionlessFunctionality::XML::ReadXMLstr(), DimensionlessFunctionality::XML::ReadXMLValues(), and DimensionlessFunctionality::XML::WriteXML().

```
591
                                                {
592             switch(typeOfValidation){
593
594                 case ValidationType::DataType:
595                     {
596                         // Retrieves the type of the variables and checks the types are identical.
597                         if(typeid(obtainedValue) == typeid(valueToValidateAgainst)){
598
599                             return true;
600
601                         } else {
602
603                             return false;
604
605                         }
606
607                         break;
608
609                     }
610                 case ValidationType::Format:
611                     {
612
613                         switch(format){
614                             case FormatType::ZlibCompressed:
615                                 {
616
617                                     try {
618
619                                         // Given the decompress function causes no errors, it is safe to
       assume the data is validly Zlib compressed.
620                                         string encodeddata =
      Compression::Decompress(obtainedValue);
621
622                                         if(encodeddata.length() > 0){
623
624                                             return true;
625
626                                         } else {
627
628                                             return false;
629
630                                         }
631
632                                     } catch (...){
633
634                                         return false;
635
636                                     }
637
638                                     break;
639
640                                 }
641                             default:
642                                 {
643
644                                     return false;
645
646                                     break;
647
648                                 }
649
650                         }
651
652                         break;
653
654                     }
655                 case ValidationType::Lookup:
656                     {
657
658                         // Check that the first string value is equal to the second string value.
659                         if( obtainedValue == valueToValidateAgainst ){
660
661                             return true;
662
663                         } else {
664
665                             return false;
666
```

```
667                         }
668 
669                         break;
670 
671                     }
672                 case ValidationType::Presence:
673                     {
674 
675                         // Most empty C++ variables will return a value of 1 here hence greater than 1
    implies data is present.
676                         if(obtainedValue.length() > 0){
677 
678                             return true;
679 
680                         } else {
681 
682                             return false;
683 
684                         }
685 
686                         break;
687 
688                     }
689                 default:
690                     {
691 
692                         return false;
693 
694                         break;
695                     }
696 
697             }
698 
699             return false;
700 
701         }
```

Here is the call graph for this function:



11.22 DimensionlessFunctionality::XML Namespace Reference

Functions

- bool CreateXMLFile (string pathToCreateFile, string rootNodeName)
- bool WriteXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributeValues)
- bool AppendXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributeValues)
- bool EditXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributesValues)
- vector< string > ReadXMLValues (string pathToFile, string rootNode, vector< string > childNames)
- vector< string > ReadXMLAttributeValues (string pathToFile, string rootNode, string childName, vector< string > attributeNames)
- string ReadXMLChildValue (string pathToFile, string rootNode, string childName)
- string ReadXMLAttribute (string pathToFile, string rootNode, string childName, string childAttribute)
- vector< string > GetXMLElementsAndAttributes (string pathToFile, string rootNode, string childNameLike, vector< string > attributeNames)
- ptree ReadXMLpath (string pathToFile)
- ptree ReadXMLstr (string &data)

11.22.1 Detailed Description

The XML namespace stores all of the functions that perform any operations on XML files.

11.22.2 Function Documentation

11.22.2.1 AppendXML()

```
bool DimensionlessFunctionality::XML::AppendXML (
            string pathToFile,
            string rootNode,
            string childName,
            string childValue,
            vector< string > childAttributes,
            vector< string > childAttributeValues )
```

This function appends xml data to a pre-existing xml file.

Parameters

in	*PathToFile*	A string that contains the path to the xml file to which data is to be written.
in	*RootNode*	A string that decides the name of the root node within the file in the path specified.
in	*ChildName*	A string that contains the name of the child node to write.
in	*ChildValue*	A string that contains the value of the child node to write.
in	*ChildAttributes*	A string array that contains all the attributes of the new child to write to the xml file.
in	*ChildAttributeValues*	A string array that contains all of the attribute values of the new child to write to the xml file.

Returns

A boolean signifying whether the xml data was appended successfully or not.

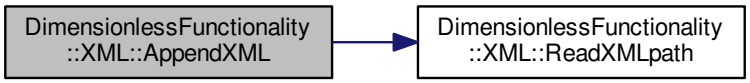
Definition at line 1575 of file DimensionlessFunctionality.cpp.

References ReadXMLpath().

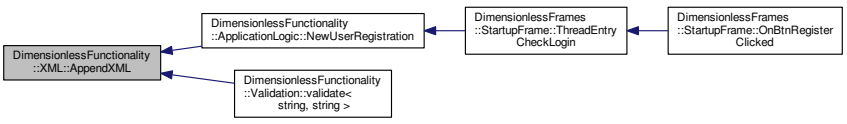
Referenced by DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and DimensionlessFunctionality::Validation::validate< string, string >().

```
1576         {
1577             // Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
       ApplicationInfo::GetPathSeparator().
1578             // Read the old property tree from the file specified.
1579             ptree tree = ReadXMLpath(pathToFile);
1580 
1581             // Store the tree to append to the old tree in a separate property tree object.
1582             ptree appendtree;
1583             appendtree.put("",childValue);
1584 
1585             // Place all the child attributes and values in the tree to be appended to the old tree.
1586             for(unsigned int i = 0; i < childAttributes.size(); i++){
1587 
1588                 appendtree.put("<xmlattr>."+childAttributes[i],childAttributeValues[i]);
1589 
1590             }
1591 
1592             tree.add_child(rootNode+"."+childName, appendtree);
1593 
1594             //Write the new tree to the XML file.
1595             write_xml(pathToFile, tree);
1596 
1597             return true;
1598         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.2 CreateXMLFile()

```
bool DimensionlessFunctionality::XML::CreateXMLFile (
             string pathToCreateFile,
             string rootNodeName )
```

This function creates an xml file on the path specified along with the root node specified.

Parameters

in	*PathToCreateFile*	A string containing the full .xml file name and path to create.
in	*RootNodeName*	A string that forms the root node of the xml file created.

Returns

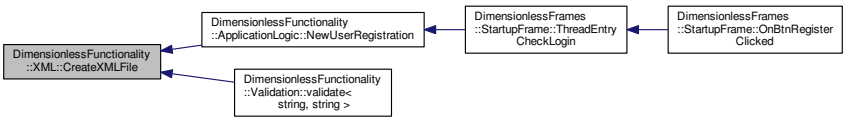
A boolean signifying whether the xml file was created successfully or not.

Definition at line 1519 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), and DimensionlessFunctionality::Validation::validate< string, string >().

```
1520         {
1521             // Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
     ApplicationInfo::GetPathSeparator().
1522             ofstream out(pathToCreateFile, ofstream::binary);
1523 
1524             // Use a boost property tree object to store the xml tree we want written.
1525             ptree tree;
1526             tree.put(rootNodeName,"");
1527 
1528             // Create the XML file and add the associated Root Node in a property tree, format the file
     with 4 spaces as an indent.
1529             write_xml(out, tree, xml_writer_settings<string>(' ', 4));
1530 
1531             return true;
1532         }
```

Here is the caller graph for this function:



11.22.2.3 EditXML()

```
bool DimensionlessFunctionality::XML::EditXML (
            string pathToFile,
            string rootNode,
            string childName,
            string childValue,
            vector< string > childAttributes,
            vector< string > childAttributesValues )
```

This function edits data in the xml file specified.

Parameters

in	*PathToFile*	A string that contains the path to the xml file in which data is to be edited.
in	*RootNode*	A string that decides the name of the root node within the file in the path specified to edit.
in	*ChildName*	A string that contains the name of the child node to edit.
in	*ChildValue*	A string that contains the value of the child node to edit.
in	*ChildAttributes*	A string array that contains all the attributes of the child to edit within the xml file.
in	*ChildAttributeValues*	A string array that contains all of the attribute values of the child to edit within the xml file.

Returns

A boolean signifying whether the xml data was edited successfully or not.

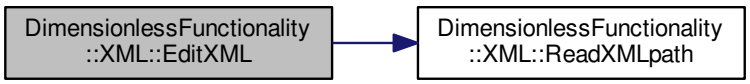
Definition at line 1610 of file DimensionlessFunctionality.cpp.

References ReadXMLpath().

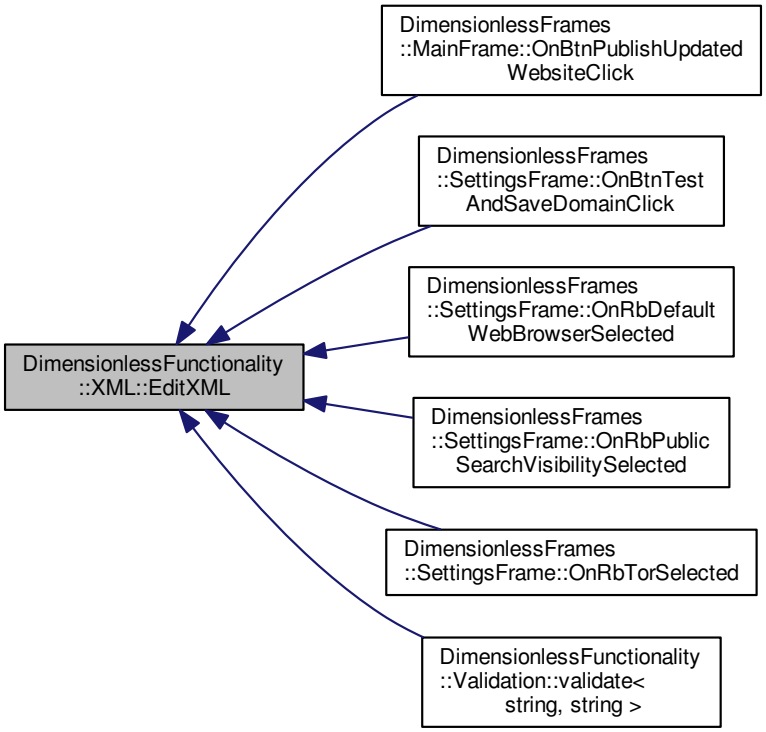
Referenced by DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick(), DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick(), DimensionlessFrames::SettingsFrame::OnRbDefaultWebBrowserSelected(), DimensionlessFrames::SettingsFrame::OnRbPublicSearchVisibilitySelected(), DimensionlessFrames::SettingsFrame::OnRbTorSelected(), and DimensionlessFunctionality::Validation::validate< string, string >().

```
1611         {
1612             // Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
      ApplicationInfo::GetPathSeparator().
1613             ptree tree = ReadXMLpath(pathToFile);
1614 
1615             // Overwrite the old child node in the tree.
1616             tree.put(rootNode+"."+childName, childValue);
1617 
1618             ptree &child = tree.get_child(rootNode+"."+childName);
1619 
1620             // Edit the specific child's attributes and values.
1621             for(unsigned int i = 0; i < childAttributes.size(); i++){
1622 
1623                 //
      tree.put(rootNode+"."+childName+".<xmlattr>."+childAttributes[i],childAttributesValues[i]);
1624                 child.put("<xmlattr>."+childAttributes[i], childAttributesValues[i]);
1625 
1626             }
1627 
1628             //Write the new tree to the XML file and format the file with 4 spaces as an indent per node
1629             write_xml(pathToFile, tree);
1630 
1631             return true;
1632         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.4 GetXMLElementsAndAttributes()

```
vector< string > DimensionlessFunctionality::XML::GetXMLElementsAndAttributes (
            string pathToFile,
            string rootNode,
            string childNameLike,
            vector< string > attributeNames )
```

This function gets all of the xml attribute values for a certain child.

Parameters

in	*PathToFile*	A string that contains the location of the xml file to read from.
in	*RootNode*	A string that contains the name of the root node within the xml file to read from.
in	*ChildNameLike*	A string that contains the name of the repeated child node to read attribute values for.

Returns

A string array containing all the attribute values of the repeated child node.

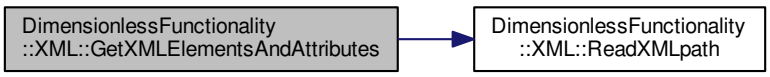
Definition at line 1739 of file DimensionlessFunctionality.cpp.

References ReadXMLpath().

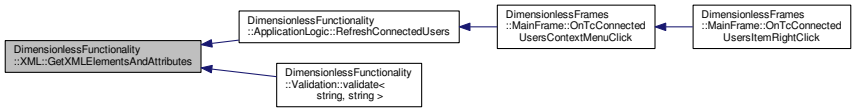
Referenced by DimensionlessFunctionality::ApplicationLogic::RefreshConnectedUsers(), and DimensionlessFunctionality::Validation::validate< string, string >().

```
1740         {
1741             ptree pt = ReadXMLpath(pathToFile);
1742 
1743             vector<string> ans;
1744 
1745             // This function is specialized for the ResumeManagementFrame to obtain all the attribute
     values for child nodes of the same name.
1746             BOOST_FOREACH( ptree::value_type const& v, pt.get_child(rootNode) ) {
1747                 if( v.first == childNameLike ) {
1748                     ans.push_back(v.second.data());
1749                     for(unsigned int i = 0; i < attributeNames.size(); i++){
1750                         ans.push_back(v.second.get<string>("<xmlattr>."+attributeNames[i], ""));
1751                     }
1752                 }
1753             }
1754 
1755             return ans;
1756         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.5 ReadXMLAttribute()

```
string DimensionlessFunctionality::XML::ReadXMLAttribute (
            string pathToFile,
            string rootNode,
            string childName,
            string childAttribute )
```

This function reads a single attribute value that a child node contains.

Parameters

in	*PathToFile*	A string that contains the path to the xml file from which data is to be read.
in	*RootNode*	A string that contains the name of the root node within the xml file specified to read from.
in	*ChildName*	A string that contains the name of the child whose value is to be read.
in	*ChildAttribute*	A string that contains the name of the child attribute to get the value for.

Returns

A string containing the value of the requested attribute for a certain child in the xml file specified.

Definition at line 1716 of file DimensionlessFunctionality.cpp.

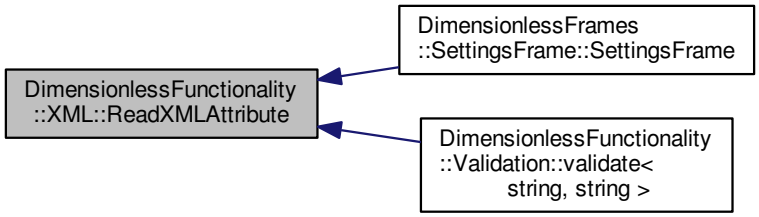
References ReadXMLpath().

Referenced by DimensionlessFrames::SettingsFrame::SettingsFrame(), and DimensionlessFunctionality::Validation::validate< string, string >().

```
1717         {
1718             ptree tree = ReadXMLpath(pathToFile);
1719 
1720             try{
1721 
1722                 // Return a particular child's attribute value.
1723                 return tree.get<string>(rootNode+"."+childName+".<xmlattr>."+childAttribute);
1724 
1725             } catch(...){
1726 
1727                 return "";
1728 
1729             }
1730         }
```

Here is the call graph for this function:

Here is the caller graph for this function:



11.22.2.6 ReadXMLAttributeValues()

```
vector< string > DimensionlessFunctionality::XML::ReadXMLAttributeValues (
            string pathToFile,
            string rootNode,
            string childName,
            vector< string > attributeNames )
```

(Not currently used in the application) This function reads xml attribute values from the path specified.

Parameters

in	*PathToFile*	A string that contains the path to the xml file from which data is to be read.
in	*RootNode*	A string of the root node of the xml tree to be read from.
in	*ChildName*	A string of the child name whose attributes are to be read.
in	*AttributeNames*	A string array of all the attribute names whose values are to be read.

Returns

A string array with all the attribute values read.

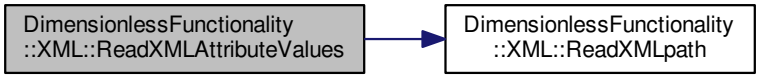
Definition at line 1667 of file DimensionlessFunctionality.cpp.

References ReadXMLpath().

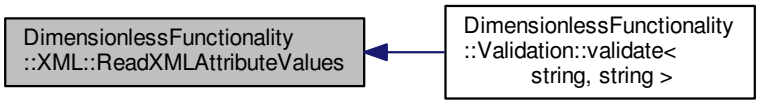
Referenced by DimensionlessFunctionality::Validation::validate< string, string >().

```
1668         {
1669             //Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
       ApplicationInfo::GetPathSeparator()
1670             ptree tree = ReadXMLpath(pathToFile);
1671 
1672             // Store the child attribute data in a string array.
1673             vector<string> ChildAttributeData (attributeNames.size());
1674 
1675             // Retrieve the child attribute data.
1676             for(unsigned int i = 0; i < attributeNames.size(); i++){
1677 
1678                 ChildAttributeData[i] = tree.get<string>(rootNode+"."+childName+".<xmlattr>."+
      attributeNames[i]);
1679 
1680             }
1681 
1682             return ChildAttributeData;
1683         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.7 ReadXMLChildValue()

```
string DimensionlessFunctionality::XML::ReadXMLChildValue (
            string pathToFile,
            string rootNode,
            string childName )
```

This function reads a single child node's value and returns it back.

Parameters

in	*PathToFile*	A string that contains the path to the xml file from which data is to be read.
in	*RootNode*	A string that contains the name of the root node within the xml file specified to read from.
in	*ChildName*	A string that contains the name of the child whose value is to be read.

Returns

A string containing the value of the desired child node.

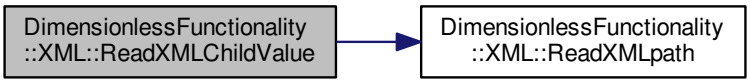
Definition at line 1692 of file DimensionlessFunctionality.cpp.

References ReadXMLpath().

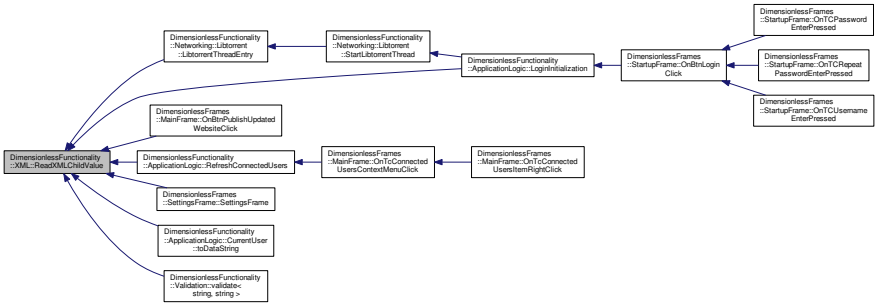
Referenced by DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry(), DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick(), DimensionlessFunctionality::ApplicationLogic::RefreshConnectedUsers(), DimensionlessFrames::SettingsFrame::SettingsFrame(), DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString(), and DimensionlessFunctionality::Validation::validate< string, string >().

```
1693         {
1694             ptree tree = ReadXMLpath(pathToFile);
1695
1696             try{
1697
1698                 // return a particular child's value.
1699                 return tree.get<string>(rootNode+"."+childName);
1700
1701             } catch(...){
1702
1703                 return "";
1704
1705             }
1706         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.8 ReadXMLpath()

```
ptree DimensionlessFunctionality::XML::ReadXMLpath (
            string pathToFile )
```

This function reads an xml tree from the file specified.

Parameters

in	*PathToFile*	A string of the path to an xml file to read the xml tree for.

Returns

A boost property tree object that contains all the xml data from the file specified.

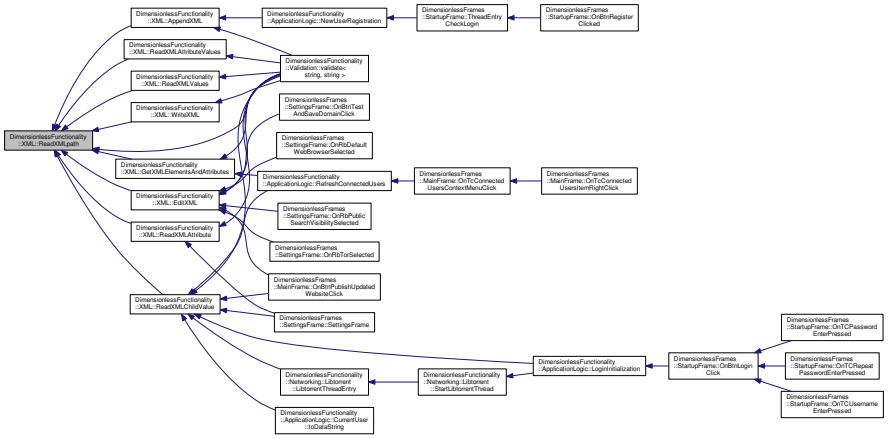
Definition at line 1763 of file DimensionlessFunctionality.cpp.

Referenced by AppendXML(), EditXML(), GetXMLElementsAndAttributes(), ReadXMLAttribute(), ReadXML↩AttributeValues(), ReadXMLChildValue(), ReadXMLValues(), DimensionlessFunctionality::Validation::validate< string, string >(), and WriteXML().

```
1764         {
```

```
1765             // Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
       ApplicationInfo::GetPathSeparator().
1766             ptree tree;
1767             read_xml(pathToFile, tree);
1768             return tree;
1769         }
```

Here is the caller graph for this function:



11.22.2.9 ReadXMLstr()

```
ptree DimensionlessFunctionality::XML::ReadXMLstr (
            string & data )
```

(Not currently used in the application) This function converts an string full of xml data into a boost property tree object.

Parameters

in	*Data*	A string passed by reference that is to be read into a boost property tree object.

Returns

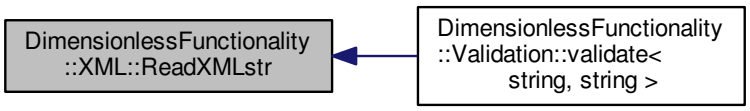
A boost property tree object that contains all the xml data from the string specified.

Definition at line 1776 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::Validation::validate< string, string >().

```
1777         {
1778             ptree tree;
1779             stringstream stream(data);
1780             read_xml(stream, tree);
1781             return tree;
1782         }
```

Here is the caller graph for this function:



11.22.2.10 ReadXMLValues()

```
vector< string > DimensionlessFunctionality::XML::ReadXMLValues (
            string pathToFile,
            string rootNode,
            vector< string > childNames )
```

(Not currently used in the application) This function reads child data from the xml file in the directory specified.

Parameters

in	*PathToFile*	A string that contains the path to the xml file from which data is to be read.
in	*RootNode*	A string that signifies the root node of the xml file in the path specified.
in	*ChildNames*	A string array of all the children whose values are to be read from the xml file.

Returns

A string array of the data read from the xml file.

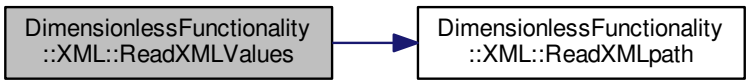
Definition at line 1641 of file DimensionlessFunctionality.cpp.

References ReadXMLpath().

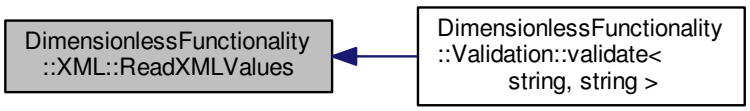
Referenced by DimensionlessFunctionality::Validation::validate< string, string >().

```
1642         {
1643             //Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
       ApplicationInfo::GetPathSeparator()
1644             ptree tree = ReadXMLpath(pathToFile);
1645 
1646             // Store the child data in a string array.
1647             vector<string> ChildData (childNames.size());
1648 
1649             // Retrieve the child data.
1650             for(unsigned int i = 0; i < childNames.size(); i++){
1651 
1652                 ChildData[i] = tree.get<string>(rootNode+"."+childNames[i]);
1653 
1654             }
1655 
1656             return ChildData;
1657         }
```

Here is the call graph for this function:



Here is the caller graph for this function:



11.22.2.11 WriteXML()

```
bool DimensionlessFunctionality::XML::WriteXML (
            string pathToFile,
            string rootNode,
            string childName,
            string childValue,
            vector< string > childAttributes,
            vector< string > childAttributeValues )
```

This function writes xml data to the file specified.

Parameters

in	*PathToFile*	A string that contains the path to the xml file to which data is to be written.
in	*RootNode*	A string that decides the name of the root node within the file in the path specified.
in	*ChildName*	A string that contains the name of the child node to write.
in	*ChildValue*	A string that contains the value of the child node to write.
in	*ChildAttributes*	A string array that contains all the attributes of the new child to write to the xml file.
in	*ChildAttributeValues*	A string array that contains all of the attribute values of the new child to write to the xml file.

Returns

A boolean signifying whether the xml data was written successfully or not.

Definition at line 1544 of file DimensionlessFunctionality.cpp.

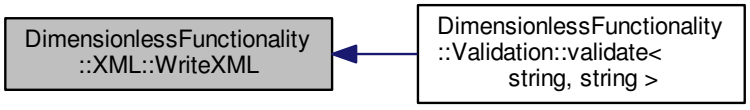
References ReadXMLpath().

Referenced by DimensionlessFunctionality::Validation::validate< string, string >().

```
1545         {
1546             // Note to self: Ensure PathToFile path separator char is escaped if required otherwise use
       ApplicationInfo::GetPathSeparator().
1547             // Retrieve the xml tree from the file specified.
1548             ptree tree = ReadXMLpath(pathToFile);
1549 
1550             tree.put(rootNode+"."+childName,childValue);
1551 
1552             // Place all the child attributes and values in the tree.
1553             for(unsigned int i = 0; i < childAttributes.size(); i++){
1554 
1555                 tree.put(rootNode+"."+childName+".<xmlattr>."+childAttributes[i],childAttributeValues[i]);
1556 
1557             }
1558 
1559             //Write the new tree to the XML file.
1560             write_xml(pathToFile, tree);
1561 
1562             return true;
1563         }
```

Here is the call graph for this function:

Here is the caller graph for this function:



Chapter 12

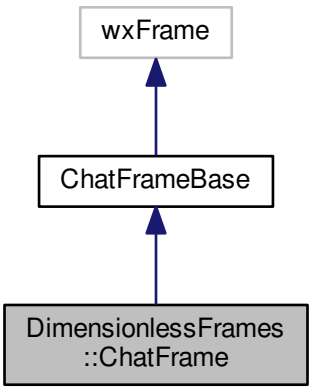
Class Documentation

12.1 DimensionlessFrames::ChatFrame Class Reference

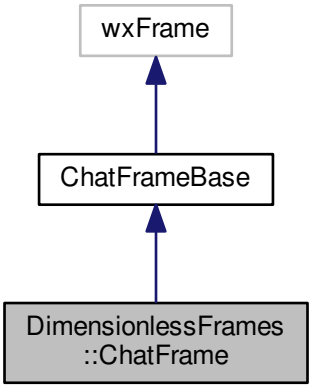
This class creates a ChatFrame to chat with other ConnectedUsers.

`#include <ChatFrame.h>`

Inheritance diagram for DimensionlessFrames::ChatFrame:



Collaboration diagram for DimensionlessFrames::ChatFrame:



Classes

- struct ChatFrameThreadHandler

  This struct handles events posted from our chatFrameThread.

Public Member Functions

- ChatFrame (wxWindow *parent, const std::shared_ptr< ConnectedUser > &connectedChatUser)
- virtual ~ChatFrame ()
- void AddNewChat (const std::shared_ptr< ConnectedUser > &connectedChatUser)

Protected Member Functions

- virtual void OnNbChatPageChanged (wxNotebookEvent &event)
- virtual void OnBtnSendMessageClick (wxCommandEvent &event)
- virtual void OnBtnCloseChatPageClick (wxCommandEvent &event)

Private Member Functions

- void ThreadEntrySendMessage (const std::shared_ptr< ConnectedUser > &connectedChatUser, const string &message, const int &index)

Private Attributes

- wxWindow * ParentWindow

  This raw pointer stores a reference to the parent window of this ChatFrame.
- std::vector< wxPanel *> nbChatPanels

  This vector of raw pointers stores a reference to the wxPanels on each ChatFrame notebook page.
- std::vector< wxBoxSizer *> nbChatBszrs

  This vector of raw pointers stores a reference to the wxBoxSizers on each ChatFrame notebook page.
- std::vector< wxRichTextCtrl *> nbChatRTChatLog

  This vector of raw pointers stores a reference to the wxRichTextCtrls on each ChatFrame notebook page.
- std::vector< wxBoxSizer *> nbNewChatMessageBszrs

  This vector of raw pointers stores a reference to the wxBoxSizers on each ChatFrame notebook page.
- std::vector< wxTextCtrl *> nbTCNewMessage

  This vector of raw pointers stores a reference to the wxTextCtrls on each ChatFrame notebook page.
- std::vector< wxButton *> nbSendMessageBtn

  This vector of raw pointers stores a reference to the wxButtons on each ChatFrame notebook page.
- std::vector< wxButton *> nbCloseChatPageBtn

  This vector of raw pointers stores a reference to the wxButtons on each ChatFrame notebook page.

Static Private Attributes

- static std::thread chatFrameThread

  This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Additional Inherited Members

12.1.1 Detailed Description

This class creates a ChatFrame to chat with other ConnectedUsers.

Definition at line 57 of file ChatFrame.h.

12.1.2 Constructor & Destructor Documentation

12.1.2.1 ChatFrame()

```
DimensionlessFrames::ChatFrame::ChatFrame (
            wxWindow * parent,
            const std::shared_ptr< ConnectedUser > & connectedChatUser )
```

This constructor initializes a new ChatFrame for the desired ConnectedUser.

Parameters

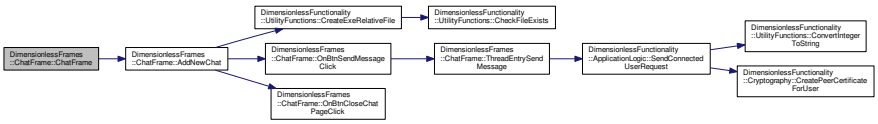
in	*parent*	The parent window of the one to be created.
in	*connectedChatUser*	A smart shared pointer reference to a ConnectedUser for which a ChatFrame is to be created.

Definition at line 96 of file ChatFrame.cpp.

References AddNewChat(), and ParentWindow.

```
96                                                                                                                  :
      ChatFrameBase(parent)
97     {
98         this->Bind(wxEVT_THREAD, ChatFrame::ChatFrameThreadHandler(this), wxID_ANY);
99         this->ParentWindow = parent;
100         this->SetMinClientSize(wxSize(500,400));
101         this->SetSize(500,400);
102         this->CentreOnParent(wxBOTH);
103         this->SetTitle("Chat");
104         AddNewChat(connectedChatUser);
105     }
```

Here is the call graph for this function:



12.1.2.2 ~ChatFrame()

```
DimensionlessFrames::ChatFrame::~ChatFrame ( )  [virtual]
```

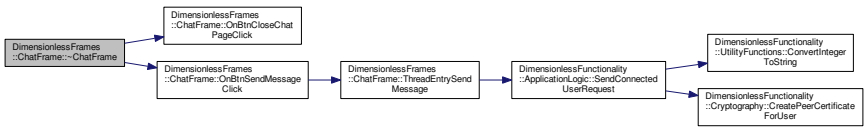
This destructor unbinds the previously binded and connected events for the ChatFrame instance.

Definition at line 110 of file ChatFrame.cpp.

References nbCloseChatPageBtn, nbSendMessageBtn, OnBtnCloseChatPageClick(), OnBtnSendMessageClick(), and ParentWindow.

```
111     {
112         this->Unbind(wxEVT_THREAD, ChatFrame::ChatFrameThreadHandler(this), wxID_ANY);
113         ((MainFrame*)this->ParentWindow)->chatFrame = nullptr;
114         for(unsigned int i = 0; i < nbSendMessageBtn.size(); i++){
115             nbSendMessageBtn[i]->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(ChatFrame::OnBtnSendMessageClick), NULL, this);
116             nbCloseChatPageBtn[i]->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(ChatFrame::OnBtnCloseChatPageClick), NULL, this);
117         }
118     }
```

Here is the call graph for this function:



12.1.3 Member Function Documentation

12.1.3.1 AddNewChat()

```
void DimensionlessFrames::ChatFrame::AddNewChat (
            const std::shared_ptr< ConnectedUser > & connectedChatUser )
```

This function adds a new page to a ChatFrame's notebook for another ConnectedUser.

Parameters

in	*connectedChatUser*	A smart shared pointer reference to a ConnectedUser for which to add a page to our ChatFrame for.

Definition at line 124 of file ChatFrame.cpp.

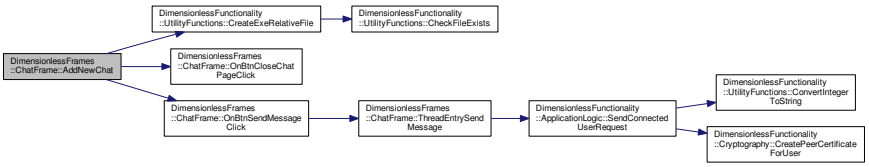
References DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile(), ChatFrameBase::nbChat, nbChatBszrs, nbChatPanels, nbChatRTChatLog, nbCloseChatPageBtn, nbNewChatMessageBszrs, nbSendMessageBtn, nbTCNewMessage, OnBtnCloseChatPageClick(), OnBtnSendMessageClick(), and WXC_FROM_DIP.

Referenced by ChatFrame(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick().

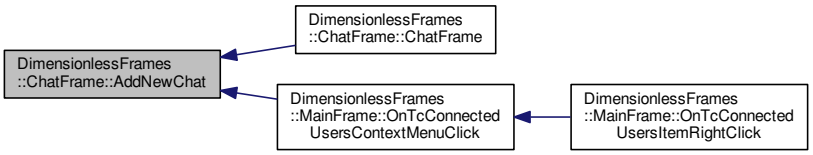
```
125     {
126         for(unsigned int i = 0; i < nbChat->GetPageCount(); i++){
127             if(nbChat->GetPageText(i).ToStdString() == connectedChatUser->username){
128                 nbChat->SetSelection(i);
129                 nbChatRTChatLog[i]->LoadFile(nbChatRTChatLog[i]->GetFilename(
    ));
130                 return;
131             }
132         }
133 
134         nbChatPanels.push_back(new wxPanel(nbChat, wxID_ANY, wxDefaultPosition,
    wxDLG_UNIT(nbChat, wxSize(-1,-1)), wxTAB_TRAVERSAL));
135         nbChat->AddPage(nbChatPanels[nbChatPanels.size()-1], _(
    connectedChatUser->username), false);
136 
137         nbChatBszrs.push_back(new wxBoxSizer(wxVERTICAL));
138         nbChatPanels[nbChatPanels.size()-1]->SetSizer(
    nbChatBszrs[nbChatBszrs.size()-1]);
139 
140         nbChatRTChatLog.push_back(new wxRichTextCtrl(nbChatPanels[
    nbChatPanels.size()-1], wxID_ANY, wxT("Loading chat log..."), wxDefaultPosition, wxDLG_UNIT(
    nbChatPanels[nbChatPanels.size()-1], wxSize(-1,-1)), wxTE_READONLY|wxTE_MULTILINE|
    wxTE_PROCESS_TAB|wxTE_PROCESS_ENTER|wxWANTS_CHARS));
141         DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile
    ("./Users/"+connectedChatUser->username+"/chatlog.txt");
142         nbChatRTChatLog[nbChatRTChatLog.size()-1]->SetFilename("./Users/"+
    connectedChatUser->username+"/chatlog.txt");
143         nbChatRTChatLog[nbChatRTChatLog.size()-1]->LoadFile(
    nbChatRTChatLog[nbChatRTChatLog.size()-1]->GetFilename());
144         nbChatBszrs[nbChatBszrs.size()-1]->Add(
    nbChatRTChatLog[nbChatRTChatLog.size()-1], 1, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
145 
146         nbNewChatMessageBszrs.push_back(new wxBoxSizer(wxHORIZONTAL));
147 
148         nbChatBszrs[nbChatBszrs.size()-1]->Add(
    nbNewChatMessageBszrs[nbNewChatMessageBszrs.size()-1], 0, wxALL|
    wxEXPAND, WXC_FROM_DIP(5));
149 
150         nbTCNewMessage.push_back(new wxTextCtrl(nbChatPanels[
    nbChatPanels.size()-1], wxID_ANY, wxT(""), wxDefaultPosition, wxDLG_UNIT(
    nbChatPanels[nbChatPanels.size()-1], wxSize(-1,-1)), 0));
151         #if wxVERSION_NUMBER >= 3000
```

```
152             nbTCNewMessage[nbTCNewMessage.size()-1]->SetHint(_("Enter chat
      message here"));
153         #endif
154
155         nbNewChatMessageBszrs[nbNewChatMessageBszrs.size()-1]->
      Add(nbTCNewMessage[nbTCNewMessage.size()-1], 1, wxALL|wxALIGN_CENTER_HORIZONTAL|
      wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
156
157         nbSendMessageBtn.push_back(new wxButton(nbChatPanels[
      nbChatPanels.size()-1], wxID_ANY, _("Send Message"), wxDefaultPosition, wxDLG_UNIT(
      nbChatPanels[nbChatPanels.size()-1], wxSize(-1,-1)), 0));
158         nbNewChatMessageBszrs[nbNewChatMessageBszrs.size()-1]->
      Add(nbSendMessageBtn[nbSendMessageBtn.size()-1], 0, wxALL,
      WXC_FROM_DIP(5));
159
160         nbCloseChatPageBtn.push_back(new wxButton(nbChatPanels[
      nbChatPanels.size()-1], wxID_ANY, _("Close Chat Page"), wxDefaultPosition, wxDLG_UNIT(
      nbChatPanels[nbChatPanels.size()-1], wxSize(-1,-1)), 0));
161         nbNewChatMessageBszrs[nbNewChatMessageBszrs.size()-1]->
      Add(nbCloseChatPageBtn[nbCloseChatPageBtn.size()-1], 0, wxALL,
      WXC_FROM_DIP(5));
162
163         nbSendMessageBtn[nbSendMessageBtn.size()-1]->Connect(
      wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(ChatFrame::OnBtnSendMessageClick), (
      wxObject*)new ChatUser(nbSendMessageBtn.size()-1, connectedChatUser), this);
164         nbCloseChatPageBtn[nbCloseChatPageBtn.size()-1]->Connect(
      wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(ChatFrame::OnBtnCloseChatPageClick
      ), (wxObject*)new ChatUser(nbCloseChatPageBtn.size()-1, connectedChatUser), this);
165     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.1.3.2 OnBtnCloseChatPageClick()

```
void DimensionlessFrames::ChatFrame::OnBtnCloseChatPageClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Close Chat" button is clicked on a ChatFrame notebook's page.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Close Chat" button is clicked.

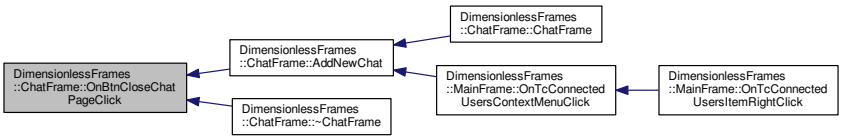
Definition at line 189 of file ChatFrame.cpp.

References ChatFrameBase::nbChat, and DimensionlessFrames::ChatUser::targetUserIndex.

Referenced by AddNewChat(), and ~ChatFrame().

```
190     {
191         ChatUser* chatuser = (ChatUser*)event.GetEventUserData();
192         if(nbChat->GetPageCount() == 1){
193             this->Destroy();
194         }
195         nbChat->RemovePage(chatuser->targetUserIndex);
196     }
```

Here is the caller graph for this function:



12.1.3.3 OnBtnSendMessageClick()

```
void DimensionlessFrames::ChatFrame::OnBtnSendMessageClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Send Message" button is clicked on a ChatFrame notebook's page.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Send Message" button is clicked.

Definition at line 171 of file ChatFrame.cpp.

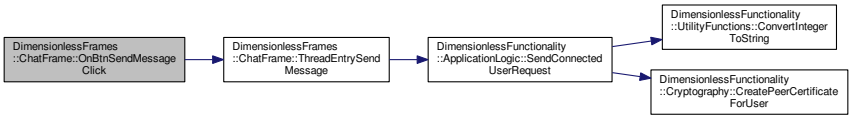
References chatFrameThread, ChatFrameBase::nbChat, nbCloseChatPageBtn, nbSendMessageBtn, nbTCNew↩ Message, DimensionlessFrames::ChatUser::targetChatUser, DimensionlessFrames::ChatUser::targetUserIndex, and ThreadEntrySendMessage().

Referenced by AddNewChat(), and ~ChatFrame().

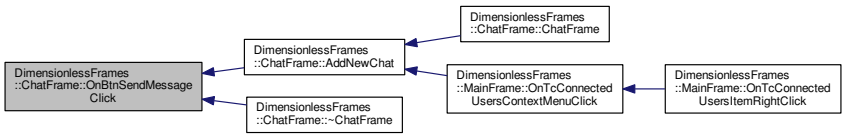
```
172     {
173         ChatUser* chatuser = (ChatUser*)event.GetEventUserData();
174         nbSendMessageBtn[chatuser->targetUserIndex]->Enable(false);
```

```
175         nbCloseChatPageBtn[chatuser->targetUserIndex]->Enable(false);
176         nbChat->Enable(false);
177         std::thread::id nothread;
178         if(chatFrameThread.get_id() != nothread){
179             chatFrameThread.~thread();
180         }
181         chatFrameThread = std::thread(&
      ChatFrame::ThreadEntrySendMessage, this, chatuser->targetChatUser,
      nbTCNewMessage[chatuser->targetUserIndex]->GetValue().ToStdString(), chatuser->
      targetUserIndex);
182         chatFrameThread.detach();
183     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.1.3.4 OnNbChatPageChanged()

```
void DimensionlessFrames::ChatFrame::OnNbChatPageChanged (
            wxNotebookEvent & event )  [protected], [virtual]
```

This function loads a chat log from file when the current ChatFrame notebook page is changed.

Parameters

in	*event*	A wxNotebookEvent that is posted when a ChatFrame's notebook has its currently selected page changed.

Reimplemented from ChatFrameBase.

Definition at line 216 of file ChatFrame.cpp.

References nbChatRTChatLog.

```
217     {
```

```
218         if(nbChatRTChatLog.size() > 0){
219             nbChatRTChatLog[event.GetSelection()]->LoadFile(
    nbChatRTChatLog[event.GetSelection()]->GetFilename());
220         }
221     }
```

12.1.3.5 ThreadEntrySendMessage()

```
void DimensionlessFrames::ChatFrame::ThreadEntrySendMessage (
            const std::shared_ptr< ConnectedUser > & connectedChatUser,
            const string & message,
            const int & index )  [private]
```

This function is called by a chatFrameThread when the "Send Message" button is clicked on a ChatFrame page.

Parameters

in	*connectedChatUser*	A smart shared pointer reference to a ConnectedUser to send a message to.
in	*message*	A string that contains the message to send to another ConnectedUser.
in	*index*	An integer that represents the index of the ConnectedUser to send a message to.

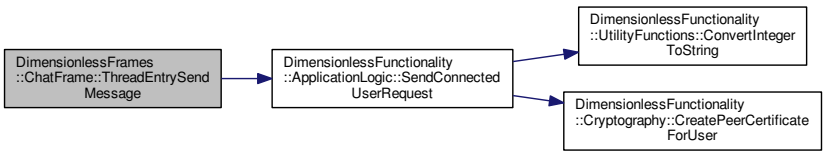
Definition at line 204 of file ChatFrame.cpp.

References DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest().

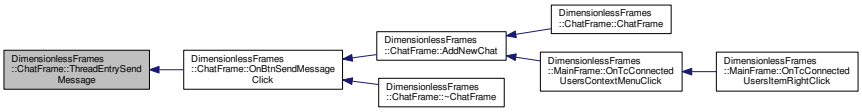
Referenced by OnBtnSendMessageClick().

```
205     {
206         wxThreadEvent event(wxEVT_THREAD, wxID_ANY);
207         SendConnectedUserRequest(Request::SendMessage, connectedChatUser, message);
208         event.SetInt(index);
209         wxPostEvent(this, event);
210     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.1.4 Member Data Documentation

12.1.4.1 chatFrameThread

```
std::thread DimensionlessFrames::ChatFrame::chatFrameThread  [static], [private]
```

This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Definition at line 79 of file ChatFrame.h.

Referenced by OnBtnSendMessageClick().

12.1.4.2 nbChatBszrs

```
std::vector< wxBoxSizer* > DimensionlessFrames::ChatFrame::nbChatBszrs  [private]
```

This vector of raw pointers stores a reference to the wxBoxSizers on each ChatFrame notebook page.

Definition at line 91 of file ChatFrame.h.

Referenced by AddNewChat().

12.1.4.3 nbChatPanels

```
std::vector< wxPanel* > DimensionlessFrames::ChatFrame::nbChatPanels  [private]
```

This vector of raw pointers stores a reference to the wxPanels on each ChatFrame notebook page.

Definition at line 88 of file ChatFrame.h.

Referenced by AddNewChat().

12.1.4.4 nbChatRTChatLog

```
std::vector< wxRichTextCtrl* > DimensionlessFrames::ChatFrame::nbChatRTChatLog  [private]
```

This vector of raw pointers stores a reference to the wxRichTextCtrls on each ChatFrame notebook page.

Definition at line 94 of file ChatFrame.h.

Referenced by AddNewChat(), and OnNbChatPageChanged().

12.1.4.5 nbCloseChatPageBtn

```
std::vector< wxButton* > DimensionlessFrames::ChatFrame::nbCloseChatPageBtn  [private]
```

This vector of raw pointers stores a reference to the wxButtons on each ChatFrame notebook page.

Definition at line 106 of file ChatFrame.h.

Referenced by AddNewChat(), OnBtnSendMessageClick(), and ~ChatFrame().

12.1.4.6 nbNewChatMessageBszrs

```
std::vector< wxBoxSizer* > DimensionlessFrames::ChatFrame::nbNewChatMessageBszrs  [private]
```

This vector of raw pointers stores a reference to the wxBoxSizers on each ChatFrame notebook page.

Definition at line 97 of file ChatFrame.h.

Referenced by AddNewChat().

12.1.4.7 nbSendMessageBtn

```
std::vector< wxButton* > DimensionlessFrames::ChatFrame::nbSendMessageBtn  [private]
```

This vector of raw pointers stores a reference to the wxButtons on each ChatFrame notebook page.

Definition at line 103 of file ChatFrame.h.

Referenced by AddNewChat(), OnBtnSendMessageClick(), and ~ChatFrame().

12.1.4.8 nbTCNewMessage

```
std::vector< wxTextCtrl* > DimensionlessFrames::ChatFrame::nbTCNewMessage  [private]
```

This vector of raw pointers stores a reference to the wxTextCtrls on each ChatFrame notebook page.

Definition at line 100 of file ChatFrame.h.

Referenced by AddNewChat(), and OnBtnSendMessageClick().

12.1.4.9 ParentWindow

```
wxWindow* DimensionlessFrames::ChatFrame::ParentWindow  [private]
```

This raw pointer stores a reference to the parent window of this ChatFrame.

Definition at line 82 of file ChatFrame.h.

Referenced by ChatFrame(), and ~ChatFrame().

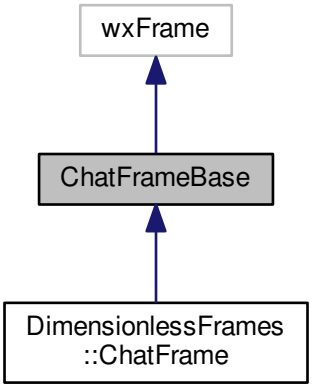
The documentation for this class was generated from the following files:

- DimensionlessClient/ChatFrame.h
- DimensionlessClient/ChatFrame.cpp

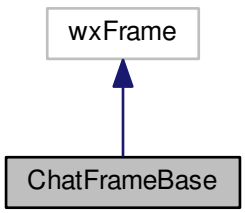
12.2 ChatFrameBase Class Reference

```
#include <Dimensionless.h>
```

Inheritance diagram for ChatFrameBase:



Collaboration diagram for ChatFrameBase:



Public Member Functions

- wxNotebook * GetNbChat ()
- ChatFrameBase (wxWindow *parent, wxWindowID id=wxID_ANY, const wxString &title=_("Chat"), const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(500, 400), long style=wxDEFAULT_FRAM↩ E_STYLE)
- virtual ~ChatFrameBase ()

Protected Member Functions

- virtual void OnNbChatPageChanged (wxNotebookEvent &event)

Protected Attributes

- wxNotebook * nbChat

12.2.1 Detailed Description

Definition at line 256 of file Dimensionless.h.

12.2.2 Constructor & Destructor Documentation

12.2.2.1 ChatFrameBase()

```
ChatFrameBase::ChatFrameBase (
              wxWindow * parent,
              wxWindowID id = wxID_ANY,
              const wxString & title = _("Chat"),
              const wxPoint & pos = wxDefaultPosition,
              const wxSize & size = wxSize(500,400),
              long style = wxDEFAULT_FRAME_STYLE )
```

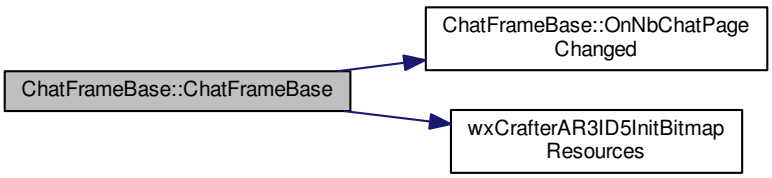
Definition at line 670 of file Dimensionless.cpp.

References bBitmapLoaded, nbChat, OnNbChatPageChanged(), WXC_FROM_DIP, and wxCrafterAR3ID5Init↩ BitmapResources().

```
671     : wxFrame(parent, id, title, pos, size, style)
672 {
673     if ( !bBitmapLoaded ) {
674         // We need to initialise the default bitmap handler
675         wxXmlResource::Get()->AddHandler(new wxBitmapXmlHandler);
676         wxCrafterAR3ID5InitBitmapResources();
677         bBitmapLoaded = true;
678     }
679
680     wxBoxSizer* bszrChat = new wxBoxSizer(wxVERTICAL);
681     this->SetSizer(bszrChat);
682
683     nbChat = new wxNotebook(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)),
      wxBK_DEFAULT);
684     nbChat->SetName(wxT("nbChat"));
685
```

```
686     bszrChat->Add(nbChat, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
687
688
689     #if wxVERSION_NUMBER >= 2900
690     if(!wxPersistenceManager::Get().Find(nbChat)){
691         wxPersistenceManager::Get().RegisterAndRestore(nbChat);
692     } else {
693         wxPersistenceManager::Get().Restore(nbChat);
694     }
695     #endif
696
697     SetName(wxT("ChatFrameBase"));
698     SetSize(500,400);
699     if (GetSizer()) {
700          GetSizer()->Fit(this);
701     }
702     if(GetParent()) {
703         CentreOnParent(wxBOTH);
704     } else {
705         CentreOnScreen(wxBOTH);
706     }
707 #if wxVERSION_NUMBER >= 2900
708     if(!wxPersistenceManager::Get().Find(this)) {
709         wxPersistenceManager::Get().RegisterAndRestore(this);
710     } else {
711         wxPersistenceManager::Get().Restore(this);
712     }
713 #endif
714     // Connect events
715     nbChat->Connect(wxEVT_COMMAND_NOTEBOOK_PAGE_CHANGED, wxNotebookEventHandler(
      ChatFrameBase::OnNbChatPageChanged), NULL, this);
716
717 }
```

Here is the call graph for this function:



12.2.2.2 ~ChatFrameBase()

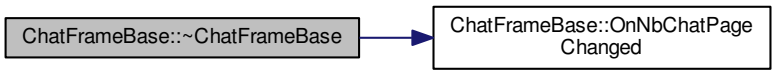
```
ChatFrameBase::~ChatFrameBase ( )  [virtual]
```

Definition at line 719 of file Dimensionless.cpp.

References nbChat, and OnNbChatPageChanged().

```
720 {
721     nbChat->Disconnect(wxEVT_COMMAND_NOTEBOOK_PAGE_CHANGED, wxNotebookEventHandler(
      ChatFrameBase::OnNbChatPageChanged), NULL, this);
722
723 }
```

Here is the call graph for this function:



12.2.3 Member Function Documentation

12.2.3.1 GetNbChat()

```
wxNotebook* ChatFrameBase::GetNbChat ( )  [inline]
```

Definition at line 265 of file Dimensionless.h.

```
265 { return nbChat; }
```

12.2.3.2 OnNbChatPageChanged()

```
virtual void ChatFrameBase::OnNbChatPageChanged (
            wxNotebookEvent & event )  [inline], [protected], [virtual]
```

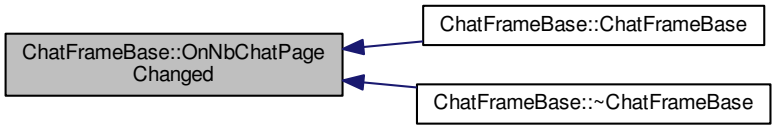
Reimplemented in DimensionlessFrames::ChatFrame.

Definition at line 262 of file Dimensionless.h.

Referenced by ChatFrameBase(), and ~ChatFrameBase().

```
262 { event.Skip(); }
```

Here is the caller graph for this function:



12.2.4 Member Data Documentation

12.2.4.1 nbChat

```
wxNotebook* ChatFrameBase::nbChat  [protected]
```

Definition at line 259 of file Dimensionless.h.

Referenced by DimensionlessFrames::ChatFrame::AddNewChat(), ChatFrameBase(), DimensionlessFrames↩ ::ChatFrame::OnBtnCloseChatPageClick(), DimensionlessFrames::ChatFrame::OnBtnSendMessageClick(), and ∼ChatFrameBase().

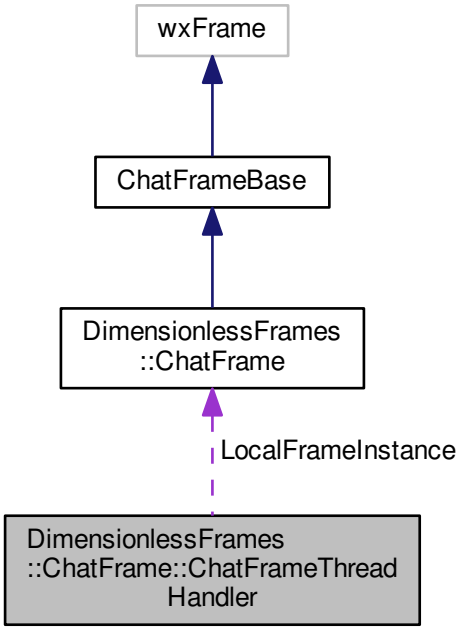
The documentation for this class was generated from the following files:

- DimensionlessClient/Dimensionless.h
- DimensionlessClient/Dimensionless.cpp

12.3 DimensionlessFrames::ChatFrame::ChatFrameThreadHandler Struct Reference

This struct handles events posted from our chatFrameThread.

Collaboration diagram for DimensionlessFrames::ChatFrame::ChatFrameThreadHandler:



Public Member Functions

- ChatFrameThreadHandler (ChatFrame *frame)
- void operator() (wxThreadEvent &event)

Private Attributes

- ChatFrame * LocalFrameInstance = nullptr

  A raw pointer reference to a ChatFrame to handle events for.

12.3.1 Detailed Description

This struct handles events posted from our chatFrameThread.

Definition at line 52 of file ChatFrame.cpp.

12.3.2 Constructor & Destructor Documentation

12.3.2.1 ChatFrameThreadHandler()

```
DimensionlessFrames::ChatFrame::ChatFrameThreadHandler::ChatFrameThreadHandler (
            ChatFrame * frame )  [inline]
```

This constructor initializes a new ChatFrameThreadHandler object with a pointer to the current ChatFrame instance so it may be modified as required.

Parameters

in	*frame*	The raw pointer to a ChatFrame instance.

Definition at line 59 of file ChatFrame.cpp.

```
60              {
61                  // Set the local ChatFrame reference to that passed.
62                  LocalFrameInstance = frame;
63              }
```

12.3.3 Member Function Documentation

12.3.3.1 operator()()

```
void DimensionlessFrames::ChatFrame::ChatFrameThreadHandler::operator() (
            wxThreadEvent & event )  [inline]
```

This function remaps the () operator of ChatFrameThreadHandler to handle chatFrameThread events.

Parameters

in	*event*	A wxThreadEvent object posted by our chatFrameThread.

Definition at line 69 of file ChatFrame.cpp.

```
69                                                  {
70                 switch(event.GetId()){
71                     case wxID_ANY:
72                     {
73                         // Update the ChatFrame chatlog with what is on disk and re-enable other UI
       elements.
74                         LocalFrameInstance->nbChatRTChatLog[event.GetInt()
      ]->LoadFile(LocalFrameInstance->nbChatRTChatLog[event.GetInt()]->
      GetFilename());
75                         LocalFrameInstance->nbSendMessageBtn[event.GetInt
      ()]->Enable(true);
76                         LocalFrameInstance->nbCloseChatPageBtn[event.
      GetInt()]->Enable(true);
77                         LocalFrameInstance->nbChat->Enable(true);
78                         break;
79                     }
80                     default:
81                     {
82                         break;
83                     }
84                 }
85             }
```

12.3.4 Member Data Documentation

12.3.4.1 LocalFrameInstance

```
ChatFrame* DimensionlessFrames::ChatFrame::ChatFrameThreadHandler::LocalFrameInstance = nullptr
[private]
```

A raw pointer reference to a ChatFrame to handle events for.

Definition at line 88 of file ChatFrame.cpp.

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

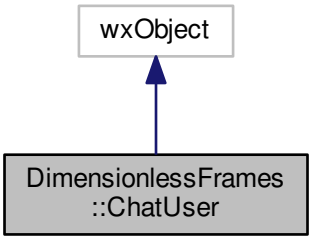
- DimensionlessClient/ChatFrame.cpp

12.4 DimensionlessFrames::ChatUser Class Reference

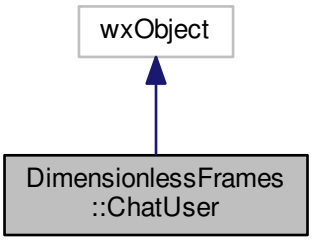
This class stores a reference to a ConnectedUser that the current user can chat with in the ChatFrame.

```
#include <ChatFrame.h>
```

Inheritance diagram for DimensionlessFrames::ChatUser:



Collaboration diagram for DimensionlessFrames::ChatUser:



Public Member Functions

- ChatUser (const int &index, const std::shared_ptr< ConnectedUser > &chatUser)

Public Attributes

- unsigned int targetUserIndex = 0

  This unsigned integer stores the index of the ConnectedUser that can be chatted with.
- std::shared_ptr< ConnectedUser > targetChatUser

  This smart shared pointer stores a reference to a ConnectedUser that can be chatted with.

12.4.1 Detailed Description

This class stores a reference to a ConnectedUser that the current user can chat with in the ChatFrame.

Definition at line 43 of file ChatFrame.h.

12.4.2 Constructor & Destructor Documentation

12.4.2.1 ChatUser()

```
DimensionlessFrames::ChatUser::ChatUser (
            const int & index,
            const std::shared_ptr< ConnectedUser > & chatUser )
```

This constructor creates a new ChatUser object that holds the current index of the ConnectedUser the object references as well as a reference to the ConnectedUser object.

Parameters

in	*index*	The index of a ConnectedUser object reference.
in	*chatUser*	A smart shared pointer reference to a ConnectedUser instance.

Definition at line 45 of file ChatFrame.cpp.

References targetChatUser, and targetUserIndex.

```
45                                                                          : wxObject()
46     {
47         this->targetUserIndex = index;
48         this->targetChatUser = chatUser;
49     };
```

12.4.3 Member Data Documentation

12.4.3.1 targetChatUser

```
std::shared_ptr<ConnectedUser> DimensionlessFrames::ChatUser::targetChatUser
```

This smart shared pointer stores a reference to a ConnectedUser that can be chatted with.

Definition at line 53 of file ChatFrame.h.

Referenced by ChatUser(), and DimensionlessFrames::ChatFrame::OnBtnSendMessageClick().

12.4.3.2 targetUserIndex

```
unsigned int DimensionlessFrames::ChatUser::targetUserIndex = 0
```

This unsigned integer stores the index of the ConnectedUser that can be chatted with.

Definition at line 50 of file ChatFrame.h.

Referenced by ChatUser(), DimensionlessFrames::ChatFrame::OnBtnCloseChatPageClick(), and DimensionlessFrames::ChatFrame::OnBtnSendMessageClick().

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

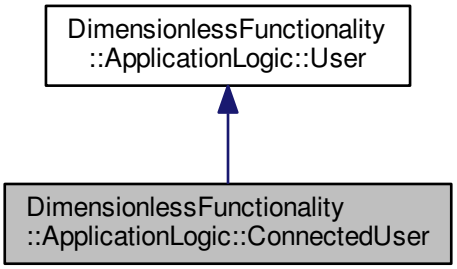
- DimensionlessClient/ChatFrame.h
- DimensionlessClient/ChatFrame.cpp

12.5 DimensionlessFunctionality::ApplicationLogic::ConnectedUser Class Reference

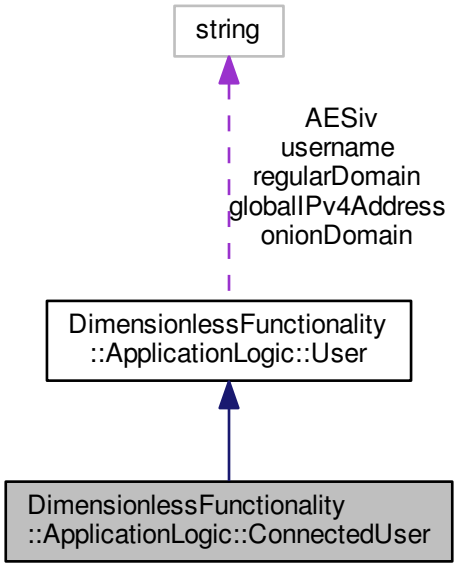
This class denotes a user that is connected to the static CurrentUser that is logged into the application.

```
#include <DimensionlessFunctionality.h>
```

Inheritance diagram for DimensionlessFunctionality::ApplicationLogic::ConnectedUser:



Collaboration diagram for DimensionlessFunctionality::ApplicationLogic::ConnectedUser:



Public Member Functions

- ConnectedUser (const string &aesiv, const string &newusername, const string &globalipv4address, const string &oniondomain, const string &regulardomain)
- ~ConnectedUser ()

Additional Inherited Members

12.5.1 Detailed Description

This class denotes a user that is connected to the static CurrentUser that is logged into the application.

Definition at line 823 of file DimensionlessFunctionality.h.

12.5.2 Constructor & Destructor Documentation

12.5.2.1 ConnectedUser()

```
DimensionlessFunctionality::ApplicationLogic::ConnectedUser::ConnectedUser (
            const string & aesiv,
            const string & newusername,
            const string & globalipv4address,
            const string & oniondomain,
            const string & regulardomain )
```

This constructor initializes a new ConnectedUser object with the variable values we want. The ConnectedUser object denotes a user connected to the currently logged in user.

Parameters

in	*aesiv*	The AES IV of the ConnectedUser object to be created.
in	*newusername*	The username of the ConnectedUser object to be created.
in	*globalipv4address*	The IPv4 address of the ConnectedUser object to be created.
in	*oniondomain*	The onion domain of the ConnectedUser object to be created.
in	*regulardomain*	The regular FreeDNS domain of the ConnectedUser object to be created.

Definition at line 1880 of file DimensionlessFunctionality.cpp.

```
1881         : User(aesiv,newusername,globalipv4address,oniondomain,regulardomain)
1882         {
1883         }
```

12.5.2.2 ~ConnectedUser()

```
DimensionlessFunctionality::ApplicationLogic::ConnectedUser::~ConnectedUser ( )
```

This destructor currently does nothing.

Definition at line 1888 of file DimensionlessFunctionality.cpp.

```
1889         {
1890         }
```

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

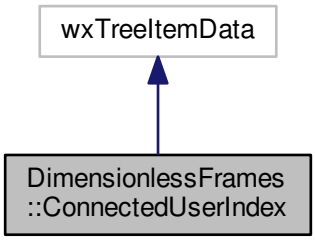
- DimensionlessClient/DimensionlessFunctionality.h
- DimensionlessClient/DimensionlessFunctionality.cpp

12.6 DimensionlessFrames::ConnectedUserIndex Class Reference

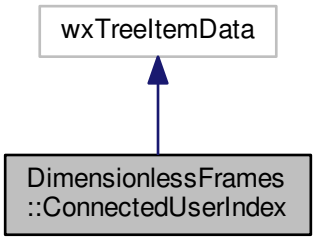
This class stores a reference to a ConnectedUser that the list of ConnectedUsers in the MainFrame.

```
#include <MainFrame.h>
```

Inheritance diagram for DimensionlessFrames::ConnectedUserIndex:



Collaboration diagram for DimensionlessFrames::ConnectedUserIndex:



Public Member Functions

- ConnectedUserIndex (const int &index)

Public Attributes

- unsigned int targetUserIndex = 0

  This variable stores the index of the ConnectedUser object this class instance references.

12.6.1 Detailed Description

This class stores a reference to a ConnectedUser that the list of ConnectedUsers in the MainFrame.

Definition at line 59 of file MainFrame.h.

12.6.2 Constructor & Destructor Documentation

12.6.2.1 ConnectedUserIndex()

```
DimensionlessFrames::ConnectedUserIndex::ConnectedUserIndex (
            const int & index )
```

This constructor creates a new ConnectedUserIndex object that holds the current index of the ConnectedUser the object references.

Parameters

in	*index*	The index of a ConnectedUser object reference.

Definition at line 60 of file MainFrame.cpp.

References targetUserIndex.

```
60                                                                  : wxTreeItemData()
61     {
62        this->targetUserIndex = index;
63     };
```

12.6.3 Member Data Documentation

12.6.3.1 targetUserIndex

```
unsigned int DimensionlessFrames::ConnectedUserIndex::targetUserIndex = 0
```

This variable stores the index of the ConnectedUser object this class instance references.

Definition at line 66 of file MainFrame.h.

Referenced by ConnectedUserIndex(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRight↩ Click().

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

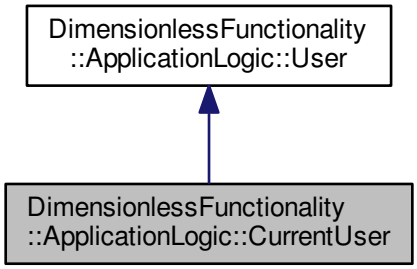
- DimensionlessClient/MainFrame.h
- DimensionlessClient/MainFrame.cpp

12.7 DimensionlessFunctionality::ApplicationLogic::CurrentUser Class Reference

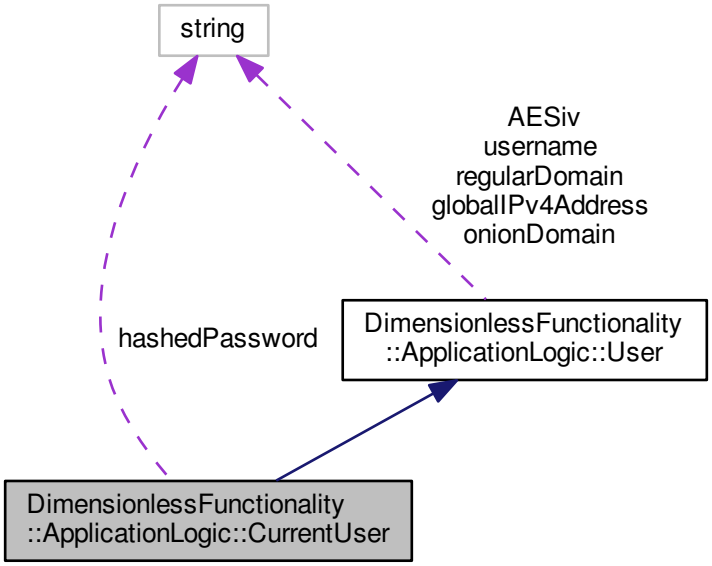
This class denotes the User that is currently logged into the application.

```
#include <DimensionlessFunctionality.h>
```

Inheritance diagram for DimensionlessFunctionality::ApplicationLogic::CurrentUser:



Collaboration diagram for DimensionlessFunctionality::ApplicationLogic::CurrentUser:



Public Member Functions

- CurrentUser (const string &aesiv, const string &newusername, const string &globalipv4address, const string &oniondomain, const string &regulardomain)
- ~CurrentUser ()
- string toDataString (const string &status="request")

Public Attributes

- string hashedPassword = ""

  This string stores the current user's password after being hashed by SHA512.

Additional Inherited Members

12.7.1 Detailed Description

This class denotes the User that is currently logged into the application.

Definition at line 833 of file DimensionlessFunctionality.h.

12.7.2 Constructor & Destructor Documentation

12.7.2.1 CurrentUser()

```
DimensionlessFunctionality::ApplicationLogic::CurrentUser::CurrentUser (
            const string & aesiv,
            const string & newusername,
            const string & globalipv4address,
            const string & oniondomain,
            const string & regulardomain )
```

This constructor initializes a new CurrentUser object with the variable values we want. The CurrentUser object denotes the currently logged in user to the application.

Parameters

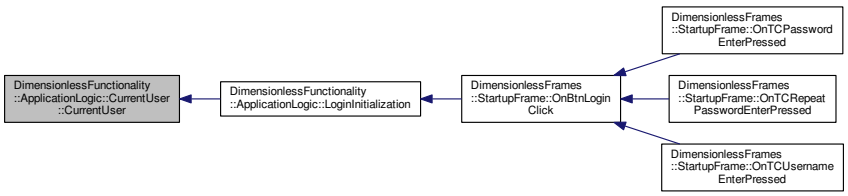
in	*aesiv*	The AES IV of the CurrentUser object to be created.
in	*newusername*	The username of the CurrentUser object to be created.
in	*globalipv4address*	The IPv4 address of the CurrentUser object to be created.
in	*oniondomain*	The onion domain of the CurrentUser object to be created.
in	*regulardomain*	The regular FreeDNS domain of the CurrentUser object to be created.

Definition at line 1900 of file DimensionlessFunctionality.cpp.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization().

```
1901         : User(aesiv,newusername,globalipv4address,oniondomain,regulardomain)
1902         {
1903         }
```

Here is the caller graph for this function:



12.7.2.2 ~CurrentUser()

```
DimensionlessFunctionality::ApplicationLogic::CurrentUser::~CurrentUser ( )
```

This destructor currently does nothing.

Definition at line 1908 of file DimensionlessFunctionality.cpp.

```
1909         {
1910         }
```

12.7.3 Member Function Documentation

12.7.3.1 toDataString()

```
string DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString (
            const string & status = "request" )
```

This function provides a single string containing a User object's username, AES IV, IPv4 address, onion domain, regular domain, last page updated time, platform, status, public search visibility and tor option in a comma separated format.

Returns

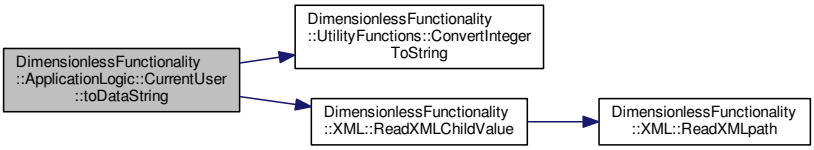
A comma separated string containing the data required for a request to be sent by PHP for our currently logged in user.

Definition at line 1916 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::connectedUsers, DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::currentUser, DimensionlessFunctionality::ApplicationLogic::User::globalIPv4Address, DimensionlessFunctionality::ApplicationLogic::User::onionDomain, DimensionlessFunctionality::XML::ReadXMLChildValue(), DimensionlessFunctionality::ApplicationLogic::User::regularDomain, and DimensionlessFunctionality::ApplicationLogic::User::username.

```
1917         {
1918             string lastpageupdatedtime = XML::ReadXMLChildValue("./Users/"+this->
      username+"/data/"+this->username+".xml", "User", "LastPageUpdatedTime");
1919             string platform = XML::ReadXMLChildValue("./Users/"+this->
      username+"/data/"+this->username+".xml","User", "Platform");
1920             string publicsearchvisibility = XML::ReadXMLChildValue("./Users/"+this->
      username+"/data/"+this->username+".xml", "User", "PublicSearchVisibility");
1921             string tor = XML::ReadXMLChildValue("./Users/"+this->
      username+"/data/"+this->username+".xml", "User", "Tor");
1922 
1923             return UtilityFunctions::ConvertIntegerToString(
      username.length()) + "," +
1924                    UtilityFunctions::ConvertIntegerToString(
      globalIPv4Address.length()) + "," +
1925                    UtilityFunctions::ConvertIntegerToString(
      onionDomain.length()) + "," +
1926                    UtilityFunctions::ConvertIntegerToString(
      regularDomain.length()) + "," +
1927                    UtilityFunctions::ConvertIntegerToString(
      lastpageupdatedtime.length()) + "," +
1928                    UtilityFunctions::ConvertIntegerToString(
      platform.length()) + "," +
1929                    UtilityFunctions::ConvertIntegerToString(status.
      length()) + "," +
1930                    UtilityFunctions::ConvertIntegerToString(
      publicsearchvisibility.length()) + "," +
1931                    UtilityFunctions::ConvertIntegerToString(tor.
      length()) + "," +
1932                    username + "," +
1933                    globalIPv4Address + "," +
1934                    onionDomain + "," +
1935                    regularDomain + "," +
1936                    lastpageupdatedtime + "," +
1937                    platform + "," +
1938                    status + "," +
1939                    publicsearchvisibility + "," +
1940                    tor;
1941         }
```

Here is the call graph for this function:



12.7.4 Member Data Documentation

12.7.4.1 hashedPassword

```
string DimensionlessFunctionality::ApplicationLogic::CurrentUser::hashedPassword = ""
```

This string stores the current user's password after being hashed by SHA512.

Definition at line 842 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization().

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

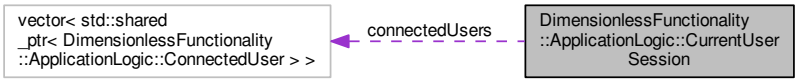
- DimensionlessClient/DimensionlessFunctionality.h
- DimensionlessClient/DimensionlessFunctionality.cpp

12.8 DimensionlessFunctionality::ApplicationLogic::CurrentUserSession Class Reference

This singleton class stores a smart shared pointer reference to the currently logged in user and a collection of their connected users.

```
#include <DimensionlessFunctionality.h>
```

Collaboration diagram for DimensionlessFunctionality::ApplicationLogic::CurrentUserSession:



Public Member Functions

- CurrentUserSession (CurrentUserSession const &)=delete

  The constructor to our singleton class referencing another instance is deleted.
- void operator= (CurrentUserSession const &)=delete

  This prevents our singleton class from having another class instance assigned to it.

Static Public Attributes

- static std::shared_ptr< CurrentUser > currentUser

  This smart shared pointer stores a reference to the currently logged in user's object.
- static vector< std::shared_ptr< ConnectedUser > > connectedUsers

  This vector of smart references to ConnectedUsers stores a collection of the users connected to our currentUser.

12.8.1 Detailed Description

This singleton class stores a smart shared pointer reference to the currently logged in user and a collection of their connected users.

Definition at line 849 of file DimensionlessFunctionality.h.

12.8.2 Constructor & Destructor Documentation

12.8.2.1 CurrentUserSession()

```
DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::CurrentUserSession (
            CurrentUserSession const &  )  [delete]
```

The constructor to our singleton class referencing another instance is deleted.

12.8.3 Member Function Documentation

12.8.3.1 operator=()

```
void DimensionlessFunctionality::ApplicationLogic::CurrentUserSession::operator= (
            CurrentUserSession const &  )  [delete]
```

This prevents our singleton class from having another class instance assigned to it.

12.8.4 Member Data Documentation

12.8.4.1 connectedUsers

```
vector< std::shared_ptr< ConnectedUser > > DimensionlessFunctionality::ApplicationLogic::↩
CurrentUserSession::connectedUsers  [static]
```

This vector of smart references to ConnectedUsers stores a collection of the users connected to our currentUser.

A vector of smart shared pointers that store references to a collection of ConnectedUsers that denote users which are connected and related to the currently logged in user.

Definition at line 855 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent(), DimensionlessFunctionality::Networking::Libtorrent::CreateTorrent(), DimensionlessFunctionality::Networking::Nmap::ScanIPAddressesWithPort(), and DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString().

12.8.4.2 currentUser

```
std::shared_ptr< CurrentUser > DimensionlessFunctionality::ApplicationLogic::CurrentUser↩
Session::currentUser  [static]
```

This smart shared pointer stores a reference to the currently logged in user's object.

A smart shared pointer that stores the reference to a CurrentUser object that denotes our currently logged in user.

Definition at line 852 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::Networking::Libtorrent::CreateTorrentFileFilter(), DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving(), DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry(), and DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString().

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

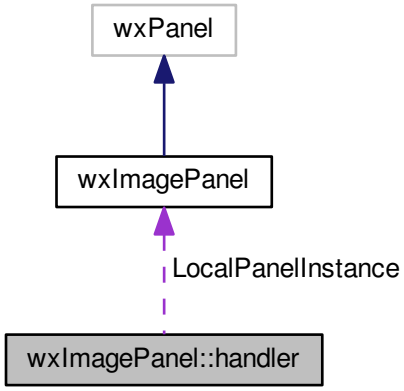
- DimensionlessClient/DimensionlessFunctionality.h
- DimensionlessClient/DimensionlessFunctionality.cpp

12.9 wxImagePanel::handler Struct Reference

This functor allows for events on the wxImagePanel to be handled.

```
#include <DimensionlessControls.h>
```

Collaboration diagram for wxImagePanel::handler:



Public Member Functions

- handler (wxImagePanel *pnl)
- void operator() (wxEvent &evt)

Private Attributes

- wxImagePanel * LocalPanelInstance

  Store the panel object to handle locally.

12.9.1 Detailed Description

This functor allows for events on the wxImagePanel to be handled.

Definition at line 76 of file DimensionlessControls.h.

12.9.2 Constructor & Destructor Documentation

12.9.2.1 handler()

```
wxImagePanel::handler::handler (
            wxImagePanel * pnl )
```

Description

Functor constructor assigns a passed wxImagePanel reference to be accessed later within the functor operator() overload.

Functor Constructor Body

Parameters

in	*pnl*	A reference to a wxImagePanel that is used later within the functor operator() overload in response to events.

Definition at line 99 of file DimensionlessControls.cpp.

```
99                                                  {
101     // To be used later on in the operator() overload.
102     LocalPanelInstance = pnl;
104 }
```

12.9.3 Member Function Documentation

12.9.3.1 operator()()

```
void wxImagePanel::handler::operator() (
            wxEvent & evt )
```

Description

Functor operator() overload that handles events on the LocalPanelInstance reference of a wxImagePanel.

Functor Operator() Overload Body

Parameters

in	*evt*	A wxEvent object that has recently been sent as its event has been triggered for the wxImagePanel binded (Captures all to most generic events).

Returns

Nothing, this functor solely performs tasks in response to events.

Definition at line 114 of file DimensionlessControls.cpp.

```
114                                                 {
116     // Stop processing events if the bitmap object to use is not available.
117     if( LocalPanelInstance->image.IsOk() )
118     {
119         // This event occurs when the wxImagePanel is to render itself to be displayed.
120         if( evt.GetEventType() == wxEVT_PAINT )
121         {
122             // Set the device context to the wxImagePanel passed to the handler (in order to allow graphics
      and text to be drawn onto the wxImagePanel).
123             wxPaintDC dc( LocalPanelInstance );
124 
125             // Retrieve the size of the device context (effectively the size of the wxImagePanel).
126             int neww, newh;
127             dc.GetSize( &neww, &newh );
128 
129             // Resize the panel background if and only if the size of the panel has changed.
130             if( neww != LocalPanelInstance->w || newh !=
      LocalPanelInstance->h )
131             {
132                 // Check to ensure that the bitmap about to be drawn is valid, exit this event handler.
133                 if( LocalPanelInstance->image.IsOk() == false )
134                 {
135                     evt.Skip();
136                     return;
137                 }
138 
139                 // Update the original size with the new one to decide if the size has changed in the
      future.
140                 LocalPanelInstance->w = neww;
141                 LocalPanelInstance->h = newh;
142             }
143 
144             // Set a white background on the panel.
145             dc.SetBackgroundMode( wxPENSTYLE_SOLID );
146             dc.SetBackground( *wxWHITE_BRUSH );
147             dc.Clear();
148 
149             // Proceed to draw the background bitmap which always remains central behind the login box.
150             dc.DrawBitmap( LocalPanelInstance->image, ( neww -
      LocalPanelInstance->image.GetWidth() ) / 2, ( newh -
      LocalPanelInstance->image.GetHeight() ) / 2, false );
151         }
152         // This event occurs when the wxImagePanel is resized.
153         else if ( evt.GetEventType() == wxEVT_SIZE )
154         {
155             // Force a redraw of the panel since a change in size has occurred.
156             LocalPanelInstance->Refresh();
157 
158             // Skip anything that the original event may execute as the wxImagePanel implements wxPanel.
159             evt.Skip();
160         }
161     }
163 }
```

12.9.4 Member Data Documentation

12.9.4.1 LocalPanelInstance

```
wxImagePanel* wxImagePanel::handler::LocalPanelInstance  [private]
```

Store the panel object to handle locally.

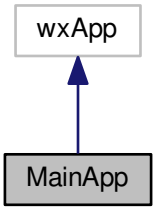
Definition at line 87 of file DimensionlessControls.h.

The documentation for this struct was generated from the following files:

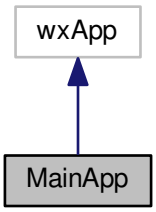
- DimensionlessClient/DimensionlessControls.h
- DimensionlessClient/DimensionlessControls.cpp

12.10 MainApp Class Reference

Inheritance diagram for MainApp:



Collaboration diagram for MainApp:



Public Member Functions

- MainApp ()

  Empty constructor.
- virtual ~MainApp ()

  Empty destructor.
- virtual bool OnInit ()

  This functition sets up all the necessary features required for our application and starts up a StartupFrame instance.

12.10.1 Detailed Description

The MainApp class starts the Dimensionless application.

Definition at line 44 of file main.cpp.

12.10.2 Constructor & Destructor Documentation

12.10.2.1 MainApp()

```
MainApp::MainApp ( )  [inline]
```

Empty constructor.

Definition at line 48 of file main.cpp.

```
48 {}
```

12.10.2.2 ~MainApp()

```
virtual MainApp::~MainApp ( )  [inline], [virtual]
```

Empty destructor.

Definition at line 50 of file main.cpp.

```
50 {}
```

12.10.3 Member Function Documentation

12.10.3.1 OnInit()

```
virtual bool MainApp::OnInit ( )  [inline], [virtual]
```

This functition sets up all the necessary features required for our application and starts up a StartupFrame instance.

Definition at line 53 of file main.cpp.

```
54     {
55         // Add the common image handlers. Call wxInitAllImageHandlers(); for all handlers.
56         wxImage::AddHandler( new wxPNGHandler );
57         wxImage::AddHandler( new wxJPEGHandler );
58 
59         // Try to clear GDB error from terminal when debugging in CodeLite.
60         std::flush(std::cout);
61 
62         // Initialize logging.
63         wxLog* logger = new wxLogStream(&std::cout);
64         wxLog::SetActiveTarget(logger);
65         wxLogMessage(_(ApplicationInfo::GetExecutablePath()));
66         wxLogMessage(_(ApplicationInfo::GetArchitecture()));
67         wxLogMessage(_(ApplicationInfo::GetCompilerVersion()));
68         wxLogMessage(_(ApplicationInfo::GetDocumentsDirectory()));
69 
70         // Initialize a StartupFrame instance.
71         StartupFrame *startupFrame = new StartupFrame(NULL);
72 
73         // Set our instance on top.
74         SetTopWindow(startupFrame);
75 
76         // Show our StartupFrame instance.
77         return GetTopWindow()->Show();
78     }
```

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

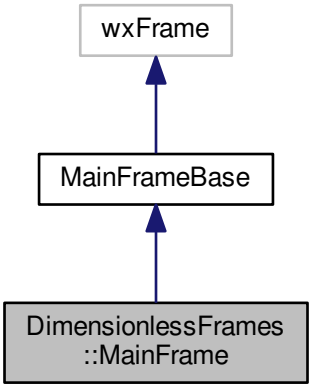
- DimensionlessClient/main.cpp

12.11 DimensionlessFrames::MainFrame Class Reference

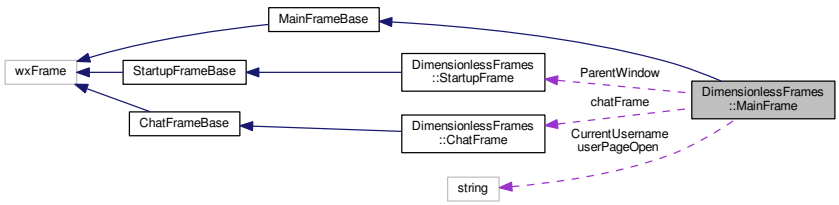
This class creates a MainFrame that is displayed after a user logs in.

`#include <MainFrame.h>`

Inheritance diagram for DimensionlessFrames::MainFrame:



Collaboration diagram for DimensionlessFrames::MainFrame:



Classes

- struct MainFrameThreadHandler

  This struct handles events posted from our mainFrameThread.

Public Member Functions

- MainFrame (StartupFrame *parent, const std::string &username)
- virtual ~MainFrame ()

Static Public Attributes

- static ChatFrame * chatFrame = nullptr

  This variable stores a raw pointer ChatFrame reference to show when the current user wants to chat with a connected user.

Protected Member Functions

- virtual void OnTcConnectedUsersContextMenuClick (wxCommandEvent &event)
- virtual void OnTcConnectedUsersItemRightClick (wxTreeEvent &event)
- virtual void OnBtnOpenPersonalAdminWebsiteClick (wxCommandEvent &event)
- virtual void OnBtnPublishUpdatedWebsiteClick (wxCommandEvent &event)
- virtual void OnBtnSearchForUsersClick (wxCommandEvent &event)
- virtual void OnBtnLogOutClick (wxCommandEvent &event)
- virtual void OnSettings (wxCommandEvent &event)
- virtual void OnAbout (wxCommandEvent &event)
- virtual void OnClose (wxCloseEvent &event)
- virtual void OnQuit (wxCommandEvent &event)

Private Member Functions

- void UpdateConnectedUsers ()

Private Attributes

- std::string CurrentUsername = ""

  This variable stores the username of the currently logged in user.
- bool LogOut = false

  This variable decides if it is time to log out.
- StartupFrame * ParentWindow

  This variable stores the parent windows of this MainFrame instance.

Static Private Attributes

- static std::thread mainFrameThread

  This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.
- static int currentTargetUserIndex = 0

  This variable stores the index of the connected user the current users wishes to chat with.
- static wxMenu * menu

  This variable stores a raw pointer to the wxMenu that is displayed upon right clicking an item in the list of connected users.
- static wxTreeItemId rootTcConnectedUsersNode

  This variable stores the id of the root node in our ConnectedUsers list.
- static string userPageOpen = ""

  This variable stores the username of the user whose page is currently open.

Additional Inherited Members

12.11.1 Detailed Description

This class creates a MainFrame that is displayed after a user logs in.

Definition at line 70 of file MainFrame.h.

12.11.2 Constructor & Destructor Documentation

12.11.2.1 MainFrame()

```
DimensionlessFrames::MainFrame::MainFrame (
            StartupFrame * parent,
            const std::string & username )
```

This constructor initializes a new MainFrame.

Parameters

in	*parent*	The parent window of the one to be created.
in	*username*	The username of the currently logged in user.

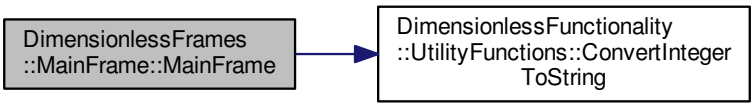
Definition at line 107 of file MainFrame.cpp.

References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), CurrentUsername, ParentWindow, rootTcConnectedUsersNode, and MainFrameBase::tcConnectedUsers.

```
107                                                                     :
      MainFrameBase(parent)
108     {
109         this->ParentWindow = parent;
110         this->CurrentUsername = username;
111         this->SetTitle("Dimensionless - " + username);
112         this->SetMinClientSize(wxSize(600,600));
113         this->SetSize(600,600);
114         this->CentreOnParent(wxBOTH);
115 
116         int onlineUsers = 0;
117         int offlineUsers = 0;
118         int rejectedUsers = 0;
119         int requestUsers = 0;
120         rootTcConnectedUsersNode = this->
      tcConnectedUsers->AddRoot("Connected Users");
121         wxTreeItemId onlineNode = this->tcConnectedUsers->AppendItem(
      rootTcConnectedUsersNode, "Online");
122         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
123             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::online){
124                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(onlineNode,
      CurrentUserSession::connectedUsers[i]->username);
125                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
      ConnectedUserIndex(i));
126                 onlineUsers++;
127             }
128         }
129         this->tcConnectedUsers->SetItemText(onlineNode, "Online ("+
      ConvertIntegerToString(onlineUsers)+" users)");
130         wxTreeItemId offlineNode = this->tcConnectedUsers->AppendItem(
      rootTcConnectedUsersNode, "Offline");
```

```
131         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
132             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::offline){
133                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(offlineNode,
    CurrentUserSession::connectedUsers[i]->username);
134                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
135                 offlineUsers++;
136             }
137         }
138         this->tcConnectedUsers->SetItemText(offlineNode, "Offline ("+
    ConvertIntegerToString(offlineUsers)+" users)");
139         wxTreeItemId rejectedNode = this->tcConnectedUsers->AppendItem(
    rootTcConnectedUsersNode, "Rejected");
140         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
141             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::rejected){
142                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(rejectedNode,
    CurrentUserSession::connectedUsers[i]->username);
143                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
144                 rejectedUsers++;
145             }
146         }
147         this->tcConnectedUsers->SetItemText(rejectedNode, "Rejected ("+
    ConvertIntegerToString(rejectedUsers)+" users)");
148         wxTreeItemId connectionRequestsNode = this->tcConnectedUsers->AppendItem(
    rootTcConnectedUsersNode, "Connection Requests");
149         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
150             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::offline){
151                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(
    connectionRequestsNode, CurrentUserSession::connectedUsers[i]->username);
152                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
153                 requestUsers++;
154             }
155         }
156         this->tcConnectedUsers->SetItemText(connectionRequestsNode, "Connection Requests ("
    +ConvertIntegerToString(requestUsers)+" requests)");
157         this->tcConnectedUsers->ExpandAll();
158     }
```

Here is the call graph for this function:



12.11.2.2 ~MainFrame()

```
DimensionlessFrames::MainFrame::~MainFrame ( )  [virtual]
```

This destructor currently does nothing.

Definition at line 163 of file MainFrame.cpp.

```
164     {
165     }
```

12.11.3 Member Function Documentation

12.11.3.1 OnAbout()

```
void DimensionlessFrames::MainFrame::OnAbout (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "About" item is clicked on a MainFrame's header menu.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "About" option is selected under the "Help" header menu.

Reimplemented from MainFrameBase.

Definition at line 181 of file MainFrame.cpp.

```
182     {
183         wxAboutDialogInfo* info = new wxAboutDialogInfo();
184         info->AddDeveloper(_("Eclectic Diemensions - http://eclecticdimensions.com"));
185         info->SetCopyright(wxString::FromUTF8("© Eclectic Dimensions"));
186         info->SetDescription(_("A fully decentralized social platform."));
187         info->SetName(_("Dimensionless"));
188         info->SetWebSite(_("http://eclecticdimensions.com"),_("Eclectic Dimensions Website"));
189         wxAboutBox(*info);
190     }
```

12.11.3.2 OnBtnLogOutClick()

```
void DimensionlessFrames::MainFrame::OnBtnLogOutClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Log Out" button is clicked on a MainFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Log Out" button is clicked.

Reimplemented from MainFrameBase.

Definition at line 220 of file MainFrame.cpp.

References LogOut, and ParentWindow.

```
221     {
222         LogOut = true;
223         this->ParentWindow->Show();
224         this->Close(true);
225     }
```

12.11.3.3 OnBtnOpenPersonalAdminWebsiteClick()

```
void DimensionlessFrames::MainFrame::OnBtnOpenPersonalAdminWebsiteClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Open Personal Admin Website" button is clicked on a MainFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Open Personal Admin Website" button is clicked.

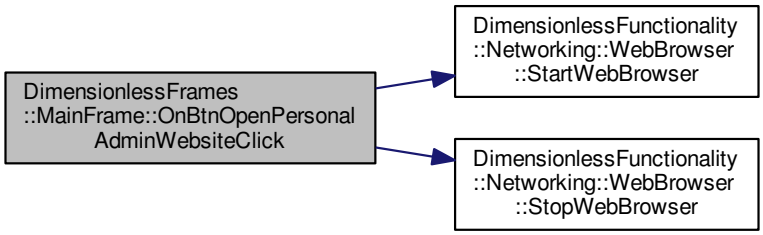
Reimplemented from MainFrameBase.

Definition at line 231 of file MainFrame.cpp.

References DimensionlessFunctionality::Networking::WebBrowser::StartWebBrowser(), and DimensionlessFunctionality::Networking::WebBrowser::StopWebBrowser().

```
232     {
233         StopWebBrowser();
234         StartWebBrowser();
235     }
```

Here is the call graph for this function:



12.11.3.4 OnBtnPublishUpdatedWebsiteClick()

```
void DimensionlessFrames::MainFrame::OnBtnPublishUpdatedWebsiteClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Publish Updated Website" button is clicked on a MainFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Publish Updated Website" button is clicked.

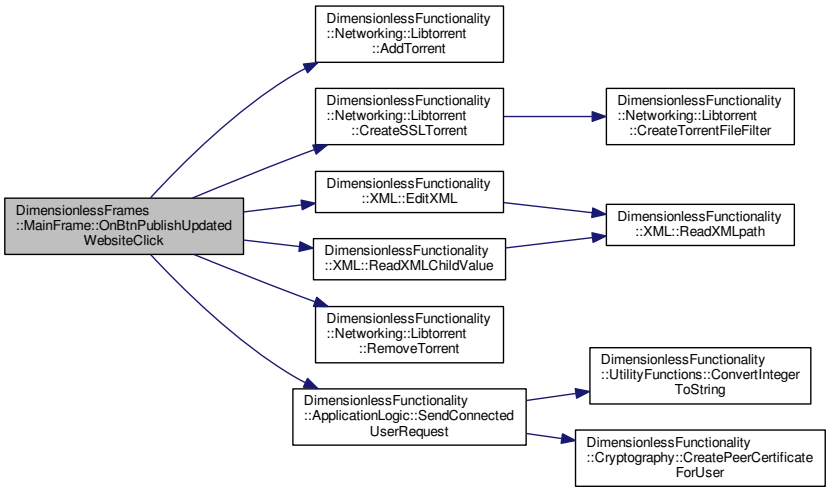
Reimplemented from MainFrameBase.

Definition at line 241 of file MainFrame.cpp.

References DimensionlessFunctionality::Networking::Libtorrent::AddTorrent(), DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent(), DimensionlessFunctionality::XML::EditXML(), DimensionlessFunctionality::XML::ReadXMLChildValue(), DimensionlessFunctionality::Networking::Libtorrent::RemoveTorrent(), and DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest().

```
242     {
243         /*if(!UtilityFunctions::CheckFileExists(ApplicationInfo::GetExecutablePath()+"Users/
    "+targetUsername+"/"+targetUsername+".torrent")){
244 
     DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent("test",XML::ReadXMLChildValue("./Users/"+targetUse
245                                                                                                         ,"./
    Users/"+targetUsername+"/CA/cacert.pem");
246         }*/
247         boost::posix_time::ptime current_date_microseconds = boost::posix_time::microsec_clock::local_time(
    );
248 
249         long milliseconds = current_date_microseconds.time_of_day().total_milliseconds();
250 
251         boost::posix_time::time_duration current_time_milliseconds = boost::posix_time::milliseconds(
    milliseconds);
252 
253         boost::posix_time::ptime current_date_milliseconds(current_date_microseconds.date(),
    current_time_milliseconds);
254 
255         //boost::posix_time::time_from_string();
256         EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
    CurrentUserSession::currentUser->username+".xml", "User", "LastPageUpdatedTime", boost::posix_time::to_simple_string(
    current_date_milliseconds));
257         RemoveTorrent(CurrentUserSession::currentUser->username);
258         boost::filesystem::remove("./Users/"+CurrentUserSession::currentUser->username+"/"+
    CurrentUserSession::currentUser->username+".torrent");
259         DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent
    (CurrentUserSession::currentUser->username,ReadXMLChildValue("./Users/"+
    CurrentUserSession::currentUser->username+"/data/"+CurrentUserSession::currentUser->username+".xml","User","AESIV"),"./U
    CurrentUserSession::currentUser->username+"/CA/cacert.pem");
260         AddTorrent("./Users/"+CurrentUserSession::currentUser->username+"/"+
    CurrentUserSession::currentUser->username+".torrent","./Users/"+CurrentUserSession::currentUser->username+"/CA/cacert.pe
    ./Users/"+CurrentUserSession::currentUser->username+"/CA/private/cakey.pem","./Users/"+
    CurrentUserSession::currentUser->username+"/CA/dhparams.pem",CurrentUserSession::currentUser->hashedPassword);
261         for(std::shared_ptr<ConnectedUser>& user : CurrentUserSession::connectedUsers){
262             if(user->currentStatus == ConnectedUser::Status::online){
263                 SendConnectedUserRequest(Request::SetOurXML, user, "online");
264                 SendConnectedUserRequest(Request::SetOurTorrent, user);
265             }
266         }
267     }
```

Here is the call graph for this function:



12.11.3.5 OnBtnSearchForUsersClick()

```
void DimensionlessFrames::MainFrame::OnBtnSearchForUsersClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Search For Users" button is clicked on a MainFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Search For Users" button is clicked.

Reimplemented from MainFrameBase.

Definition at line 273 of file MainFrame.cpp.

```
274     {
275         SearchForUsersFrame* searchForUsersFrame = new SearchForUsersFrame(this);
276         searchForUsersFrame->Show();
277     }
```

12.11.3.6 OnClose()

```
void DimensionlessFrames::MainFrame::OnClose (
            wxCloseEvent & event )  [protected], [virtual]
```

This function manages what happens after the MainFrame is closed.

Parameters

in	*event*	A wxCloseEvent object that is posted when the MainFrame must be closed.

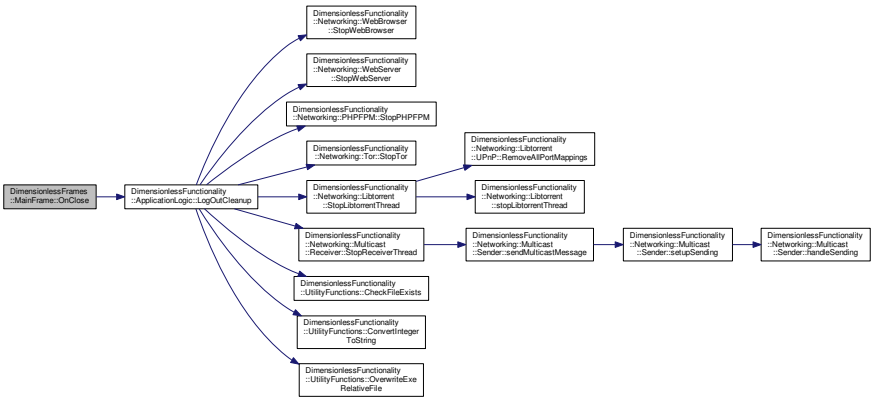
Reimplemented from MainFrameBase.

Definition at line 196 of file MainFrame.cpp.

References CurrentUsername, LogOut, DimensionlessFunctionality::ApplicationLogic::LogOutCleanup(), and ParentWindow.

```
197     {
198         LogOutCleanup(this->CurrentUsername);
199 
200         if(LogOut == false){
201             this->ParentWindow->Close(true);
202         }
203 
204         this->Destroy();
205     }
```

Here is the call graph for this function:



12.11.3.7 OnQuit()

```
void DimensionlessFrames::MainFrame::OnQuit (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Quit" item is clicked on a MainFrame's header menu.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Quit" option is selected under the "File" header menu.

Reimplemented from MainFrameBase.

Definition at line 211 of file MainFrame.cpp.

```
212     {
213         this->Close(true);
214     }
```

12.11.3.8 OnSettings()

```
void DimensionlessFrames::MainFrame::OnSettings (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Settings" item is clicked on a MainFrame's header menu.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Settings" option is selected under the "File" header menu.

Reimplemented from MainFrameBase.

Definition at line 171 of file MainFrame.cpp.

```
172     {
173         SettingsFrame* settingsFrame = new SettingsFrame(this);
174         settingsFrame->Show();
175     }
```

12.11.3.9 OnTcConnectedUsersContextMenuClick()

```
void DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after a context menu item is clicked over a ConnectedUser in our list.

Parameters

in	*event*	A wxCommandEvent object that is posted after a context menu item is clicked.

Definition at line 333 of file MainFrame.cpp.

References DimensionlessFrames::ChatFrame::AddNewChat(), DimensionlessFunctionality::Networking::↩
Libtorrent::AddTorrent(), chatFrame, DimensionlessFunctionality::UtilityFunctions::CheckFileExists(), Dimensionless↩
Functionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::Networking::Libtorrent↩
::currentSession, idContextMenuAcceptConnectionRequest, idContextMenuRejectConnectionRequest, id↩
ContextMenuSendMessage, idContextMenuViewSiteOffline, idContextMenuViewSiteOnline, menu, Dimensionless↩
Functionality::UtilityFunctions::OverwriteExeRelativeFile(), DimensionlessFunctionality::ApplicationLogic::↩

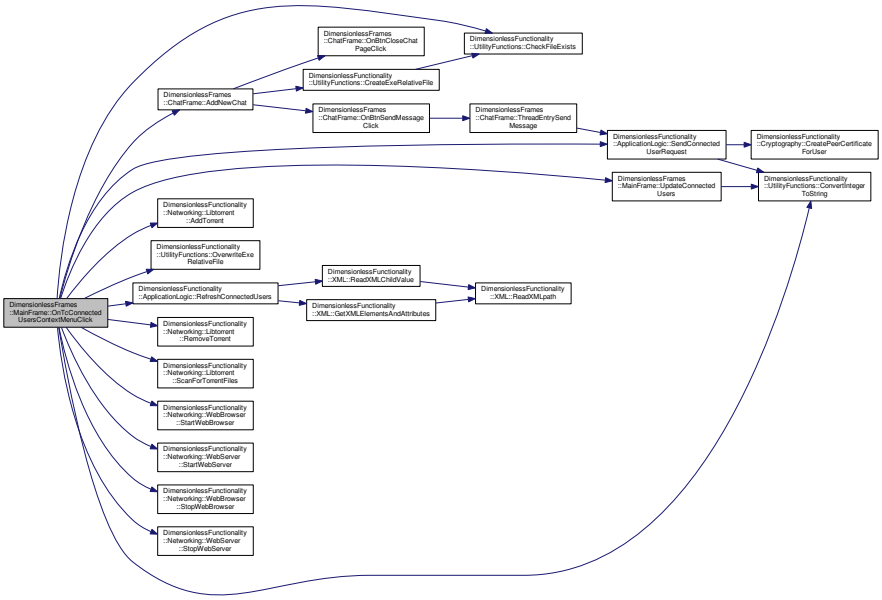
RefreshConnectedUsers(), DimensionlessFunctionality::Networking::Libtorrent::RemoveTorrent(), DimensionlessFunctionality::Networking::Libtorrent::ScanForTorrentFiles(), DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest(), DimensionlessFunctionality::Networking::WebBrowser::StartWebBrowser(), DimensionlessFunctionality::Networking::WebServer::StartWebServer(), DimensionlessFunctionality::Networking::WebBrowser::StopWebBrowser(), DimensionlessFunctionality::Networking::WebServer::StopWebServer(), UpdateConnectedUsers(), and userPageOpen.

Referenced by OnTcConnectedUsersItemRightClick().

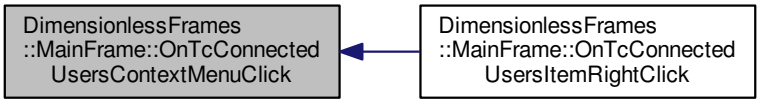
```
334     {
335         string menuTitle = menu->GetTitle().ToStdString();
336         menu->~wxMenu();
337         int connectedUserIndex = atoi(menuTitle.c_str());
338         switch(event.GetId()){
339             case idContextMenuAcceptConnectionRequest:
340             {
341                 //do the php request
342                 SendConnectedUserRequest(Request::AcceptConnectionRequest,
    CurrentUserSession::connectedUsers[connectedUserIndex]);
343                 //refresh
344                 RefreshConnectedUsers();
345                 UpdateConnectedUsers();
346                 break;
347             }
348             case idContextMenuViewSiteOnline:
349             {
350                 //do the php request
351                 SendConnectedUserRequest(Request::GetConnectedUserXML,
    CurrentUserSession::connectedUsers[connectedUserIndex]);
352                 SendConnectedUserRequest(Request::GetConnectedUserTorrent,
    CurrentUserSession::connectedUsers[connectedUserIndex]);
353                 vector<string> torrentFiles = ScanForTorrentFiles("./Users");
354                 RemoveTorrent(CurrentUserSession::connectedUsers[connectedUserIndex]->username
    );
355                 AddTorrent("./Users/"+CurrentUserSession::connectedUsers[connectedUserIndex]->
    username+"/"+CurrentUserSession::connectedUsers[connectedUserIndex]->username+".torrent","./Users/"+
    CurrentUserSession::connectedUsers[connectedUserIndex]->username+"/CA/"+CurrentUserSession::currentUser->username+"
    -cert.pem","./Users/"+CurrentUserSession::connectedUsers[connectedUserIndex]->username+"/CA/private/"+
    CurrentUserSession::currentUser->username+"-privatekey.pem","./Users/"+CurrentUserSession::connectedUsers[
    connectedUserIndex]->username+"/CA/dhparams.pem",CurrentUserSession::currentUser->AESiv);
356                 for(libtorrent::torrent_handle& h : currentSession->get_torrents()){
357                     libtorrent::torrent_status s = h.status();
358                     if(boost::algorithm::contains(s.name, CurrentUserSession::connectedUsers[
    connectedUserIndex]->username)){
359                         while(!s.is_finished){
360                             //wait.
361                         }
362                         break;
363                     }
364                 }
365                 // libtorrent scan_dir and load torrents up
366                 // wait until finished
367                 // load website
368                 if(userPageOpen == ""){
369                     userPageOpen = CurrentUserSession::connectedUsers[connectedUserIndex]->
    username;
370                 } else {
371                     StopWebServer("Users/"+userPageOpen);
372                     if(CheckFileExists(GetExecutablePath()+"Users/"+
    userPageOpen+"/nginx.conf")){
373                         std::ifstream file(GetExecutablePath()+"Users/"+
    userPageOpen+"/nginx.conf");
374                         std::stringstream buffer;
375                         buffer << file.rdbuf();
376                         std::string str = buffer.str();
377                         file.close();
378                         boost::replace_all(str, ConvertIntegerToString(80),
    ConvertIntegerToString(1337));
379                         OverwriteExeRelativeFile("Users/"+
    userPageOpen+"/nginx.conf", str);
380                     }
381                     userPageOpen = CurrentUserSession::connectedUsers[connectedUserIndex]->
    username;
382                 }
383                 if(CheckFileExists(GetExecutablePath()+"Users/"+
    userPageOpen+"/nginx.conf")){
384                     std::ifstream file(GetExecutablePath()+"Users/"+userPageOpen+"/nginx.conf")
    ;
385                     std::stringstream buffer;
386                     buffer << file.rdbuf();
387                     std::string str = buffer.str();
388                     file.close();
```

```
389                     boost::replace_all(str, ConvertIntegerToString(1337),
    ConvertIntegerToString(80));
390                     OverwriteExeRelativeFile("Users/"+
    userPageOpen+"/nginx.conf", str);
391                 }
392                 StartWebServer("Users/"+userPageOpen, "");
393                 StopWebBrowser();
394                 StartWebBrowser("http://localhost");
395                 break;
396             }
397             case idContextMenuViewSiteOffline:
398             {
399                 if(userPageOpen == ""){
400                     userPageOpen = CurrentUserSession::connectedUsers[connectedUserIndex]->
    username;
401                 } else {
402                     StopWebServer("Users/"+userPageOpen);
403                     if(CheckFileExists(GetExecutablePath()+"Users/"+
    userPageOpen+"/nginx.conf")){
404                         std::ifstream file(GetExecutablePath()+"Users/"+
    userPageOpen+"/nginx.conf");
405                         std::stringstream buffer;
406                         buffer << file.rdbuf();
407                         std::string str = buffer.str();
408                         file.close();
409                         boost::replace_all(str, ConvertIntegerToString(80),
    ConvertIntegerToString(1337));
410                         OverwriteExeRelativeFile("Users/"+
    userPageOpen+"/nginx.conf", str);
411                     }
412                     userPageOpen = CurrentUserSession::connectedUsers[connectedUserIndex]->
    username;
413                 }
414                 if(CheckFileExists(GetExecutablePath()+"Users/"+
    userPageOpen+"/nginx.conf")){
415                     std::ifstream file(GetExecutablePath()+"Users/"+userPageOpen+"/nginx.conf")
    ;
416                     std::stringstream buffer;
417                     buffer << file.rdbuf();
418                     std::string str = buffer.str();
419                     file.close();
420                     boost::replace_all(str, ConvertIntegerToString(1337),
    ConvertIntegerToString(80));
421                     OverwriteExeRelativeFile("Users/"+
    userPageOpen+"/nginx.conf", str);
422                 }
423                 StartWebServer("Users/"+userPageOpen, "");
424                 StopWebBrowser();
425                 StartWebBrowser("http://localhost");
426                 //load offline site.
427                 break;
428             }
429             case idContextMenuSendMessage:
430             {
431                 if(chatFrame == nullptr){
432                     chatFrame = new ChatFrame(this, std::ref(CurrentUserSession::connectedUsers[
    connectedUserIndex]));
433                     chatFrame->Show();
434                 } else {
435                     chatFrame->AddNewChat(std::ref(CurrentUserSession::connectedUsers[
    connectedUserIndex]));
436                     chatFrame->CenterOnParent();
437                     chatFrame->Raise();
438                 }
439                 break;
440             }
441             case idContextMenuRejectConnectionRequest:
442             {
443                 //send php request
444                 SendConnectedUserRequest(Request::RejectConnectionRequest,
    CurrentUserSession::connectedUsers[connectedUserIndex]);
445                 RefreshConnectedUsers();
446                 UpdateConnectedUsers();
447                 //block everywhere necessary
448                 break;
449             }
450         }
451         //std::cout << event.GetString().ToStdString() << std::endl;
452     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.11.3.10 OnTcConnectedUsersItemRightClick()

```
void DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick (
            wxTreeEvent & event )  [protected], [virtual]
```

This function manages what happens after the right mouse button is clicked over a ConnectedUser in our list.

Parameters

in	*event*	A wxTreeEvent object that is posted after the right mouse button is clicked.

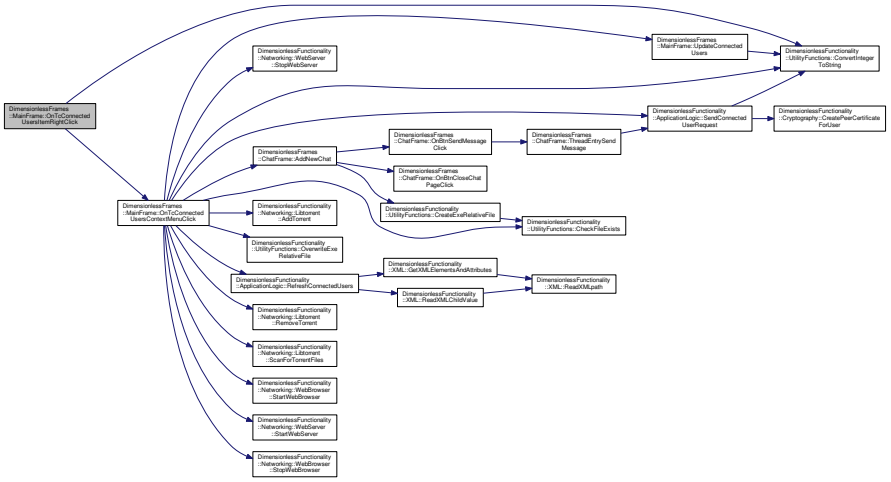
Reimplemented from MainFrameBase.

Definition at line 458 of file MainFrame.cpp.

References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), idContextMenuAcceptConnectionRequest, idContextMenuRejectConnectionRequest, idContextMenuSendMessage, idContextMenuViewSiteOffline, idContextMenuViewSiteOnline, menu, OnTcConnectedUsersContextMenuClick(), DimensionlessFrames::ConnectedUserIndex::targetUserIndex, and MainFrameBase::tcConnectedUsers.

```
459     {
460         wxTreeItemId t1 = event.GetItem();
461 
462         if(boost::algorithm::contains(this->tcConnectedUsers->GetItemText(t1).ToStdString()
    , "Connected Users")
463         || boost::algorithm::contains(this->tcConnectedUsers->GetItemText(t1).ToStdString()
    , "Online")
464         || boost::algorithm::contains(this->tcConnectedUsers->GetItemText(t1).ToStdString()
    , "Offline")
465         || boost::algorithm::contains(this->tcConnectedUsers->GetItemText(t1).ToStdString()
    , "Rejected")
466         || boost::algorithm::contains(this->tcConnectedUsers->GetItemText(t1).ToStdString()
    , "Connection Requests")){
467             event.Skip();
468             return;
469         }
470 
471         ConnectedUserIndex* tt = (ConnectedUserIndex*)this->tcConnectedUsers->GetItemData(
    t1);
472         menu = new wxMenu();
473         menu->SetTitle(ConvertIntegerToString(tt->targetUserIndex));
474         if (boost::algorithm::contains(this->tcConnectedUsers->GetItemText(this->
    tcConnectedUsers->GetItemParent(t1)).ToStdString(), "Online")){
475             menu->Append(idContextMenuAcceptConnectionRequest, "
    Accept Connection Request", "Accept Connection Request", false);
476             menu->Append(idContextMenuViewSiteOnline, "View Site", "View
     Site", false);
477             menu->Append(idContextMenuSendMessage, "Send Message", "Send
     Message", false);
478             menu->Append(idContextMenuRejectConnectionRequest, "
    Reject Connection Request", "Reject Connection Request", false);
479         } else if (boost::algorithm::contains(this->tcConnectedUsers->GetItemText(this->
    tcConnectedUsers->GetItemParent(t1)).ToStdString(), "Offline")){
480             menu->Append(idContextMenuViewSiteOffline, "View Site", "View
     Site", false);
481             menu->Append(idContextMenuRejectConnectionRequest, "
    Reject Connection Request", "Reject Connection Request", false);
482         } else if (boost::algorithm::contains(this->tcConnectedUsers->GetItemText(this->
    tcConnectedUsers->GetItemParent(t1)).ToStdString(), "Rejected")){
483             menu->Append(idContextMenuViewSiteOffline, "View Site", "View
     Site", false);
484             menu->Append(idContextMenuRejectConnectionRequest, "
    Reject Connection Request", "Reject Connection Request", false);
485         } else {
486             menu->Append(idContextMenuAcceptConnectionRequest, "
    Accept Connection Request", "Accept Connection Request", false);
487             menu->Append(idContextMenuRejectConnectionRequest, "
    Reject Connection Request", "Reject Connection Request", false);
488         }
489         menu->Bind(wxEVT_COMMAND_MENU_SELECTED, &
    MainFrame::OnTcConnectedUsersContextMenuClick, this, wxID_ANY)
    ;
490         PopupMenu(menu);
491     }
```

Here is the call graph for this function:



12.11.3.11 UpdateConnectedUsers()

```
void DimensionlessFrames::MainFrame::UpdateConnectedUsers ( )  [private]
```

This function refreshes the list of ConnectedUsers that are visible to our current user.

Definition at line 282 of file MainFrame.cpp.

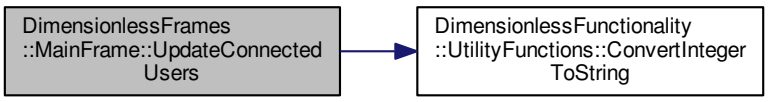
References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), rootTcConnectedUsersNode, and MainFrameBase::tcConnectedUsers.

Referenced by OnTcConnectedUsersContextMenuClick().

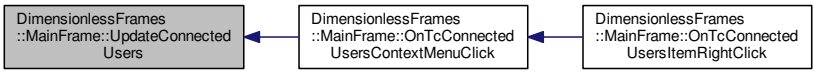
```
283     {
284         this->tcConnectedUsers->DeleteAllItems();
285         int onlineUsers = 0;
286         int offlineUsers = 0;
287         int rejectedUsers = 0;
288         int requestUsers = 0;
289         rootTcConnectedUsersNode = this->
    tcConnectedUsers->AddRoot("Connected Users");
290         wxTreeItemId onlineNode = this->tcConnectedUsers->AppendItem(
    rootTcConnectedUsersNode, "Online");
291         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
292             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::online){
293                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(onlineNode,
    CurrentUserSession::connectedUsers[i]->username);
294                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
295                 onlineUsers++;
296             }
297         }
298         this->tcConnectedUsers->SetItemText(onlineNode, "Online ("+
    ConvertIntegerToString(onlineUsers)+" users)");
299         wxTreeItemId offlineNode = this->tcConnectedUsers->AppendItem(
    rootTcConnectedUsersNode, "Offline");
300         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
301             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::offline){
302                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(offlineNode,
    CurrentUserSession::connectedUsers[i]->username);
303                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
304                 offlineUsers++;
```

```
305             }
306         }
307         this->tcConnectedUsers->SetItemText(offlineNode, "Offline ("+
    ConvertIntegerToString(offlineUsers)+" users)");
308         wxTreeItemId rejectedNode = this->tcConnectedUsers->AppendItem(
    rootTcConnectedUsersNode, "Rejected");
309         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
310             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::rejected){
311                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(rejectedNode,
    CurrentUserSession::connectedUsers[i]->username);
312                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
313                 rejectedUsers++;
314             }
315         }
316         this->tcConnectedUsers->SetItemText(rejectedNode, "Rejected ("+
    ConvertIntegerToString(rejectedUsers)+" users)");
317         wxTreeItemId connectionRequestsNode = this->tcConnectedUsers->AppendItem(
    rootTcConnectedUsersNode, "Connection Requests");
318         for(unsigned int i = 0; i < CurrentUserSession::connectedUsers.size(); i++){
319             if(CurrentUserSession::connectedUsers[i]->currentStatus == User::Status::offline){
320                 wxTreeItemId temp = this->tcConnectedUsers->AppendItem(
    connectionRequestsNode, CurrentUserSession::connectedUsers[i]->username);
321                 this->tcConnectedUsers->SetItemData(temp, (wxTreeItemData*) new
    ConnectedUserIndex(i));
322                 requestUsers++;
323             }
324         }
325         this->tcConnectedUsers->SetItemText(connectionRequestsNode, "Connection Requests ("
    +ConvertIntegerToString(requestUsers)+" requests)");
326         this->tcConnectedUsers->ExpandAll();
327     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.11.4 Member Data Documentation

12.11.4.1 chatFrame

```
ChatFrame * DimensionlessFrames::MainFrame::chatFrame = nullptr  [static]
```

This variable stores a raw pointer ChatFrame reference to show when the current user wants to chat with a connected user.

Definition at line 80 of file MainFrame.h.

Referenced by OnTcConnectedUsersContextMenuClick().

12.11.4.2 currentTargetUserIndex

```
int DimensionlessFrames::MainFrame::currentTargetUserIndex = 0  [static], [private]
```

This variable stores the index of the connected user the current users wishes to chat with.

Definition at line 119 of file MainFrame.h.

12.11.4.3 CurrentUsername

```
std::string DimensionlessFrames::MainFrame::CurrentUsername = ""  [private]
```

This variable stores the username of the currently logged in user.

Definition at line 134 of file MainFrame.h.

Referenced by MainFrame(), and OnClose().

12.11.4.4 LogOut

```
bool DimensionlessFrames::MainFrame::LogOut = false  [private]
```

This variable decides if it is time to log out.

Definition at line 137 of file MainFrame.h.

Referenced by OnBtnLogOutClick(), and OnClose().

12.11.4.5 mainFrameThread

```
std::thread DimensionlessFrames::MainFrame::mainFrameThread  [static], [private]
```

This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Definition at line 116 of file MainFrame.h.

12.11.4.6 menu

```
wxMenu * DimensionlessFrames::MainFrame::menu  [static], [private]
```

This variable stores a raw pointer to the wxMenu that is displayed upon right clicking an item in the list of connected users.

Definition at line 125 of file MainFrame.h.

Referenced by OnTcConnectedUsersContextMenuClick(), and OnTcConnectedUsersItemRightClick().

12.11.4.7 ParentWindow

```
StartupFrame* DimensionlessFrames::MainFrame::ParentWindow  [private]
```

This variable stores the parent windows of this MainFrame instance.

Definition at line 140 of file MainFrame.h.

Referenced by MainFrame(), OnBtnLogOutClick(), and OnClose().

12.11.4.8 rootTcConnectedUsersNode

```
wxTreeItemId DimensionlessFrames::MainFrame::rootTcConnectedUsersNode  [static], [private]
```

This variable stores the id of the root node in our ConnectedUsers list.

Definition at line 128 of file MainFrame.h.

Referenced by MainFrame(), and UpdateConnectedUsers().

12.11.4.9 userPageOpen

```
string DimensionlessFrames::MainFrame::userPageOpen = ""  [static], [private]
```

This variable stores the username of the user whose page is currently open.

Definition at line 131 of file MainFrame.h.

Referenced by OnTcConnectedUsersContextMenuClick().

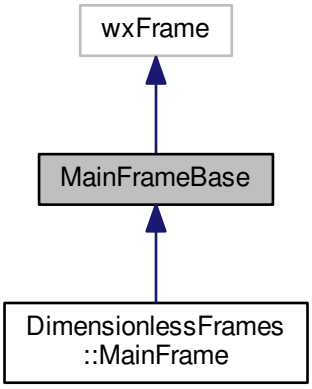
The documentation for this class was generated from the following files:

- DimensionlessClient/MainFrame.h
- DimensionlessClient/MainFrame.cpp

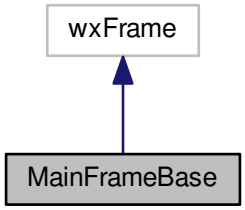
12.12 MainFrameBase Class Reference

`#include <Dimensionless.h>`

Inheritance diagram for MainFrameBase:



Collaboration diagram for MainFrameBase:



Public Types

- enum { idmiFileQuit = 10001, idmiFileSettings = 10002, idmiHelpAbout = 10003 }

Public Member Functions

- wxMenuBar * GetMbHeader ()
- wxStaticBitmap * GetBmpDimensionlessLogo ()
- wxStaticText * GetStApplicationLog ()
- wxRichTextCtrl * GetRtApplicationLog ()

- wxPanel * GetPnlLeftSidebarBackground ()
- wxStaticText * GetStConnectedUsersList ()
- wxTreeCtrl * GetTcConnectedUsers ()
- wxPanel * GetPnlRightSidebar ()
- wxButton * GetBtnOpenPersonalAdminWebsite ()
- wxButton * GetBtnSearchForUsers ()
- wxButton * GetBtnPublishUpdatedWebsite ()
- wxButton * GetBtnLogOut ()
- MainFrameBase (wxWindow *parent, wxWindowID id=wxID_ANY, const wxString &title=_("Dimensionless"), const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(600, 600), long style=wxDEFAULT_FRAME_STYLE|wxTAB_TRAVERSAL)
- virtual ~MainFrameBase ()

Protected Member Functions

- virtual void OnClose (wxCloseEvent &event)
- virtual void OnSettings (wxCommandEvent &event)
- virtual void OnQuit (wxCommandEvent &event)
- virtual void OnAbout (wxCommandEvent &event)
- virtual void OnTcConnectedUsersItemRightClick (wxTreeEvent &event)
- virtual void OnBtnOpenPersonalAdminWebsiteClick (wxCommandEvent &event)
- virtual void OnBtnSearchForUsersClick (wxCommandEvent &event)
- virtual void OnBtnPublishUpdatedWebsiteClick (wxCommandEvent &event)
- virtual void OnBtnLogOutClick (wxCommandEvent &event)

Protected Attributes

- wxMenuBar * mbHeader
- wxMenu * mFile
- wxMenuItem * miFileSettings
- wxMenuItem * miFileQuit
- wxMenu * mHelp
- wxMenuItem * miHelpAbout
- wxPanel * pnlLeftSidebarBackground
- wxStaticBitmap * bmpDimensionlessLogo
- wxStaticText * stApplicationLog
- wxRichTextCtrl * rtApplicationLog
- wxPanel * pnlRightSidebar
- wxStaticText * stConnectedUsersList
- wxTreeCtrl * tcConnectedUsers
- wxButton * btnOpenPersonalAdminWebsite
- wxButton * btnSearchForUsers
- wxButton * btnPublishUpdatedWebsite
- wxButton * btnLogOut

12.12.1 Detailed Description

Definition at line 108 of file Dimensionless.h.

12.12.2 Member Enumeration Documentation

12.12.2.1 anonymous enum

```
anonymous enum
```

Enumerator

idmiFileQuit	
idmiFileSettings	
idmiHelpAbout	

Definition at line 111 of file Dimensionless.h.

```
111          {
112         idmiFileQuit = 10001,
113         idmiFileSettings = 10002,
114         idmiHelpAbout = 10003,
115     };
```

12.12.3 Constructor & Destructor Documentation

12.12.3.1 MainFrameBase()

```
MainFrameBase::MainFrameBase (
              wxWindow * parent,
              wxWindowID id = wxID_ANY,
              const wxString & title = _("Dimensionless"),
              const wxPoint & pos = wxDefaultPosition,
              const wxSize & size = wxSize(600,600),
              long style = wxDEFAULT_FRAME_STYLE|wxTAB_TRAVERSAL )
```

Definition at line 174 of file Dimensionless.cpp.

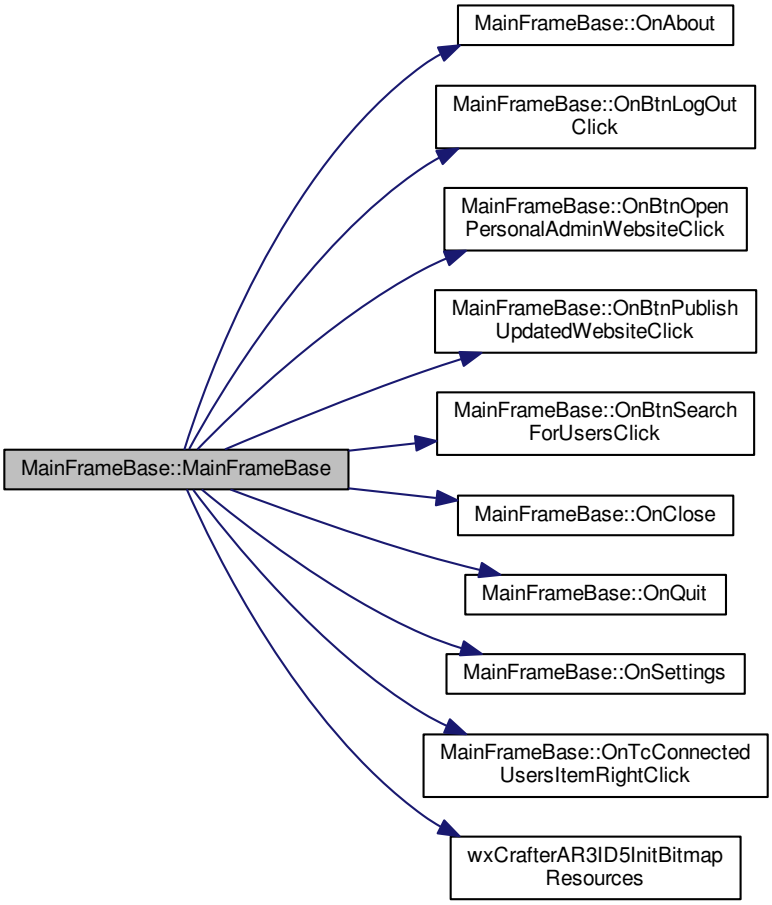
References bBitmapLoaded, bmpDimensionlessLogo, btnLogOut, btnOpenPersonalAdminWebsite, btnPublish↩UpdatedWebsite, btnSearchForUsers, idmiFileQuit, idmiFileSettings, idmiHelpAbout, mbHeader, mFile, m↩Help, miFileQuit, miFileSettings, miHelpAbout, OnAbout(), OnBtnLogOutClick(), OnBtnOpenPersonalAdmin↩WebsiteClick(), OnBtnPublishUpdatedWebsiteClick(), OnBtnSearchForUsersClick(), OnClose(), OnQuit(), On↩Settings(), OnTcConnectedUsersItemRightClick(), pnlLeftSidebarBackground, pnlRightSidebar, rtApplicationLog, stApplicationLog, stConnectedUsersList, tcConnectedUsers, WXC_FROM_DIP, and wxCrafterAR3ID5InitBitmap↩Resources().

```
175     : wxFrame(parent, id, title, pos, size, style)
176 {
177     if ( !bBitmapLoaded ) {
178         // We need to initialise the default bitmap handler
179         wxXmlResource::Get()->AddHandler(new wxBitmapXmlHandler);
180         wxCrafterAR3ID5InitBitmapResources();
181         bBitmapLoaded = true;
182     }
183
184     mbHeader = new wxMenuBar(0);
185     this->SetMenuBar(mbHeader);
186
187     mFile = new wxMenu();
188     mbHeader->Append(mFile, _("&File"));
189
190     miFileSettings = new wxMenuItem(mFile, idmiFileSettings, _("
      &Settings\tAlt+F1"), _("View Application Settings"), wxITEM_NORMAL);
191     mFile->Append(miFileSettings);
192
193     miFileQuit = new wxMenuItem(mFile, idmiFileQuit, _("&Quit\tAlt+F4"), _("Quit
       the application"), wxITEM_NORMAL);
194     mFile->Append(miFileQuit);
195
```

```
196     mHelp = new wxMenu();
197     mbHeader->Append(mHelp, _("&Help"));
198
199     miHelpAbout = new wxMenuItem(mHelp, idmiHelpAbout, _("&About\tF1"), _("
    Show info about this application"), wxITEM_NORMAL);
200     mHelp->Append(miHelpAbout);
201
202     wxBoxSizer* bszrMain = new wxBoxSizer(wxVERTICAL);
203     this->SetSizer(bszrMain);
204
205     wxBoxSizer* bszrTop = new wxBoxSizer(wxHORIZONTAL);
206
207     bszrMain->Add(bszrTop, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
208
209     wxBoxSizer* bszrLeftSidebar = new wxBoxSizer(wxVERTICAL);
210     bszrLeftSidebar->SetMinSize(250,700);
211
212     bszrTop->Add(bszrLeftSidebar, 1, wxEXPAND, WXC_FROM_DIP(5));
213
214     pnlLeftSidebarBackground = new wxPanel(this, wxID_ANY, wxDefaultPosition,
    wxDLG_UNIT(this, wxSize(-1,-1)), wxTAB_TRAVERSAL);
215
216     bszrLeftSidebar->Add(pnlLeftSidebarBackground, 1, wxEXPAND,
    WXC_FROM_DIP(0));
217
218     wxBoxSizer* bszrInnerLeftSidebar = new wxBoxSizer(wxVERTICAL);
219     pnlLeftSidebarBackground->SetSizer(bszrInnerLeftSidebar);
220
221     bmpDimensionlessLogo = new wxStaticBitmap(
    pnlLeftSidebarBackground, wxID_ANY, wxXmlResource::Get()->LoadBitmap(wxT("
    DimensionlessLogo")), wxDefaultPosition, wxDLG_UNIT(pnlLeftSidebarBackground, wxSize(250,100)
    ), 0 );
222
223     bszrInnerLeftSidebar->Add(bmpDimensionlessLogo, 0, wxALL|wxALIGN_CENTER_HORIZONTAL,
     WXC_FROM_DIP(5));
224     bmpDimensionlessLogo->SetMinSize(wxSize(250,100));
225
226     stApplicationLog = new wxStaticText(pnlLeftSidebarBackground,
    wxID_ANY, _("Application Log"), wxDefaultPosition, wxDLG_UNIT(
    pnlLeftSidebarBackground, wxSize(-1, -1)), 0);
227
228     bszrInnerLeftSidebar->Add(stApplicationLog, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|
    wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
229
230     rtApplicationLog = new wxRichTextCtrl(
    pnlLeftSidebarBackground, wxID_ANY, wxT(""), wxDefaultPosition, wxDLG_UNIT(
    pnlLeftSidebarBackground, wxSize(-1, -1)), wxTE_AUTO_URL|wxTE_READONLY|
    wxTE_MULTILINE|wxWANTS_CHARS|wxBORDER_NONE|wxVSCROLL);
231
232     bszrInnerLeftSidebar->Add(rtApplicationLog, 1, wxALL|wxEXPAND,
    WXC_FROM_DIP(10));
233     bszrLeftSidebar->SetMinSize(wxSize(250,700));
234
235     wxBoxSizer* bszrRightSidebar = new wxBoxSizer(wxVERTICAL);
236     bszrRightSidebar->SetMinSize(250,700);
237
238     bszrTop->Add(bszrRightSidebar, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
239
240     pnlRightSidebar = new wxPanel(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this, wxSize
    (-1,-1)), wxTAB_TRAVERSAL);
241
242     bszrRightSidebar->Add(pnlRightSidebar, 1, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
243
244     wxBoxSizer* bszrConnectedUsers = new wxBoxSizer(wxVERTICAL);
245     pnlRightSidebar->SetSizer(bszrConnectedUsers);
246
247     stConnectedUsersList = new wxStaticText(pnlRightSidebar, wxID_ANY, _
    ("Connected Users List"), wxDefaultPosition, wxDLG_UNIT(pnlRightSidebar, wxSize(-1, -1)),
    wxALIGN_CENTRE);
248
249     bszrConnectedUsers->Add(stConnectedUsersList, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|
    wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
250
251     tcConnectedUsers = new wxTreeCtrl(pnlRightSidebar, wxID_ANY,
    wxDefaultPosition, wxDLG_UNIT(pnlRightSidebar, wxSize(-1,-1)), wxTR_DEFAULT_STYLE);
252
253     bszrConnectedUsers->Add(tcConnectedUsers, 1, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
254     bszrRightSidebar->SetMinSize(wxSize(250,700));
255
256     wxBoxSizer* bszrBottom = new wxBoxSizer(wxVERTICAL);
257
258     bszrMain->Add(bszrBottom, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
259
260     wxBoxSizer* bszrUserActions = new wxBoxSizer(wxHORIZONTAL);
261
```

```
262     bszrBottom->Add(bszrUserActions, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
263
264     btnOpenPersonalAdminWebsite = new wxButton(this, wxID_ANY, _("Open Personal
       Admin Website"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
265
266     bszrUserActions->Add(btnOpenPersonalAdminWebsite, 1, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
267
268     btnSearchForUsers = new wxButton(this, wxID_ANY, _("Search for Users"),
      wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
269
270     bszrUserActions->Add(btnSearchForUsers, 1, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
271
272     wxBoxSizer* bszrLogOut = new wxBoxSizer(wxVERTICAL);
273
274     bszrBottom->Add(bszrLogOut, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
275
276     btnPublishUpdatedWebsite = new wxButton(this, wxID_ANY, _("Publish Updated
       Website"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
277
278     bszrLogOut->Add(btnPublishUpdatedWebsite, 0, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
279
280     btnLogOut = new wxButton(this, wxID_ANY, _("Log Out"), wxDefaultPosition, wxDLG_UNIT(this,
      wxSize(-1, -1)), 0);
281
282     bszrLogOut->Add(btnLogOut, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
283
284     SetName(wxT("MainFrameBase"));
285     SetMinClientSize(wxSize(600,600));
286     SetSize(600,600);
287     if (GetSizer()) {
288         GetSizer()->Fit(this);
289     }
290     if(GetParent()) {
291         CentreOnParent(wxBOTH);
292     } else {
293         CentreOnScreen(wxBOTH);
294     }
295 #if wxVERSION_NUMBER >= 2900
296     if(!wxPersistenceManager::Get().Find(this)) {
297         wxPersistenceManager::Get().RegisterAndRestore(this);
298     } else {
299         wxPersistenceManager::Get().Restore(this);
300     }
301 #endif
302     // Connect events
303     this->Connect(wxEVT_CLOSE_WINDOW, wxCloseEventHandler(MainFrameBase::OnClose),
      NULL, this);
304     this->Connect(miFileSettings->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler
      (MainFrameBase::OnSettings), NULL, this);
305     this->Connect(miFileQuit->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
      MainFrameBase::OnQuit), NULL, this);
306     this->Connect(miHelpAbout->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
      MainFrameBase::OnAbout), NULL, this);
307     tcConnectedUsers->Connect(wxEVT_COMMAND_TREE_ITEM_RIGHT_CLICK, wxTreeEventHandler(
      MainFrameBase::OnTcConnectedUsersItemRightClick), NULL, this
      );
308     btnOpenPersonalAdminWebsite->Connect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(MainFrameBase::OnBtnOpenPersonalAdminWebsiteClick
      ), NULL, this);
309     btnSearchForUsers->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      MainFrameBase::OnBtnSearchForUsersClick), NULL, this);
310     btnPublishUpdatedWebsite->Connect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(MainFrameBase::OnBtnPublishUpdatedWebsiteClick), NULL,
      this);
311     btnLogOut->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      MainFrameBase::OnBtnLogOutClick), NULL, this);
312
313 }
```

Here is the call graph for this function:



12.12.3.2 ~MainFrameBase()

```
MainFrameBase::~MainFrameBase ( )  [virtual]
```

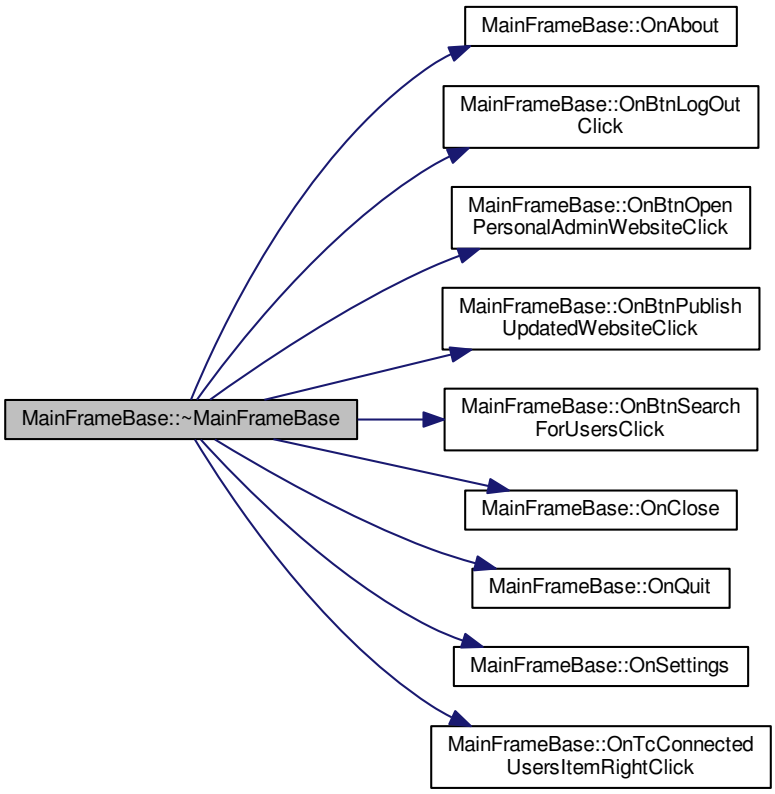
Definition at line 315 of file Dimensionless.cpp.

References btnLogOut, btnOpenPersonalAdminWebsite, btnPublishUpdatedWebsite, btnSearchForUsers, miFileQuit, miFileSettings, miHelpAbout, OnAbout(), OnBtnLogOutClick(), OnBtnOpenPersonalAdminWebsiteClick(), OnBtnPublishUpdatedWebsiteClick(), OnBtnSearchForUsersClick(), OnClose(), OnQuit(), OnSettings(), OnTcConnectedUsersItemRightClick(), and tcConnectedUsers.

```
316 {
317     this->Disconnect(wxEVT_CLOSE_WINDOW, wxCloseEventHandler(
      MainFrameBase::OnClose), NULL, this);
318     this->Disconnect(miFileSettings->GetId(), wxEVT_COMMAND_MENU_SELECTED,
      wxCommandEventHandler(MainFrameBase::OnSettings), NULL, this);
319     this->Disconnect(miFileQuit->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
      MainFrameBase::OnQuit), NULL, this);
```

```
320     this->Disconnect(miHelpAbout->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
      MainFrameBase::OnAbout), NULL, this);
321     tcConnectedUsers->Disconnect(wxEVT_COMMAND_TREE_ITEM_RIGHT_CLICK, wxTreeEventHandler(
      MainFrameBase::OnTcConnectedUsersItemRightClick), NULL, this
      );
322     btnOpenPersonalAdminWebsite->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(MainFrameBase::OnBtnOpenPersonalAdminWebsiteClick
      ), NULL, this);
323     btnSearchForUsers->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      MainFrameBase::OnBtnSearchForUsersClick), NULL, this);
324     btnPublishUpdatedWebsite->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(MainFrameBase::OnBtnPublishUpdatedWebsiteClick),
      NULL, this);
325     btnLogOut->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      MainFrameBase::OnBtnLogOutClick), NULL, this);
326
327 }
```

Here is the call graph for this function:



12.12.4 Member Function Documentation

12.12.4.1 GetBmpDimensionlessLogo()

```
wxStaticBitmap* MainFrameBase::GetBmpDimensionlessLogo ( )  [inline]
```

Definition at line 148 of file Dimensionless.h.

References StartupFrameBase::bmpDimensionlessLogo.

```
148 { return bmpDimensionlessLogo; }
```

12.12.4.2 GetBtnLogOut()

```
wxButton* MainFrameBase::GetBtnLogOut ( )  [inline]
```

Definition at line 158 of file Dimensionless.h.

```
158 { return btnLogOut; }
```

12.12.4.3 GetBtnOpenPersonalAdminWebsite()

```
wxButton* MainFrameBase::GetBtnOpenPersonalAdminWebsite ( )  [inline]
```

Definition at line 155 of file Dimensionless.h.

```
155 { return btnOpenPersonalAdminWebsite; }
```

12.12.4.4 GetBtnPublishUpdatedWebsite()

```
wxButton* MainFrameBase::GetBtnPublishUpdatedWebsite ( )  [inline]
```

Definition at line 157 of file Dimensionless.h.

```
157 { return btnPublishUpdatedWebsite; }
```

12.12.4.5 GetBtnSearchForUsers()

```
wxButton* MainFrameBase::GetBtnSearchForUsers ( )  [inline]
```

Definition at line 156 of file Dimensionless.h.

```
156 { return btnSearchForUsers; }
```

12.12.4.6 GetMbHeader()

```
wxMenuBar* MainFrameBase::GetMbHeader ( )  [inline]
```

Definition at line 147 of file Dimensionless.h.

References StartupFrameBase::mbHeader.

```
147 { return mbHeader; }
```

12.12.4.7 GetPnlLeftSidebarBackground()

```
wxPanel* MainFrameBase::GetPnlLeftSidebarBackground ( )  [inline]
```

Definition at line 151 of file Dimensionless.h.

```
151 { return pnlLeftSidebarBackground; }
```

12.12.4.8 GetPnlRightSidebar()

```
wxPanel* MainFrameBase::GetPnlRightSidebar ( )  [inline]
```

Definition at line 154 of file Dimensionless.h.

```
154 { return pnlRightSidebar; }
```

12.12.4.9 GetRtApplicationLog()

```
wxRichTextCtrl* MainFrameBase::GetRtApplicationLog ( )  [inline]
```

Definition at line 150 of file Dimensionless.h.

```
150 { return rtApplicationLog; }
```

12.12.4.10 GetStApplicationLog()

```
wxStaticText* MainFrameBase::GetStApplicationLog ( )  [inline]
```

Definition at line 149 of file Dimensionless.h.

```
149 { return stApplicationLog; }
```

12.12.4.11 GetStConnectedUsersList()

```
wxStaticText* MainFrameBase::GetStConnectedUsersList ( )  [inline]
```

Definition at line 152 of file Dimensionless.h.

```
152 { return stConnectedUsersList; }
```

12.12.4.12 GetTcConnectedUsers()

```
wxTreeCtrl* MainFrameBase::GetTcConnectedUsers ( )  [inline]
```

Definition at line 153 of file Dimensionless.h.

```
153 { return tcConnectedUsers; }
```

12.12.4.13 OnAbout()

```
virtual void MainFrameBase::OnAbout (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

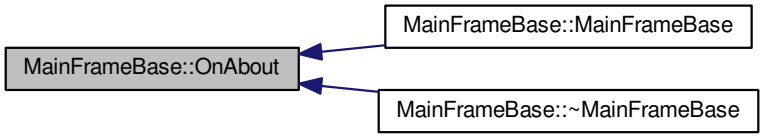
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 139 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
139 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.14 OnBtnLogOutClick()

```
virtual void MainFrameBase::OnBtnLogOutClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

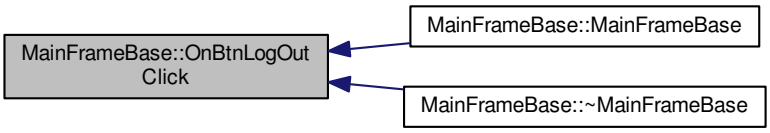
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 144 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
144 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.15 OnBtnOpenPersonalAdminWebsiteClick()

```
virtual void MainFrameBase::OnBtnOpenPersonalAdminWebsiteClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

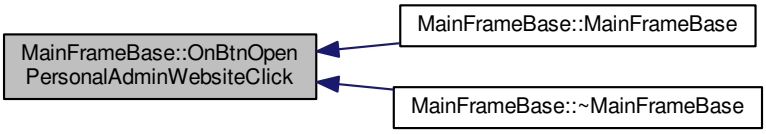
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 141 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
141 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.16 OnBtnPublishUpdatedWebsiteClick()

```
virtual void MainFrameBase::OnBtnPublishUpdatedWebsiteClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

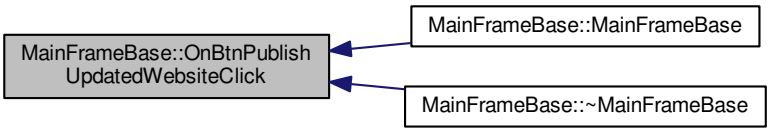
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 143 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
143 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.17 OnBtnSearchForUsersClick()

```
virtual void MainFrameBase::OnBtnSearchForUsersClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

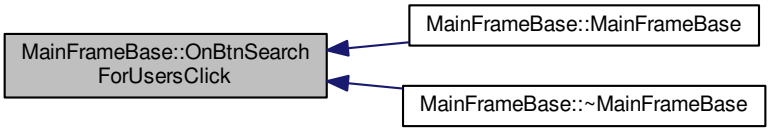
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 142 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
142 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.18 OnClose()

```
virtual void MainFrameBase::OnClose (
            wxCloseEvent & event )  [inline], [protected], [virtual]
```

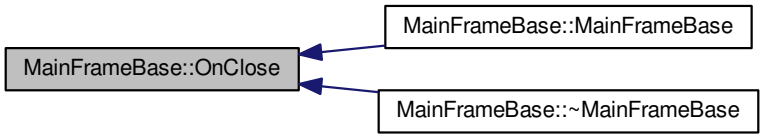
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 136 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
136 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.19 OnQuit()

```
virtual void MainFrameBase::OnQuit (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

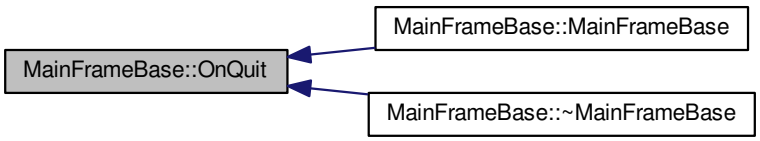
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 138 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
138 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.20 OnSettings()

```
virtual void MainFrameBase::OnSettings (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

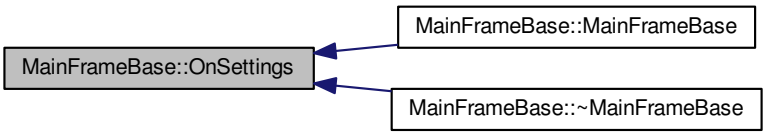
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 137 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
137 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.4.21 OnTcConnectedUsersItemRightClick()

```
virtual void MainFrameBase::OnTcConnectedUsersItemRightClick (
            wxTreeEvent & event )  [inline], [protected], [virtual]
```

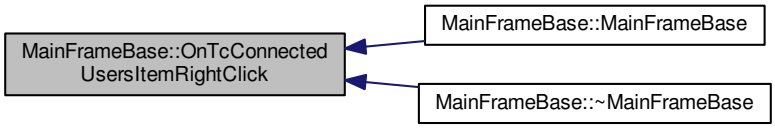
Reimplemented in DimensionlessFrames::MainFrame.

Definition at line 140 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

```
140 { event.Skip(); }
```

Here is the caller graph for this function:



12.12.5 Member Data Documentation

12.12.5.1 bmpDimensionlessLogo

```
wxStaticBitmap* MainFrameBase::bmpDimensionlessLogo  [protected]
```

Definition at line 124 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.2 btnLogOut

```
wxButton* MainFrameBase::btnLogOut  [protected]
```

Definition at line 133 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.3 btnOpenPersonalAdminWebsite

```
wxButton* MainFrameBase::btnOpenPersonalAdminWebsite  [protected]
```

Definition at line 130 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.4 btnPublishUpdatedWebsite

```
wxButton* MainFrameBase::btnPublishUpdatedWebsite  [protected]
```

Definition at line 132 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.5 btnSearchForUsers

```
wxButton* MainFrameBase::btnSearchForUsers  [protected]
```

Definition at line 131 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.6 mbHeader

```
wxMenuBar* MainFrameBase::mbHeader  [protected]
```

Definition at line 117 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.7 mFile

```
wxMenu* MainFrameBase::mFile  [protected]
```

Definition at line 118 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.8 mHelp

```
wxMenu* MainFrameBase::mHelp  [protected]
```

Definition at line 121 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.9 miFileQuit

```
wxMenuItem* MainFrameBase::miFileQuit  [protected]
```

Definition at line 120 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.10 miFileSettings

```
wxMenuItem* MainFrameBase::miFileSettings  [protected]
```

Definition at line 119 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.11 miHelpAbout

```
wxMenuItem* MainFrameBase::miHelpAbout  [protected]
```

Definition at line 122 of file Dimensionless.h.

Referenced by MainFrameBase(), and ~MainFrameBase().

12.12.5.12 pnlLeftSidebarBackground

```
wxPanel* MainFrameBase::pnlLeftSidebarBackground  [protected]
```

Definition at line 123 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.13 pnlRightSidebar

```
wxPanel* MainFrameBase::pnlRightSidebar  [protected]
```

Definition at line 127 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.14 rtApplicationLog

```
wxRichTextCtrl* MainFrameBase::rtApplicationLog  [protected]
```

Definition at line 126 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.15 stApplicationLog

```
wxStaticText* MainFrameBase::stApplicationLog  [protected]
```

Definition at line 125 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.16 stConnectedUsersList

```
wxStaticText* MainFrameBase::stConnectedUsersList  [protected]
```

Definition at line 128 of file Dimensionless.h.

Referenced by MainFrameBase().

12.12.5.17 tcConnectedUsers

```
wxTreeCtrl* MainFrameBase::tcConnectedUsers  [protected]
```

Definition at line 129 of file Dimensionless.h.

Referenced by DimensionlessFrames::MainFrame::MainFrame(), MainFrameBase(), DimensionlessFrames↩ ::MainFrame::OnTcConnectedUsersItemRightClick(), DimensionlessFrames::MainFrame::UpdateConnected↩ Users(), and ~MainFrameBase().

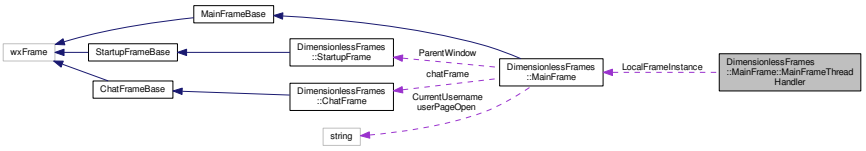
The documentation for this class was generated from the following files:

- DimensionlessClient/Dimensionless.h
- DimensionlessClient/Dimensionless.cpp

12.13 DimensionlessFrames::MainFrame::MainFrameThreadHandler Struct Reference

This struct handles events posted from our mainFrameThread.

Collaboration diagram for DimensionlessFrames::MainFrame::MainFrameThreadHandler:



Public Member Functions

- MainFrameThreadHandler (MainFrame *frame)
- void operator() (wxThreadEvent &evt)

Private Attributes

- MainFrame * LocalFrameInstance = nullptr

  A raw pointer reference to a ChatFrame to handle events for.

12.13.1 Detailed Description

This struct handles events posted from our mainFrameThread.

Definition at line 66 of file MainFrame.cpp.

12.13.2 Constructor & Destructor Documentation

12.13.2.1 MainFrameThreadHandler()

```
DimensionlessFrames::MainFrame::MainFrameThreadHandler::MainFrameThreadHandler (
            MainFrame * frame )  [inline]
```

This constructor initializes a new MainFrameThreadHandler object with a pointer to the current MainFrame instance so it may be modified as required.

Parameters

in	*frame*	The raw pointer to a MainFrame instance.

Definition at line 73 of file MainFrame.cpp.

```
74             {
75                 // Set the local MainFrame reference to that passed.
76                 LocalFrameInstance = frame;
77             }
```

12.13.3 Member Function Documentation

12.13.3.1 operator()()

```
void DimensionlessFrames::MainFrame::MainFrameThreadHandler::operator() (
            wxThreadEvent & evt )  [inline]
```

This function remaps the () operator of MainFrameThreadHandler to handle mainFrameThread events.

Parameters

in	*event*	A wxThreadEvent object posted by our mainFrameThread.

Definition at line 83 of file MainFrame.cpp.

References idMainFrame.

```
84              {
85                  switch(evt.GetId()){
86                      case idMainFrame:
87                      {
88                          //wxProgressDialog
89                          break;
90                      }
91                      default:
92                      {
93                          break;
94                      }
95                  }
96              }
```

12.13.4 Member Data Documentation

12.13.4.1 LocalFrameInstance

```
MainFrame* DimensionlessFrames::MainFrame::MainFrameThreadHandler::LocalFrameInstance = nullptr
[private]
```

A raw pointer reference to a ChatFrame to handle events for.

Definition at line 99 of file MainFrame.cpp.

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

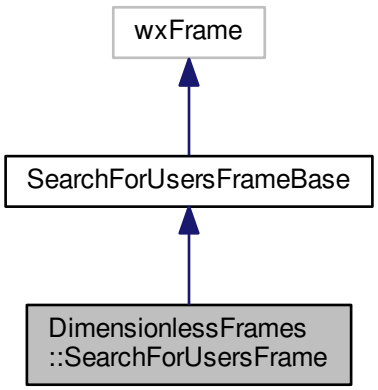
- DimensionlessClient/MainFrame.cpp

12.14 DimensionlessFrames::SearchForUsersFrame Class Reference

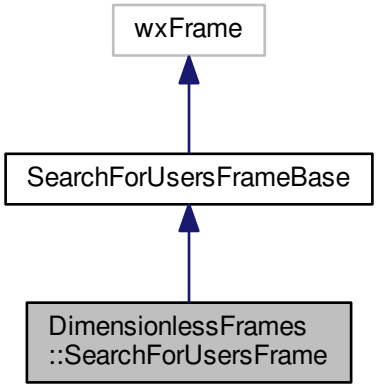
This class creates a SearchForUsersFrame that is displayed when the currently logged in user wishes to search for other users.

```
#include <SearchForUsersFrame.h>
```

Inheritance diagram for DimensionlessFrames::SearchForUsersFrame:



Collaboration diagram for DimensionlessFrames::SearchForUsersFrame:



Classes

- struct SearchForUsersFrameThreadHandler

  This struct handles events posted from our searchForUsersFrameThread.

Public Member Functions

- SearchForUsersFrame (MainFrame *parent)
- virtual ~SearchForUsersFrame ()

Protected Member Functions

- virtual void OnBtnCancelSearchClick (wxCommandEvent &event)
- virtual void OnChSearchCountryChoiceSelected (wxCommandEvent &event)
- virtual void OnChSearchTypeChoiceSelected (wxCommandEvent &event)
- virtual void OnBtnSearchClick (wxCommandEvent &event)
- virtual void OnBtnSendConnectionRequestClick (wxCommandEvent &event)

Private Member Functions

- void ThreadEntrySearchForUsers (const std::vector< string > &ips, const std::string searchUsername="")
- void ThreadEntrySearchForUser (const string &ipOrDomain, const std::string searchUsername="")
- void ThreadEntrySearchForTorUser (const string &onionDomain, const std::string searchUsername="")
- void ThreadEntrySendConnectionRequest (const std::shared_ptr< ConnectedUser > &targetUser)

Static Private Attributes

- static std::map< string, std::vector< string > > mapCountriesCities

  This map of strings and string vectors stores the collection of countries and cities available for searching.
- static std::vector< string > cities

  This string vector stores the cities for the currently selected country.
- static std::vector< std::shared_ptr< ConnectedUser > > foundUsers

  This smart pointer vector of ConnectedUser objects stores a collection of users found from a search.
- static std::thread searchForUsersFrameThread

  This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Additional Inherited Members

12.14.1 Detailed Description

This class creates a SearchForUsersFrame that is displayed when the currently logged in user wishes to search for other users.

Definition at line 50 of file SearchForUsersFrame.h.

12.14.2 Constructor & Destructor Documentation

12.14.2.1 SearchForUsersFrame()

```
DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrame (
            MainFrame * parent )
```

This constructor initializes a new SearchForUsersFrame.

Parameters

in	*parent*	The parent window of the one to be created.

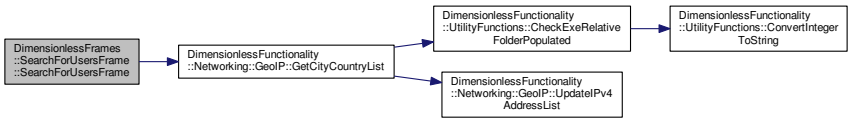
Definition at line 160 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::chSearchCity, SearchForUsersFrameBase::chSearchCountry, cities, DimensionlessFunctionality::Networking::GeoIP::GetCityCountryList(), and mapCountriesCities.

```
160                                                                 :
      SearchForUsersFrameBase(parent)
161     {
162         this->SetMinClientSize(wxSize(700,700));
163         this->SetSize(700,700);
164         this->CentreOnParent(wxBOTH);
165         mapCountriesCities = GetCityCountryList();
166         for(auto const& item : mapCountriesCities){
167             wxString country = wxString::FromUTF8(item.first.c_str());
168             this->chSearchCountry->AppendString(country);
169         }
170         this->chSearchCountry->SetSelection(0);
```

```
171         this->chSearchCity->Clear();
172         string country = "";
173         int count = 0;
174         for(auto const& item : mapCountriesCities){
175             if(count == this->chSearchCountry->GetCurrentSelection()){
176                 country = item.first;
177                 break;
178             } else {
179                 count++;
180             }
181         }
182         cities = mapCountriesCities[country];
183         this->chSearchCity->AppendString("All cities in "+country);
184         this->chSearchCity->SetSelection(0);
185         for(unsigned int i = 1; i < cities.size(); i+=2){
186             wxString tempCity = wxString::FromUTF8(cities[i].c_str());
187             if(tempCity.ToStdString() == "" || tempCity.ToStdString() == " "){
188                 tempCity = "Unknown City ("+cities[i-1]+")";
189             }
190             this->chSearchCity->AppendString(tempCity);
191         }
192     }
```

Here is the call graph for this function:



12.14.2.2 ~SearchForUsersFrame()

```
DimensionlessFrames::SearchForUsersFrame::~SearchForUsersFrame ( )  [virtual]
```

This destructor currently does nothing.

Definition at line 197 of file SearchForUsersFrame.cpp.

```
198     {
199     }
```

12.14.3 Member Function Documentation

12.14.3.1 OnBtnCancelSearchClick()

```
void DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Cancel Search" button is clicked on a SearchForUsersFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Cancel Search" button is clicked.

Reimplemented from SearchForUsersFrameBase.

Definition at line 286 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::btnCancelSearch, SearchForUsersFrameBase::btnSearch, SearchForUsersFrameBase::btnSendConnectionRequest, SearchForUsersFrameBase::chSearchCity, SearchForUsersFrameBase::chSearchCountry, SearchForUsersFrameBase::chSearchType, SearchForUsersFrameBase::gaugeFindingUsers, and searchForUsersFrameThread.

```
287     {
288         searchForUsersFrameThread.~thread();
289         this->btnSearch->Enable(true);
290         this->btnCancelSearch->Enable(false);
291         this->btnCancelSearch->Hide();
292         this->btnSendConnectionRequest->Enable(true);
293         this->chSearchType->Enable(true);
294         this->chSearchCountry->Enable(true);
295         this->chSearchCity->Enable(true);
296         this->gaugeFindingUsers->Hide();
297         this->gaugeFindingUsers->SetValue(0);
298         this->Layout();
299     }
```

12.14.3.2 OnBtnSearchClick()

```
void DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Search" button is clicked on a SearchForUsersFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Search" button is clicked.

Reimplemented from SearchForUsersFrameBase.

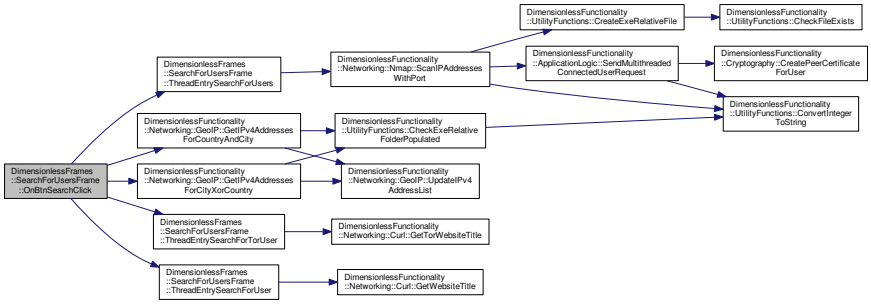
Definition at line 205 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::btnCancelSearch, SearchForUsersFrameBase::btnSearch, SearchForUsersFrameBase::btnSendConnectionRequest, SearchForUsersFrameBase::chSearchCity, SearchForUsersFrameBase::chSearchCountry, SearchForUsersFrameBase::chSearchType, foundUsers, SearchForUsersFrameBase::gaugeFindingUsers, DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCityXorCountry(), DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCountryAndCity(), SearchForUsersFrameBase::lbFoundUsers, mapCountriesCities, searchForUsersFrameThread, SearchForUsersFrameBase::tcSearchIPv4AddressOrDomain, SearchForUsersFrameBase::tcSearchUsername, ThreadEntrySearchForTorUser(), ThreadEntrySearchForUser(), and ThreadEntrySearchForUsers().

```
206     {
207         this->foundUsers.clear();
208         this->lbFoundUsers->Clear();
```

```
209         this->btnSearch->Enable(false);
210         this->btnCancelSearch->Enable(true);
211         this->btnCancelSearch->Show();
212         this->btnSendConnectionRequest->Enable(false);
213         this->chSearchType->Enable(false);
214         this->chSearchCountry->Enable(false);
215         this->chSearchCity->Enable(false);
216         this->gaugeFindingUsers->Show();
217         this->gaugeFindingUsers->SetValue(10);
218         this->Layout();
219         switch(chSearchType->GetSelection()){
220             case 0:
221             {
222                 //regular domain/ip
223                 searchForUsersFrameThread = std::thread(&
    SearchForUsersFrame::ThreadEntrySearchForUser, this,
    tcSearchIPv4AddressOrDomain->GetValue().ToStdString(), "");
224                 searchForUsersFrameThread.detach();
225                 break;
226             }
227             case 1:
228             {
229                 //onion
230                 searchForUsersFrameThread = std::thread(&
    SearchForUsersFrame::ThreadEntrySearchForTorUser, this,
    tcSearchIPv4AddressOrDomain->GetValue().ToStdString(), "");
231                 searchForUsersFrameThread.detach();
232                 break;
233             }
234             case 2:
235             {
236                 //geoip
237                 string country = "";
238                 string city = "";
239 
240                 int count = 1;
241                 for(auto const& item : mapCountriesCities){
242                     if(count == this->chSearchCountry->GetCurrentSelection()){
243                         country = item.first;
244                         break;
245                     } else {
246                         count++;
247                     }
248                 }
249                 if(this->chSearchCity->GetCurrentSelection() != 0){
250                     count = 1;
251                     for(auto const& item : mapCountriesCities){
252                         if(count == this->chSearchCity->GetCurrentSelection()){
253                             city = item.second[this->chSearchCity->GetCurrentSelection()];
254                             break;
255                         } else {
256                             count++;
257                         }
258                     }
259                 }
260                 std::vector<string> ips;
261                 if(this->chSearchCity->GetCurrentSelection() == 0){
262                     ips = GetIPv4AddressesForCityXorCountry(country);
263                 } else {
264                     ips = GetIPv4AddressesForCountryAndCity(country, city)
    ;
265                 }
266                 std::thread::id nothread;
267                 if(searchForUsersFrameThread.get_id() != nothread){
268                     searchForUsersFrameThread.~thread();
269                 }
270 
271                 searchForUsersFrameThread = std::thread(&
    SearchForUsersFrame::ThreadEntrySearchForUsers, this, ips,
    tcSearchUsername->GetValue().ToStdString());
272                 searchForUsersFrameThread.detach();
273                 break;
274             }
275             default:
276             {
277                 break;
278             }
279         }
280     }
```

Here is the call graph for this function:



12.14.3.3 OnBtnSendConnectionRequestClick()

```
void DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Send Connection Request" button is clicked on a SearchForUsers↩ Frame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Send Connection Request" button is clicked.

Reimplemented from SearchForUsersFrameBase.

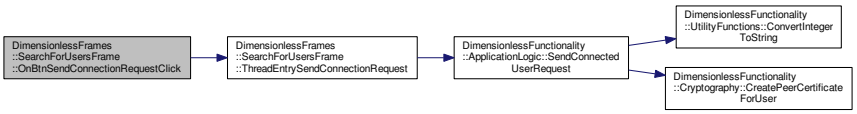
Definition at line 305 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::btnSearch, SearchForUsersFrameBase::btnSendConnectionRequest, SearchForUsersFrameBase::chSearchCity, SearchForUsersFrameBase::chSearchCountry, SearchForUsers↩ FrameBase::chSearchType, foundUsers, SearchForUsersFrameBase::gaugeFindingUsers, SearchForUsers↩ FrameBase::lbFoundUsers, searchForUsersFrameThread, and ThreadEntrySendConnectionRequest().

```
306     {
307         this->btnSearch->Enable(false);
308         this->btnSendConnectionRequest->Enable(false);
309         this->chSearchType->Enable(false);
310         this->chSearchCountry->Enable(false);
311         this->chSearchCity->Enable(false);
312         this->gaugeFindingUsers->Show();
313         this->gaugeFindingUsers->SetValue(10);
314         this->Layout();
315         std::thread::id nothread;
316         if(searchForUsersFrameThread.get_id() != nothread){
317             searchForUsersFrameThread.~thread();
318         }
319 
320         if(this->lbFoundUsers->GetSelection() >= 0){
321             searchForUsersFrameThread = std::thread(&
      SearchForUsersFrame::ThreadEntrySendConnectionRequest,
       this, foundUsers[this->lbFoundUsers->GetSelection()]);
322             searchForUsersFrameThread.detach();
323         } else {
324             wxMessageBox("Please select a user to send a connection request to.","Error - no found user
       selected",wxOK|wxCENTRE,this);
325             this->btnSearch->Enable(true);
```

```
326             this->btnSendConnectionRequest->Enable(true);
327             this->chSearchType->Enable(true);
328             this->chSearchCountry->Enable(true);
329             this->chSearchCity->Enable(true);
330             this->gaugeFindingUsers->Hide();
331             this->gaugeFindingUsers->SetValue(0);
332             this->Layout();
333         }
334     }
```

Here is the call graph for this function:



12.14.3.4 OnChSearchCountryChoiceSelected()

```
void DimensionlessFrames::SearchForUsersFrame::OnChSearchCountryChoiceSelected (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after a country is selected in the appropriate dropdown.

Parameters

in	*event*	A wxCommandEvent object that is posted after a country is selected in the appropriate dropdown.

Reimplemented from SearchForUsersFrameBase.

Definition at line 417 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::chSearchCity, SearchForUsersFrameBase::chSearchCountry, cities, and mapCountriesCities.

```
418     {
419         this->chSearchCity->Clear();
420         string country = "";
421         int count = 0;
422         for(auto const& item : mapCountriesCities){
423             if(count == this->chSearchCountry->GetCurrentSelection()){
424                 country = item.first;
425                 break;
426             } else {
427                 count++;
428             }
429         }
430         cities = mapCountriesCities[country];
431         this->chSearchCity->AppendString("All cities in "+country);
432         this->chSearchCity->SetSelection(0);
433         for(unsigned int i = 1; i < cities.size(); i+=2){
434             wxString tempCity = wxString::FromUTF8(cities[i].c_str());
435             if(tempCity.ToStdString() == "" || tempCity.ToStdString() == " "){
436                 tempCity = "Unknown City ("+cities[i-1]+")";
437             }
438             this->chSearchCity->AppendString(tempCity);
439         }
440     }
```

12.14.3.5 OnChSearchTypeChoiceSelected()

```
void DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after a search type is selected in the appropriate dropdown.

Parameters

in	*event*	A wxCommandEvent object that is posted after a search type is selected in the appropriate dropdown.

Reimplemented from SearchForUsersFrameBase.

Definition at line 340 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::chSearchCity, SearchForUsersFrameBase::chSearchCountry, SearchForUsersFrameBase::chSearchType, SearchForUsersFrameBase::stSearchCountryCity, SearchForUsersFrameBase::stSearchIPv4AddressOrDomain, SearchForUsersFrameBase::stSearchOnionDomain, SearchForUsersFrameBase::stSearchUsername, SearchForUsersFrameBase::tcSearchIPv4AddressOrDomain, SearchForUsersFrameBase::tcSearchOnionDomain, and SearchForUsersFrameBase::tcSearchUsername.

```
341     {
342         switch(chSearchType->GetSelection()){
343             case 0:
344             {
345                 stSearchIPv4AddressOrDomain->Show(true);
346                 tcSearchIPv4AddressOrDomain->Show(true);
347                 tcSearchIPv4AddressOrDomain->Enable(true);
348 
349                 stSearchOnionDomain->Show(false);
350                 tcSearchOnionDomain->Show(false);
351                 tcSearchOnionDomain->Enable(false);
352 
353                 stSearchUsername->Show(false);
354                 tcSearchUsername->Show(false);
355                 tcSearchUsername->Enable(false);
356                 stSearchCountryCity->Show(false);
357                 chSearchCountry->Show(false);
358                 chSearchCountry->Enable(false);
359                 chSearchCity->Show(false);
360                 chSearchCity->Enable(false);
361                 this->Layout();
362                 break;
363             }
364             case 1:
365             {
366                 stSearchIPv4AddressOrDomain->Show(false);
367                 tcSearchIPv4AddressOrDomain->Show(false);
368                 tcSearchIPv4AddressOrDomain->Enable(false);
369 
370                 stSearchOnionDomain->Show(true);
371                 tcSearchOnionDomain->Show(true);
372                 tcSearchOnionDomain->Enable(true);
373 
374                 stSearchUsername->Show(false);
375                 tcSearchUsername->Show(false);
376                 tcSearchUsername->Enable(false);
377                 stSearchCountryCity->Show(false);
378                 chSearchCountry->Show(false);
379                 chSearchCountry->Enable(false);
380                 chSearchCity->Show(false);
381                 chSearchCity->Enable(false);
382                 this->Layout();
383                 break;
384             }
385             case 2:
386             {
387                 stSearchIPv4AddressOrDomain->Show(false);
388                 tcSearchIPv4AddressOrDomain->Show(false);
389                 tcSearchIPv4AddressOrDomain->Enable(false);
390 
391                 stSearchOnionDomain->Show(false);
```

```
392                 tcSearchOnionDomain->Show(false);
393                 tcSearchOnionDomain->Enable(false);
394
395                 stSearchUsername->Show(true);
396                 tcSearchUsername->Show(true);
397                 tcSearchUsername->Enable(true);
398                 stSearchCountryCity->Show(true);
399                 chSearchCountry->Show(true);
400                 chSearchCountry->Enable(true);
401                 chSearchCity->Show(true);
402                 chSearchCity->Enable(true);
403                 this->Layout();
404                 break;
405             }
406             default:
407             {
408                 break;
409             }
410         }
411     }
```

12.14.3.6 ThreadEntrySearchForTorUser()

```
void DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForTorUser (
            const string & onionDomain,
            const std::string searchUsername = "" )  [private]
```

This function is called by a searchForUsersFrameThread when the "Search" button is clicked on a SearchFor↩ UsersFrame page.

Parameters

in	*onionDomain*	A string containing a Tor onion domain to search.
in	*searchUsername*	A string that contains the username of a user we are searching for.

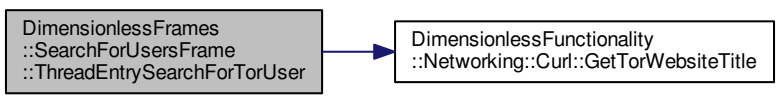
Definition at line 477 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::gaugeFindingUsers, DimensionlessFunctionality::Networking::Curl::↩ GetTorWebsiteTitle(), and idThreadSearchForTorUser.

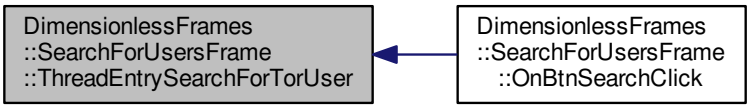
Referenced by OnBtnSearchClick().

```
478     {
479         wxThreadEvent event(wxEVT_THREAD, idThreadSearchForTorUser);
480
481         event.SetPayload<string>(GetTorWebsiteTitle(onionDomain+":1337/user.html"));
482         this->gaugeFindingUsers->SetValue(100);
483
484         wxPostEvent(this, event);
485     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.14.3.7 ThreadEntrySearchForUser()

```
void DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUser (
            const string & ipOrDomain,
            const std::string searchUsername = "" )  [private]
```

This function is called by a searchForUsersFrameThread when the "Search" button is clicked on a SearchFor↩ UsersFrame page.

Parameters

in	*ipOrDomain*	A string containing an IPv4 address or FreeDNS domain to search.
in	*searchUsername*	A string that contains the username of a user we are searching for.

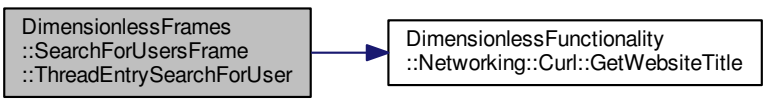
Definition at line 462 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::gaugeFindingUsers, DimensionlessFunctionality::Networking::Curl::↩ GetWebsiteTitle(), and idThreadSearchForUser.

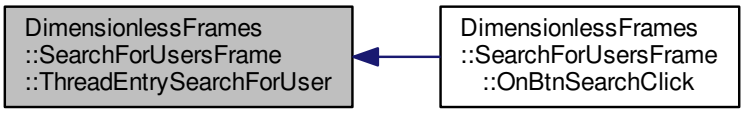
Referenced by OnBtnSearchClick().

```
463     {
464         wxThreadEvent event(wxEVT_THREAD, idThreadSearchForUser);
465 
466         event.SetPayload<string>(GetWebsiteTitle(ipOrDomain+":1337/user.html"));
467         this->gaugeFindingUsers->SetValue(100);
468 
469         wxPostEvent(this, event);
470     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.14.3.8 ThreadEntrySearchForUsers()

```
void DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUsers (
            const std::vector< string > & ips,
            const std::string searchUsername = "" )  [private]
```

This function is called by a searchForUsersFrameThread when the "Search" button is clicked on a SearchFor↩ UsersFrame page.

Parameters

in	*ips*	A string vector of IPv4 addresses to search.
in	*searchUsername*	A string that contains the username of a user we are searching for.

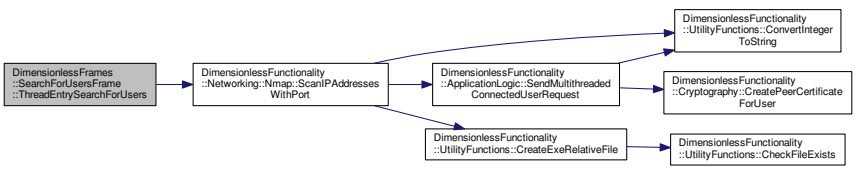
Definition at line 447 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::gaugeFindingUsers, idThreadSearchForUsers, and Dimensionless↩ Functionality::Networking::Nmap::ScanIPAddressesWithPort().

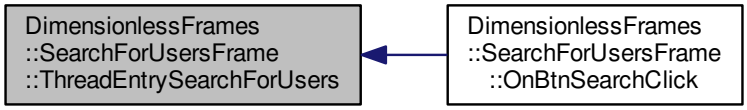
Referenced by OnBtnSearchClick().

```
448     {
449         wxThreadEvent event(wxEVT_THREAD, idThreadSearchForUsers);
450 
451         event.SetPayload< vector<string> >(ScanIPAddressesWithPort(searchUsername,
      ips, 1337));
452         this->gaugeFindingUsers->SetValue(100);
453 
454         wxPostEvent(this, event);
455     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.14.3.9 ThreadEntrySendConnectionRequest()

```
void DimensionlessFrames::SearchForUsersFrame::ThreadEntrySendConnectionRequest (
            const std::shared_ptr< ConnectedUser > & targetUser )  [private]
```

This function is called by a searchForUsersFrameThread when the "Send Connection Request" button is clicked on a SearchForUsersFrame page.

Parameters

in	*targetUser*	A smart pointer reference to the ConnectedUser object to send a connection request to.

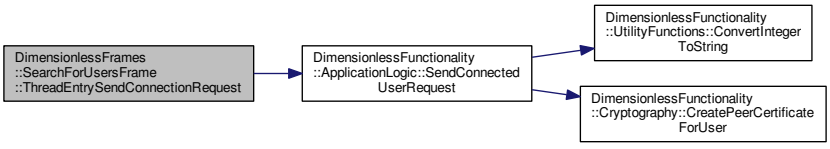
Definition at line 491 of file SearchForUsersFrame.cpp.

References idThreadSendConnectionRequest, and DimensionlessFunctionality::ApplicationLogic::Send↩ ConnectedUserRequest().

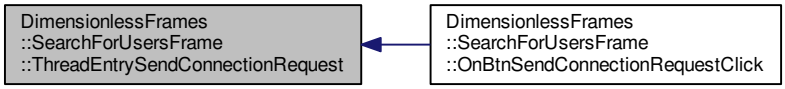
Referenced by OnBtnSendConnectionRequestClick().

```
492     {
493         wxThreadEvent event(wxEVT_THREAD, idThreadSendConnectionRequest);
494 
495         SendConnectedUserRequest(Request::SendConnectionRequest, targetUser);
496 
497         wxPostEvent(this, event);
498     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.14.4 Member Data Documentation

12.14.4.1 cities

```
std::vector< string > DimensionlessFrames::SearchForUsersFrame::cities  [static], [private]
```

This string vector stores the cities for the currently selected country.

Definition at line 78 of file SearchForUsersFrame.h.

Referenced by OnChSearchCountryChoiceSelected(), and SearchForUsersFrame().

12.14.4.2 foundUsers

```
std::vector< std::shared_ptr< ConnectedUser > > DimensionlessFrames::SearchForUsersFrame↩
::foundUsers  [static], [private]
```

This smart pointer vector of ConnectedUser objects stores a collection of users found from a search.

Definition at line 81 of file SearchForUsersFrame.h.

Referenced by OnBtnSearchClick(), OnBtnSendConnectionRequestClick(), and DimensionlessFrames::Search↩ForUsersFrame::SearchForUsersFrameThreadHandler::operator()().

12.14.4.3 mapCountriesCities

```
std::map< string, std::vector< string > > DimensionlessFrames::SearchForUsersFrame::map↩
CountriesCities  [static], [private]
```

This map of strings and string vectors stores the collection of countries and cities available for searching.

Definition at line 75 of file SearchForUsersFrame.h.

Referenced by OnBtnSearchClick(), OnChSearchCountryChoiceSelected(), and SearchForUsersFrame().

12.14.4.4 searchForUsersFrameThread

`std::thread DimensionlessFrames::SearchForUsersFrame::searchForUsersFrameThread [static], [private]`

This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Definition at line 84 of file SearchForUsersFrame.h.

Referenced by OnBtnCancelSearchClick(), OnBtnSearchClick(), and OnBtnSendConnectionRequestClick().

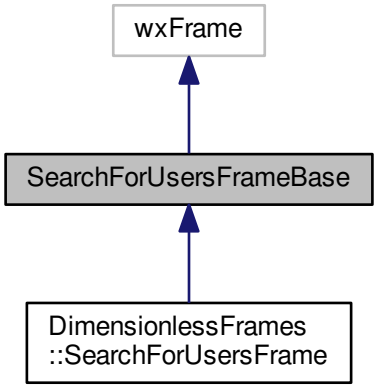
The documentation for this class was generated from the following files:

- DimensionlessClient/SearchForUsersFrame.h
- DimensionlessClient/SearchForUsersFrame.cpp

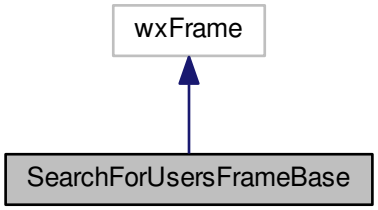
12.15 SearchForUsersFrameBase Class Reference

`#include <Dimensionless.h>`

Inheritance diagram for SearchForUsersFrameBase:



Collaboration diagram for SearchForUsersFrameBase:



Public Member Functions

- wxStaticText * GetStSearchType ()
- wxChoice * GetChSearchType ()
- wxStaticText * GetStSearchIPv4AddressOrDomain ()
- wxTextCtrl * GetTcSearchIPv4AddressOrDomain ()
- wxStaticText * GetStSearchOnionDomain ()
- wxTextCtrl * GetTcSearchOnionDomain ()
- wxStaticText * GetStSearchUsername ()
- wxTextCtrl * GetTcSearchUsername ()
- wxStaticText * GetStSearchCountryCity ()
- wxChoice * GetChSearchCountry ()
- wxChoice * GetChSearchCity ()
- wxButton * GetBtnSearch ()
- wxButton * GetBtnCancelSearch ()
- wxStaticText * GetStFoundUsersList ()
- wxGauge * GetGaugeFindingUsers ()
- wxListBox * GetLbFoundUsers ()
- wxButton * GetBtnSendConnectionRequest ()
- SearchForUsersFrameBase (wxWindow *parent, wxWindowID id=wxID_ANY, const wxString &title=_↩ ("Search for Users"), const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(700, 700), long style=wxFRAME_FLOAT_ON_PARENT|wxCAPTION|wxCLOSE_BOX)
- virtual ~SearchForUsersFrameBase ()

Protected Member Functions

- virtual void OnChSearchTypeChoiceSelected (wxCommandEvent &event)
- virtual void OnChSearchCountryChoiceSelected (wxCommandEvent &event)
- virtual void OnBtnSearchClick (wxCommandEvent &event)
- virtual void OnBtnCancelSearchClick (wxCommandEvent &event)
- virtual void OnBtnSendConnectionRequestClick (wxCommandEvent &event)

Protected Attributes

- wxStaticText * stSearchType
- wxChoice * chSearchType
- wxStaticText * stSearchIPv4AddressOrDomain
- wxTextCtrl * tcSearchIPv4AddressOrDomain
- wxStaticText * stSearchOnionDomain
- wxTextCtrl * tcSearchOnionDomain
- wxStaticText * stSearchUsername
- wxTextCtrl * tcSearchUsername
- wxStaticText * stSearchCountryCity
- wxChoice * chSearchCountry
- wxChoice * chSearchCity
- wxButton * btnSearch
- wxButton * btnCancelSearch
- wxStaticText * stFoundUsersList
- wxGauge * gaugeFindingUsers
- wxListBox * lbFoundUsers
- wxButton * btnSendConnectionRequest

12.15.1 Detailed Description

Definition at line 164 of file Dimensionless.h.

12.15.2 Constructor & Destructor Documentation

12.15.2.1 SearchForUsersFrameBase()

```
SearchForUsersFrameBase::SearchForUsersFrameBase (
            wxWindow * parent,
            wxWindowID id = wxID_ANY,
            const wxString & title = _("Search for Users"),
            const wxPoint & pos = wxDefaultPosition,
            const wxSize & size = wxSize(700,700),
            long style = wxFRAME_FLOAT_ON_PARENT|wxCAPTION|wxCLOSE_BOX )
```

Definition at line 329 of file Dimensionless.cpp.

References bBitmapLoaded, btnCancelSearch, btnSearch, btnSendConnectionRequest, chSearchCity, chSearchCountry, chSearchType, gaugeFindingUsers, lbFoundUsers, OnBtnCancelSearchClick(), OnBtnSearchClick(), OnBtnSendConnectionRequestClick(), OnChSearchCountryChoiceSelected(), OnChSearchTypeChoiceSelected(), stFoundUsersList, stSearchCountryCity, stSearchIPv4AddressOrDomain, stSearchOnionDomain, stSearchType, stSearchUsername, tcSearchIPv4AddressOrDomain, tcSearchOnionDomain, tcSearchUsername, WXC_FROM_DIP, and wxCrafterAR3ID5InitBitmapResources().

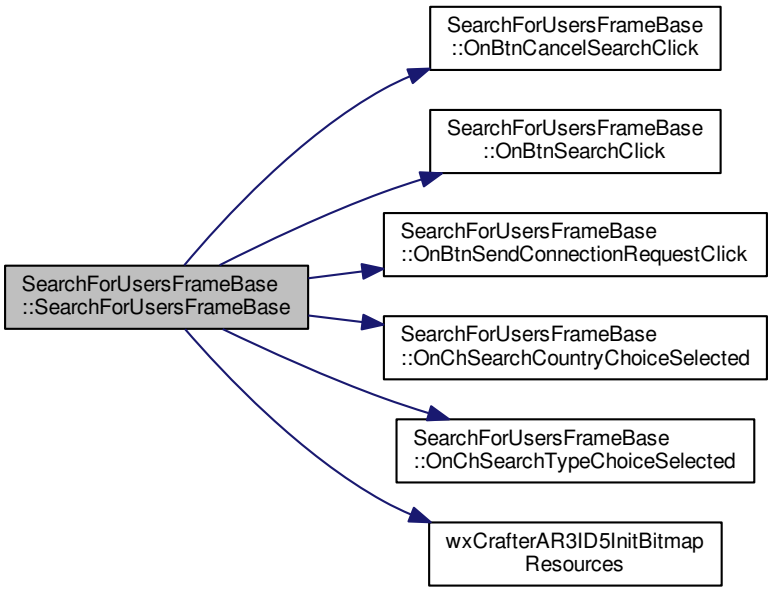
```
330     : wxFrame(parent, id, title, pos, size, style)
331 {
332     if ( !bBitmapLoaded ) {
333         // We need to initialise the default bitmap handler
334         wxXmlResource::Get()->AddHandler(new wxBitmapXmlHandler);
335         wxCrafterAR3ID5InitBitmapResources();
336         bBitmapLoaded = true;
337     }
338
339     wxBoxSizer* bszrSearchForUsers = new wxBoxSizer(wxVERTICAL);
340     this->SetSizer(bszrSearchForUsers);
341
342     wxBoxSizer* bszrSearchType = new wxBoxSizer(wxVERTICAL);
343
344     bszrSearchForUsers->Add(bszrSearchType, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
345
346     stSearchType = new wxStaticText(this, wxID_ANY, _("Search Type:"), wxDefaultPosition,
    wxDLG_UNIT(this, wxSize(-1,-1)), 0);
347
348     bszrSearchType->Add(stSearchType, 0, wxALL, WXC_FROM_DIP(5));
349
350     wxArrayString chSearchTypeArr;
351     chSearchTypeArr.Add(wxT("By IPv4 Address / Domain"));
352     chSearchTypeArr.Add(wxT("Onion Domain"));
353     chSearchTypeArr.Add(wxT("Country, City"));
354     chSearchType = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-
    1)), chSearchTypeArr, 0);
355     chSearchType->SetSelection(0);
356
357     bszrSearchType->Add(chSearchType, 0, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
358
359     wxBoxSizer* bszrSearchIPv4AddressOrRegularDomain = new wxBoxSizer(wxVERTICAL);
360
361     bszrSearchForUsers->Add(bszrSearchIPv4AddressOrRegularDomain, 0, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
362
363     stSearchIPv4AddressOrDomain = new wxStaticText(this, wxID_ANY, _("Search
     IPv4 Address / Domain:"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
364
```

```
365     bszrSearchIPv4AddressOrRegularDomain->Add(stSearchIPv4AddressOrDomain, 0,
      wxALL|wxALIGN_LEFT|wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
366
367     tcSearchIPv4AddressOrDomain = new wxTextCtrl(this, wxID_ANY, wxT(""),
      wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
368     #if wxVERSION_NUMBER >= 3000
369     tcSearchIPv4AddressOrDomain->SetHint(_("Enter IPv4 address or domain to
       search"));
370     #endif
371
372     bszrSearchIPv4AddressOrRegularDomain->Add(tcSearchIPv4AddressOrDomain, 0,
      wxALL|wxEXPAND, WXC_FROM_DIP(5));
373
374     wxBoxSizer* bszrSearchOnionDomain = new wxBoxSizer(wxVERTICAL);
375
376     bszrSearchForUsers->Add(bszrSearchOnionDomain, 0, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
377
378     stSearchOnionDomain = new wxStaticText(this, wxID_ANY, _("Search Onion Domain:"),
      wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
379     stSearchOnionDomain->Hide();
380
381     bszrSearchOnionDomain->Add(stSearchOnionDomain, 0, wxALL|wxALIGN_LEFT|
      wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
382
383     tcSearchOnionDomain = new wxTextCtrl(this, wxID_ANY, wxT(""), wxDefaultPosition,
      wxDLG_UNIT(this, wxSize(-1,-1)), 0);
384     tcSearchOnionDomain->Hide();
385     tcSearchOnionDomain->Enable(false);
386     #if wxVERSION_NUMBER >= 3000
387     tcSearchOnionDomain->SetHint(_("Enter onion domain to search"));
388     #endif
389
390     bszrSearchOnionDomain->Add(tcSearchOnionDomain, 0, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
391
392     wxBoxSizer* bszrSearchUsername = new wxBoxSizer(wxVERTICAL);
393
394     bszrSearchForUsers->Add(bszrSearchUsername, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
395
396     stSearchUsername = new wxStaticText(this, wxID_ANY, _("Search Username:"),
      wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
397     stSearchUsername->Hide();
398
399     bszrSearchUsername->Add(stSearchUsername, 0, wxALL|wxALIGN_LEFT|wxALIGN_CENTER_VERTICAL
      , WXC_FROM_DIP(5));
400
401     tcSearchUsername = new wxTextCtrl(this, wxID_ANY, wxT(""), wxDefaultPosition,
      wxDLG_UNIT(this, wxSize(-1,-1)), 0);
402     tcSearchUsername->Hide();
403     tcSearchUsername->Enable(false);
404     #if wxVERSION_NUMBER >= 3000
405     tcSearchUsername->SetHint(_("Enter username to search for (or none to see all visible
       users)"));
406     #endif
407
408     bszrSearchUsername->Add(tcSearchUsername, 0, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
409
410     wxBoxSizer* bszrSearchCountryCity = new wxBoxSizer(wxVERTICAL);
411
412     bszrSearchForUsers->Add(bszrSearchCountryCity, 0, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
413
414     stSearchCountryCity = new wxStaticText(this, wxID_ANY, _("Search Country, City:"),
      wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
415     stSearchCountryCity->Hide();
416
417     bszrSearchCountryCity->Add(stSearchCountryCity, 0, wxALL|wxALIGN_LEFT|
      wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
418
419     wxBoxSizer* bszrCountryCityChoices = new wxBoxSizer(wxHORIZONTAL);
420
421     bszrSearchCountryCity->Add(bszrCountryCityChoices, 0, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
422
423     wxArrayString chSearchCountryArr;
424     chSearchCountry = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this,
      wxSize(-1,-1)), chSearchCountryArr, 0);
425     chSearchCountry->Hide();
426     chSearchCountry->Enable(false);
427
428     bszrCountryCityChoices->Add(chSearchCountry, 1, wxALL|wxEXPAND,
      WXC_FROM_DIP(5));
429
430     wxArrayString chSearchCityArr;
431     chSearchCity = new wxChoice(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-
```

```
    1)), chSearchCityArr, 0);
432     chSearchCity->Hide();
433     chSearchCity->Enable(false);
434 
435     bszrCountryCityChoices->Add(chSearchCity, 1, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
436 
437     wxBoxSizer* bszrSearch = new wxBoxSizer(wxHORIZONTAL);
438 
439     bszrSearchForUsers->Add(bszrSearch, 0, wxALL|wxALIGN_RIGHT, WXC_FROM_DIP(5));
440 
441     btnSearch = new wxButton(this, wxID_ANY, _("Search"), wxDefaultPosition, wxDLG_UNIT(this,
    wxSize(-1,-1)), 0);
442 
443     bszrSearch->Add(btnSearch, 0, wxALL|wxALIGN_RIGHT, WXC_FROM_DIP(5));
444 
445     btnCancelSearch = new wxButton(this, wxID_ANY, _("Cancel Search"), wxDefaultPosition,
    wxDLG_UNIT(this, wxSize(-1,-1)), 0);
446     btnCancelSearch->Hide();
447     btnCancelSearch->Enable(false);
448 
449     bszrSearch->Add(btnCancelSearch, 0, wxALL|wxALIGN_RIGHT,
    WXC_FROM_DIP(5));
450 
451     wxBoxSizer* bszrUsers = new wxBoxSizer(wxVERTICAL);
452 
453     bszrSearchForUsers->Add(bszrUsers, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
454 
455     wxBoxSizer* bszrFoundUsers = new wxBoxSizer(wxHORIZONTAL);
456 
457     bszrUsers->Add(bszrFoundUsers, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
458 
459     stFoundUsersList = new wxStaticText(this, wxID_ANY, _("Found Users List:"),
    wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
460 
461     bszrFoundUsers->Add(stFoundUsersList, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|
    wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
462 
463     gaugeFindingUsers = new wxGauge(this, wxID_ANY, 100, wxDefaultPosition, wxDLG_UNIT(
    this, wxSize(-1,-1)), wxGA_HORIZONTAL);
464     gaugeFindingUsers->Hide();
465     gaugeFindingUsers->SetValue(0);
466 
467     bszrFoundUsers->Add(gaugeFindingUsers, 0, wxALL,
    WXC_FROM_DIP(5));
468 
469     wxArrayString lbFoundUsersArr;
470     lbFoundUsers = new wxListBox(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,
    -1)), lbFoundUsersArr, wxLB_SINGLE|wxTRANSPARENT_WINDOW|wxVSCROLL);
471 
472     bszrUsers->Add(lbFoundUsers, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
473 
474     btnSendConnectionRequest = new wxButton(this, wxID_ANY, _("Send Connection
     Request"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
475 
476     bszrSearchForUsers->Add(btnSendConnectionRequest, 0, wxALL|wxALIGN_RIGHT,
    WXC_FROM_DIP(5));
477 
478     SetName(wxT("SearchForUsersFrameBase"));
479     SetMinClientSize(wxSize(700,700));
480     SetSize(700,700);
481     if (GetSizer()) {
482          GetSizer()->Fit(this);
483     }
484     if(GetParent()) {
485         CentreOnParent(wxBOTH);
486     } else {
487         CentreOnScreen(wxBOTH);
488     }
489 #if wxVERSION_NUMBER >= 2900
490     if(!wxPersistenceManager::Get().Find(this)) {
491         wxPersistenceManager::Get().RegisterAndRestore(this);
492     } else {
493         wxPersistenceManager::Get().Restore(this);
494     }
495 #endif
496     // Connect events
497     chSearchType->Connect(wxEVT_COMMAND_CHOICE_SELECTED, wxCommandEventHandler(
    SearchForUsersFrameBase::OnChSearchTypeChoiceSelected)
    , NULL, this);
498     chSearchCountry->Connect(wxEVT_COMMAND_CHOICE_SELECTED, wxCommandEventHandler(
    SearchForUsersFrameBase::OnChSearchCountryChoiceSelected
    ), NULL, this);
499     btnSearch->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
    SearchForUsersFrameBase::OnBtnSearchClick), NULL, this);
500     btnCancelSearch->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
    SearchForUsersFrameBase::OnBtnCancelSearchClick), NULL, this
```

```
      );
501     btnSendConnectionRequest->Connect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(SearchForUsersFrameBase::OnBtnSendConnectionRequestClick
      ), NULL, this);
502
503 }
```

Here is the call graph for this function:



12.15.2.2 ~SearchForUsersFrameBase()

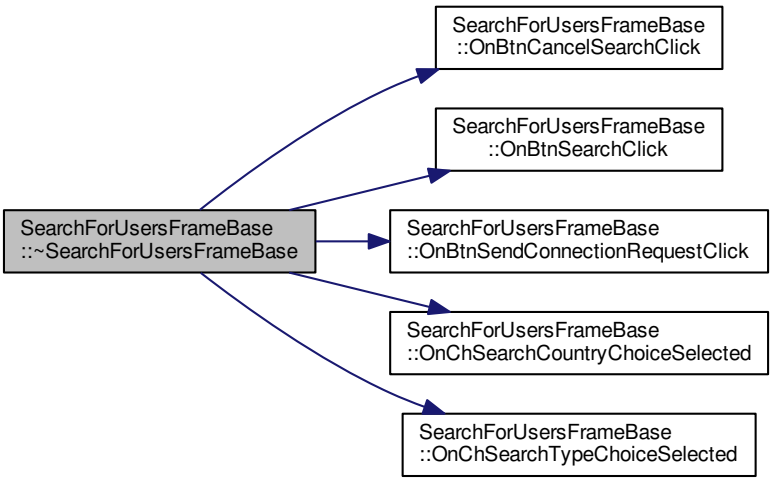
```
SearchForUsersFrameBase::~SearchForUsersFrameBase ( )  [virtual]
```

Definition at line 505 of file Dimensionless.cpp.

References btnCancelSearch, btnSearch, btnSendConnectionRequest, chSearchCountry, chSearchType, OnBtn↩CancelSearchClick(), OnBtnSearchClick(), OnBtnSendConnectionRequestClick(), OnChSearchCountryChoice↩Selected(), and OnChSearchTypeChoiceSelected().

```
506 {
507     chSearchType->Disconnect(wxEVT_COMMAND_CHOICE_SELECTED, wxCommandEventHandler(
      SearchForUsersFrameBase::OnChSearchTypeChoiceSelected)
      , NULL, this);
508     chSearchCountry->Disconnect(wxEVT_COMMAND_CHOICE_SELECTED, wxCommandEventHandler(
      SearchForUsersFrameBase::OnChSearchCountryChoiceSelected
      ), NULL, this);
509     btnSearch->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      SearchForUsersFrameBase::OnBtnSearchClick), NULL, this);
510     btnCancelSearch->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      SearchForUsersFrameBase::OnBtnCancelSearchClick), NULL, this
      );
511     btnSendConnectionRequest->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(SearchForUsersFrameBase::OnBtnSendConnectionRequestClick
      ), NULL, this);
512
513 }
```

Here is the call graph for this function:



12.15.3 Member Function Documentation

12.15.3.1 GetBtnCancelSearch()

```
wxButton* SearchForUsersFrameBase::GetBtnCancelSearch ( )  [inline]
```

Definition at line 205 of file Dimensionless.h.

```
205 { return btnCancelSearch; }
```

12.15.3.2 GetBtnSearch()

```
wxButton* SearchForUsersFrameBase::GetBtnSearch ( )  [inline]
```

Definition at line 204 of file Dimensionless.h.

```
204 { return btnSearch; }
```

12.15.3.3 GetBtnSendConnectionRequest()

```
wxButton* SearchForUsersFrameBase::GetBtnSendConnectionRequest ( )  [inline]
```

Definition at line 209 of file Dimensionless.h.

```
209 { return btnSendConnectionRequest; }
```

12.15.3.4 GetChSearchCity()

```
wxChoice* SearchForUsersFrameBase::GetChSearchCity ( )  [inline]
```

Definition at line 203 of file Dimensionless.h.

```
203 { return chSearchCity; }
```

12.15.3.5 GetChSearchCountry()

```
wxChoice* SearchForUsersFrameBase::GetChSearchCountry ( )  [inline]
```

Definition at line 202 of file Dimensionless.h.

```
202 { return chSearchCountry; }
```

12.15.3.6 GetChSearchType()

```
wxChoice* SearchForUsersFrameBase::GetChSearchType ( )  [inline]
```

Definition at line 194 of file Dimensionless.h.

```
194 { return chSearchType; }
```

12.15.3.7 GetGaugeFindingUsers()

```
wxGauge* SearchForUsersFrameBase::GetGaugeFindingUsers ( )  [inline]
```

Definition at line 207 of file Dimensionless.h.

```
207 { return gaugeFindingUsers; }
```

12.15.3.8 GetLbFoundUsers()

```
wxListBox* SearchForUsersFrameBase::GetLbFoundUsers ( )  [inline]
```

Definition at line 208 of file Dimensionless.h.

```
208 { return lbFoundUsers; }
```

12.15.3.9 GetStFoundUsersList()

```
wxStaticText* SearchForUsersFrameBase::GetStFoundUsersList ( )  [inline]
```

Definition at line 206 of file Dimensionless.h.

```
206 { return stFoundUsersList; }
```

12.15.3.10 GetStSearchCountryCity()

```
wxStaticText* SearchForUsersFrameBase::GetStSearchCountryCity ( )  [inline]
```

Definition at line 201 of file Dimensionless.h.

```
201 { return stSearchCountryCity; }
```

12.15.3.11 GetStSearchIPv4AddressOrDomain()

```
wxStaticText* SearchForUsersFrameBase::GetStSearchIPv4AddressOrDomain ( )  [inline]
```

Definition at line 195 of file Dimensionless.h.

```
195 { return stSearchIPv4AddressOrDomain; }
```

12.15.3.12 GetStSearchOnionDomain()

```
wxStaticText* SearchForUsersFrameBase::GetStSearchOnionDomain ( )  [inline]
```

Definition at line 197 of file Dimensionless.h.

```
197 { return stSearchOnionDomain; }
```

12.15.3.13 GetStSearchType()

```
wxStaticText* SearchForUsersFrameBase::GetStSearchType ( )  [inline]
```

Definition at line 193 of file Dimensionless.h.

```
193 { return stSearchType; }
```

12.15.3.14 GetStSearchUsername()

```
wxStaticText* SearchForUsersFrameBase::GetStSearchUsername ( )  [inline]
```

Definition at line 199 of file Dimensionless.h.

```
199 { return stSearchUsername; }
```

12.15.3.15 GetTcSearchIPv4AddressOrDomain()

```
wxTextCtrl* SearchForUsersFrameBase::GetTcSearchIPv4AddressOrDomain ( )  [inline]
```

Definition at line 196 of file Dimensionless.h.

```
196 { return tcSearchIPv4AddressOrDomain; }
```

12.15.3.16 GetTcSearchOnionDomain()

```
wxTextCtrl* SearchForUsersFrameBase::GetTcSearchOnionDomain ( )  [inline]
```

Definition at line 198 of file Dimensionless.h.

```
198 { return tcSearchOnionDomain; }
```

12.15.3.17 GetTcSearchUsername()

```
wxTextCtrl* SearchForUsersFrameBase::GetTcSearchUsername ( )  [inline]
```

Definition at line 200 of file Dimensionless.h.

```
200 { return tcSearchUsername; }
```

12.15.3.18 OnBtnCancelSearchClick()

```
virtual void SearchForUsersFrameBase::OnBtnCancelSearchClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

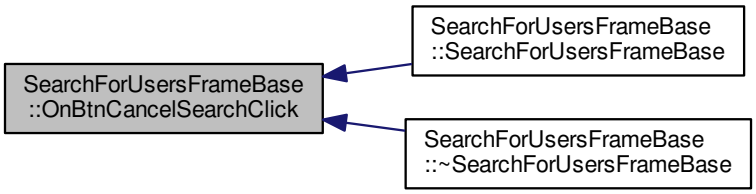
Reimplemented in DimensionlessFrames::SearchForUsersFrame.

Definition at line 189 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

```
189 { event.Skip(); }
```

Here is the caller graph for this function:



12.15.3.19 OnBtnSearchClick()

```
virtual void SearchForUsersFrameBase::OnBtnSearchClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

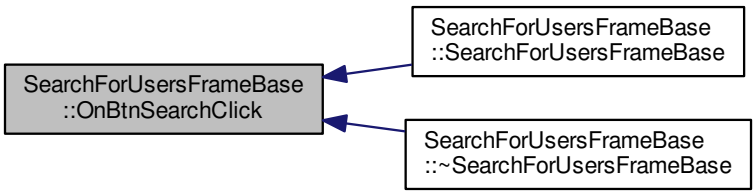
Reimplemented in DimensionlessFrames::SearchForUsersFrame.

Definition at line 188 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

```
188 { event.Skip(); }
```

Here is the caller graph for this function:



12.15.3.20 OnBtnSendConnectionRequestClick()

```
virtual void SearchForUsersFrameBase::OnBtnSendConnectionRequestClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

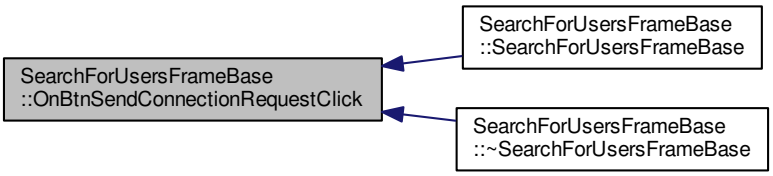
Reimplemented in DimensionlessFrames::SearchForUsersFrame.

Definition at line 190 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

```
190 { event.Skip(); }
```

Here is the caller graph for this function:



12.15.3.21 OnChSearchCountryChoiceSelected()

```
virtual void SearchForUsersFrameBase::OnChSearchCountryChoiceSelected (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

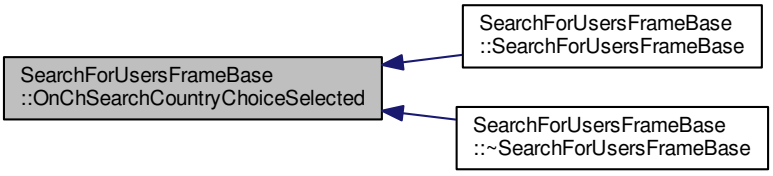
Reimplemented in DimensionlessFrames::SearchForUsersFrame.

Definition at line 187 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

```
187 { event.Skip(); }
```

Here is the caller graph for this function:



12.15.3.22 OnChSearchTypeChoiceSelected()

```
virtual void SearchForUsersFrameBase::OnChSearchTypeChoiceSelected (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

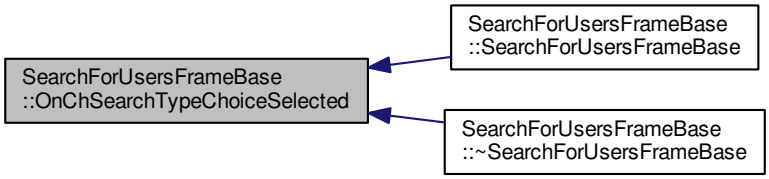
Reimplemented in DimensionlessFrames::SearchForUsersFrame.

Definition at line 186 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

```
186 { event.Skip(); }
```

Here is the caller graph for this function:



12.15.4 Member Data Documentation

12.15.4.1 btnCancelSearch

```
wxButton* SearchForUsersFrameBase::btnCancelSearch  [protected]
```

Definition at line 179 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

12.15.4.2 btnSearch

```
wxButton* SearchForUsersFrameBase::btnSearch  [protected]
```

Definition at line 178 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

12.15.4.3 btnSendConnectionRequest

```
wxButton* SearchForUsersFrameBase::btnSendConnectionRequest  [protected]
```

Definition at line 183 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

12.15.4.4 chSearchCity

```
wxChoice* SearchForUsersFrameBase::chSearchCity  [protected]
```

Definition at line 177 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::OnChSearchCountryChoiceSelected(), DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrame(), and SearchForUsersFrameBase().

12.15.4.5 chSearchCountry

```
wxChoice* SearchForUsersFrameBase::chSearchCountry  [protected]
```

Definition at line 176 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::OnChSearchCountryChoiceSelected(), DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrame(), SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

12.15.4.6 chSearchType

```
wxChoice* SearchForUsersFrameBase::chSearchType  [protected]
```

Definition at line 168 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), SearchForUsersFrameBase(), and ~SearchForUsersFrameBase().

12.15.4.7 gaugeFindingUsers

```
wxGauge* SearchForUsersFrameBase::gaugeFindingUsers  [protected]
```

Definition at line 181 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnCancelSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), SearchForUsersFrameBase(), DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForTorUser(), DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUser(), and DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUsers().

12.15.4.8 lbFoundUsers

```
wxListBox* SearchForUsersFrameBase::lbFoundUsers  [protected]
```

Definition at line 182 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnBtnSendConnectionRequestClick(), DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), and SearchForUsersFrameBase().

12.15.4.9 stFoundUsersList

```
wxStaticText* SearchForUsersFrameBase::stFoundUsersList  [protected]
```

Definition at line 180 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase().

12.15.4.10 stSearchCountryCity

```
wxStaticText* SearchForUsersFrameBase::stSearchCountryCity  [protected]
```

Definition at line 175 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

12.15.4.11 stSearchIPv4AddressOrDomain

```
wxStaticText* SearchForUsersFrameBase::stSearchIPv4AddressOrDomain  [protected]
```

Definition at line 169 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

12.15.4.12 stSearchOnionDomain

```
wxStaticText* SearchForUsersFrameBase::stSearchOnionDomain  [protected]
```

Definition at line 171 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

12.15.4.13 stSearchType

```
wxStaticText* SearchForUsersFrameBase::stSearchType  [protected]
```

Definition at line 167 of file Dimensionless.h.

Referenced by SearchForUsersFrameBase().

12.15.4.14 stSearchUsername

```
wxStaticText* SearchForUsersFrameBase::stSearchUsername  [protected]
```

Definition at line 173 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

12.15.4.15 tcSearchIPv4AddressOrDomain

```
wxTextCtrl* SearchForUsersFrameBase::tcSearchIPv4AddressOrDomain  [protected]
```

Definition at line 170 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

12.15.4.16 tcSearchOnionDomain

```
wxTextCtrl* SearchForUsersFrameBase::tcSearchOnionDomain  [protected]
```

Definition at line 172 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

12.15.4.17 tcSearchUsername

```
wxTextCtrl* SearchForUsersFrameBase::tcSearchUsername  [protected]
```

Definition at line 174 of file Dimensionless.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::OnBtnSearchClick(), DimensionlessFrames::SearchForUsersFrame::OnChSearchTypeChoiceSelected(), and SearchForUsersFrameBase().

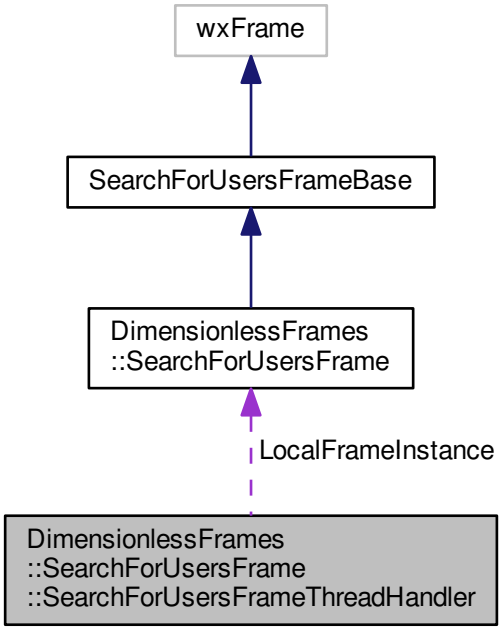
The documentation for this class was generated from the following files:

- DimensionlessClient/Dimensionless.h
- DimensionlessClient/Dimensionless.cpp

12.16 DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler Struct Reference

This struct handles events posted from our searchForUsersFrameThread.

Collaboration diagram for DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler:



Public Member Functions

- SearchForUsersFrameThreadHandler (SearchForUsersFrame *frame)
- void operator() (wxThreadEvent &evt)

Private Attributes

- SearchForUsersFrame * LocalFrameInstance = nullptr

  A raw pointer reference to a SearchForUsersFrame to handle events for.

12.16.1 Detailed Description

This struct handles events posted from our searchForUsersFrameThread.

Definition at line 49 of file SearchForUsersFrame.cpp.

12.16.2 Constructor & Destructor Documentation

12.16.2.1 SearchForUsersFrameThreadHandler()

```
DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::SearchForUsers↩
FrameThreadHandler (
            SearchForUsersFrame * frame )  [inline]
```

This constructor initializes a new SearchForUsersFrameThreadHandler object with a pointer to the current SearchForUsersFrame instance so it may be modified as required.

Parameters

in	*frame*	The raw pointer to a SearchForUsersFrame instance.

Definition at line 56 of file SearchForUsersFrame.cpp.

References LocalFrameInstance.

```
57             {
58                 // Set the local SearchForUsersFrame reference to that passed.
59                 LocalFrameInstance = frame;
60             }
```

12.16.3 Member Function Documentation

12.16.3.1 operator()()

```
void DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator() (
            wxThreadEvent & evt )  [inline]
```

This function remaps the () operator of SearchForUsersFrameThreadHandler to handle searchForUsersFrame↩ Thread events.

Parameters

in	*event*	A wxThreadEvent object posted by our searchForUsersFrameThread.

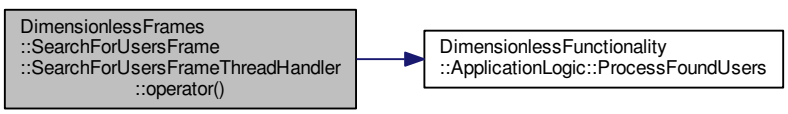
Definition at line 66 of file SearchForUsersFrame.cpp.

References SearchForUsersFrameBase::btnCancelSearch, SearchForUsersFrameBase::btnSearch, SearchFor↩ UsersFrameBase::btnSendConnectionRequest, SearchForUsersFrameBase::chSearchCity, SearchForUsers↩ FrameBase::chSearchCountry, SearchForUsersFrameBase::chSearchType, DimensionlessFrames::SearchFor↩ UsersFrame::foundUsers, SearchForUsersFrameBase::gaugeFindingUsers, idThreadSearchForTorUser, id↩ ThreadSearchForUser, idThreadSearchForUsers, idThreadSendConnectionRequest, SearchForUsersFrame↩ Base::lbFoundUsers, LocalFrameInstance, and DimensionlessFunctionality::ApplicationLogic::ProcessFound↩ Users().

```
67              {
68                  switch(evt.GetId()){
69                      case idThreadSearchForUsers:
70                      {
71                          //LocalFrameInstance->nbSendMessageBtn[evt.GetInt()]->Enable(true);
72                          vector<string> users = evt.GetPayload< vector<string> >();
73                          for(unsigned int i = 0; i < users.size(); i++){
74                              std::vector< std::shared_ptr<ConnectedUser> > temp =
    ProcessFoundUsers(users[i]);
75                              LocalFrameInstance->foundUsers.insert(
    LocalFrameInstance->foundUsers.begin(), temp.begin(), temp.end());
76                          }
77                          for(unsigned int i = 0; i < LocalFrameInstance->
    foundUsers.size(); i++){
78                              LocalFrameInstance->lbFoundUsers->Append(
    LocalFrameInstance->foundUsers[i]->toFoundString());
79                          }
80                          LocalFrameInstance->btnSearch->Enable(true);
81                          LocalFrameInstance->btnCancelSearch->Enable(false)
    ;
82                          LocalFrameInstance->btnCancelSearch->Hide();
83                          LocalFrameInstance->
    btnSendConnectionRequest->Enable(true);
84                          LocalFrameInstance->chSearchType->Enable(true);
85                          LocalFrameInstance->chSearchCountry->Enable(true);
86                          LocalFrameInstance->chSearchCity->Enable(true);
87                          LocalFrameInstance->gaugeFindingUsers->Hide();
88                          LocalFrameInstance->gaugeFindingUsers->SetValue(
    0);
89                          LocalFrameInstance->Layout();
90                          break;
91                      }
92                      case idThreadSearchForUser:
93                      {
94                          string users = evt.GetPayload<string>();
95                          std::vector< std::shared_ptr<ConnectedUser> > temp =
    ProcessFoundUsers(users);
96                          LocalFrameInstance->foundUsers.insert(
    LocalFrameInstance->foundUsers.begin(), temp.begin(), temp.end());
97                          for(unsigned int i = 0; i < LocalFrameInstance->
    foundUsers.size(); i++){
98                              LocalFrameInstance->lbFoundUsers->Append(
    LocalFrameInstance->foundUsers[i]->toFoundString());
99                          }
100                          LocalFrameInstance->btnSearch->Enable(true);
101                          LocalFrameInstance->btnCancelSearch->Enable(false)
    ;
102                          LocalFrameInstance->btnCancelSearch->Hide();
103                          LocalFrameInstance->
    btnSendConnectionRequest->Enable(true);
104                          LocalFrameInstance->chSearchType->Enable(true);
```

```
105                         LocalFrameInstance->chSearchCountry->Enable(true);
106                         LocalFrameInstance->chSearchCity->Enable(true);
107                         LocalFrameInstance->gaugeFindingUsers->Hide();
108                         LocalFrameInstance->gaugeFindingUsers->SetValue(
    0);
109                         LocalFrameInstance->Layout();
110                         break;
111                     }
112                     case idThreadSearchForTorUser:
113                     {
114                         string users = evt.GetPayload<string>();
115                         std::vector< std::shared_ptr<ConnectedUser> > temp =
    ProcessFoundUsers(users);
116                         LocalFrameInstance->foundUsers.insert(
    LocalFrameInstance->foundUsers.begin(), temp.begin(), temp.end());
117                         for(unsigned int i = 0; i < LocalFrameInstance->
    foundUsers.size(); i++){
118                             LocalFrameInstance->lbFoundUsers->Append(
    LocalFrameInstance->foundUsers[i]->toFoundString());
119                         }
120                         LocalFrameInstance->btnSearch->Enable(true);
121                         LocalFrameInstance->btnCancelSearch->Enable(false)
    ;
122                         LocalFrameInstance->btnCancelSearch->Hide();
123                         LocalFrameInstance->
    btnSendConnectionRequest->Enable(true);
124                         LocalFrameInstance->chSearchType->Enable(true);
125                         LocalFrameInstance->chSearchCountry->Enable(true);
126                         LocalFrameInstance->chSearchCity->Enable(true);
127                         LocalFrameInstance->gaugeFindingUsers->Hide();
128                         LocalFrameInstance->gaugeFindingUsers->SetValue(
    0);
129                         LocalFrameInstance->Layout();
130                         break;
131                     }
132                     case idThreadSendConnectionRequest:
133                     {
134                         LocalFrameInstance->btnSearch->Enable(false);
135                         LocalFrameInstance->
    btnSendConnectionRequest->Enable(false);
136                         LocalFrameInstance->chSearchType->Enable(false);
137                         LocalFrameInstance->chSearchCountry->Enable(false)
    ;
138                         LocalFrameInstance->chSearchCity->Enable(false);
139                         LocalFrameInstance->gaugeFindingUsers->Show();
140                         LocalFrameInstance->gaugeFindingUsers->SetValue(
    0);
141                         LocalFrameInstance->Layout();
142                         wxMessageBox("Connection request sent!","Connection Request",wxOK|wxCENTRE,
    LocalFrameInstance);
143                         break;
144                     }
145                     default:
146                     {
147                         break;
148                     }
149                 }
150             }
```

Here is the call graph for this function:



12.16.4 Member Data Documentation

12.16.4.1 LocalFrameInstance

```
SearchForUsersFrame* DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThread↩
Handler::LocalFrameInstance = nullptr  [private]
```

A raw pointer reference to a SearchForUsersFrame to handle events for.

Definition at line 153 of file SearchForUsersFrame.cpp.

Referenced by operator()(), and SearchForUsersFrameThreadHandler().

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

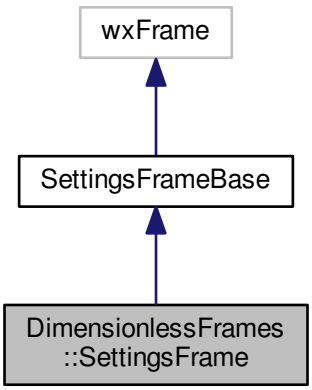
- DimensionlessClient/SearchForUsersFrame.cpp

12.17 DimensionlessFrames::SettingsFrame Class Reference

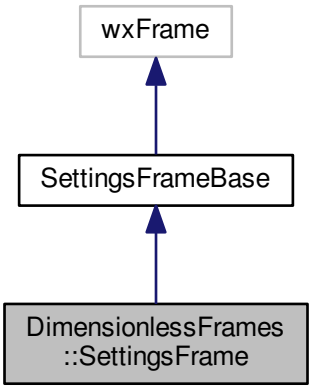
This class creates a SettingsFrame that is displayed when the currently logged in user wishes to view or modify their settings.

```
#include <SettingsFrame.h>
```

Inheritance diagram for DimensionlessFrames::SettingsFrame:



Collaboration diagram for DimensionlessFrames::SettingsFrame:



Classes

- struct SettingsFrameThreadHandler

  This struct handles events posted from our settingsFrameThread.

Public Member Functions

- SettingsFrame (MainFrame *parent)
- virtual ~SettingsFrame ()

Protected Member Functions

- virtual void OnRbDefaultWebBrowserSelected (wxCommandEvent &event)
- virtual void OnRbPublicSearchVisibilitySelected (wxCommandEvent &event)
- virtual void OnRbTorSelected (wxCommandEvent &event)
- virtual void OnBtnCheckForClientUpdatesClick (wxCommandEvent &event)
- virtual void OnBtnTestAndSaveDomainClick (wxCommandEvent &event)

Static Private Attributes

- static std::thread settingsFrameThread

  This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Additional Inherited Members

12.17.1 Detailed Description

This class creates a SettingsFrame that is displayed when the currently logged in user wishes to view or modify their settings.

Definition at line 45 of file SettingsFrame.h.

12.17.2 Constructor & Destructor Documentation

12.17.2.1 SettingsFrame()

```
DimensionlessFrames::SettingsFrame::SettingsFrame (
            MainFrame * parent )
```

This constructor initializes a new SettingsFrame.

Parameters

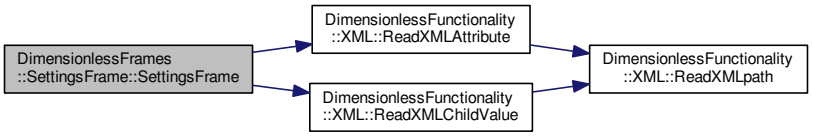
in	*parent*	The parent window of the one to be created.

Definition at line 79 of file SettingsFrame.cpp.

References DimensionlessFunctionality::XML::ReadXMLAttribute(), DimensionlessFunctionality::XML::ReadXMLChildValue(), SettingsFrameBase::scUpdatePeriod, SettingsFrameBase::tcAPIKey, and SettingsFrameBase::tcFQDN.

```
79                                              : SettingsFrameBase(parent)
80     {
81         this->SetMinClientSize(wxSize(650,550));
82         this->SetSize(650,550);
83         this->CentreOnParent(wxBOTH);
84         tcFQDN->SetValue(ReadXMLChildValue("./Users/"+
       CurrentUserSession::currentUser->username+"/data/"+CurrentUserSession::currentUser->username+".xml", "User", "FreeDNSAf
85         tcAPIKey->SetValue(ReadXMLAttribute("./Users/"+
       CurrentUserSession::currentUser->username+"/data/"+CurrentUserSession::currentUser->username+".xml", "User", "FreeDNSAf
       APIKey"));
86         scUpdatePeriod->SetValue(atoi(ReadXMLAttribute("./Users/"+
       CurrentUserSession::currentUser->username+"/data/"+CurrentUserSession::currentUser->username+".xml", "User", "
       FreeDNSAfraidOrgDomain", "UpdatePeriod").c_str()));
87     }
```

Here is the call graph for this function:



12.17.2.2 ~SettingsFrame()

```
DimensionlessFrames::SettingsFrame::~SettingsFrame ( )  [virtual]
```

This destructor currently does nothing.

Definition at line 92 of file SettingsFrame.cpp.

```
93     {
94     }
```

12.17.3 Member Function Documentation

12.17.3.1 OnBtnCheckForClientUpdatesClick()

```
void DimensionlessFrames::SettingsFrame::OnBtnCheckForClientUpdatesClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Check For Client Updates" button is clicked on a SettingsFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Check For Client Updates" button is clicked.

Reimplemented from SettingsFrameBase.

Definition at line 100 of file SettingsFrame.cpp.

```
101     {
102         //php/curl
103         wxMessageBox("Not yet implemented - coming soon.","Coming Soon");
104     }
```

12.17.3.2 OnBtnTestAndSaveDomainClick()

```
void DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Test And Save Domain" button is clicked on a SettingsFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Test And Save Domain" button is clicked.

Reimplemented from SettingsFrameBase.

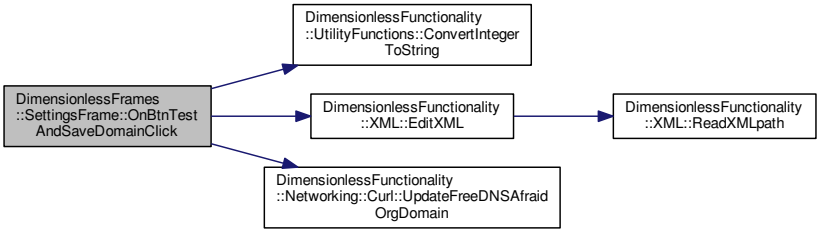
Definition at line 110 of file SettingsFrame.cpp.

References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString(), DimensionlessFunctionality::XML::EditXML(), SettingsFrameBase::scUpdatePeriod, SettingsFrameBase::stFreeDNSTestResult, SettingsFrameBase::tcAPIKey, SettingsFrameBase::tcFQDN, and DimensionlessFunctionality::Networking::Curl::UpdateFreeDNSAfraidOrgDomain().

```
111     {
112         //php/curl
113         string result = UpdateFreeDNSAfraidOrgDomain("
```

```
      https://freedns.afraid.org/dynamic/update.php?"+tcAPIKey->GetValue().ToStdString());
114         stFreeDNSTestResult->SetLabel(result);
115         stFreeDNSTestResult->Show();
116         this->Layout();
117         std::cout << CurrentUserSession::currentUser->username << std::endl;
118         EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
      CurrentUserSession::currentUser->username+".xml", "User", "FreeDNSAfraidOrgDomain", tcFQDN->GetValue().ToStdString(), {"
      APIKey", "UpdatePeriod"}, {tcAPIKey->GetValue().ToStdString(),
      ConvertIntegerToString(scUpdatePeriod->GetValue())});
119     }
```

Here is the call graph for this function:



12.17.3.3 OnRbDefaultWebBrowserSelected()

```
void DimensionlessFrames::SettingsFrame::OnRbDefaultWebBrowserSelected (
              wxCommandEvent & event )  [protected], [virtual]
```

This function saves the appropriate default web browser setting to XML when the radio button selected is changed.

Parameters

in	*event*	A wxCommandEvent object that is posted after a default web browser radio button selection is changed.

Reimplemented from SettingsFrameBase.

Definition at line 125 of file SettingsFrame.cpp.

References DimensionlessFunctionality::XML::EditXML(), and SettingsFrameBase::rbDefaultWebBrowser.

```
126     {
127         switch(rbDefaultWebBrowser->GetSelection()){
128             case 0:
129             {
130                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
      CurrentUserSession::currentUser->username+".xml", "User", "DefaultWebBrowser", "GNU-IceCat");
131                 break;
132             }
133             case 1:
134             {
135                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
      CurrentUserSession::currentUser->username+".xml", "User", "DefaultWebBrowser", "System-Default");
136                 break;
137             }
```

```
138             default:
139             {
140                 break;
141             }
142         }
143     }
```

Here is the call graph for this function:

12.17.3.4 OnRbPublicSearchVisibilitySelected()

```
void DimensionlessFrames::SettingsFrame::OnRbPublicSearchVisibilitySelected (
            wxCommandEvent & event )  [protected], [virtual]
```

This function saves the appropriate public search visibility setting to XML when the radio button selected is changed.

Parameters

in	*event*	A wxCommandEvent object that is posted after a public search visibility radio button selection is changed.

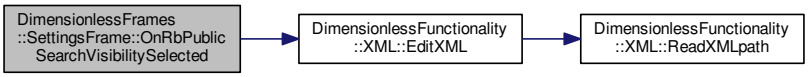
Reimplemented from SettingsFrameBase.

Definition at line 149 of file SettingsFrame.cpp.

References DimensionlessFunctionality::XML::EditXML(), and SettingsFrameBase::rbPublicSearchVisibility.

```
150     {
151         switch(rbPublicSearchVisibility->GetSelection()){
152             case 0:
153             {
154                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
    CurrentUserSession::currentUser->username+".xml", "User", "PublicSearchVisibility", "Visible");
155                 break;
156             }
157             case 1:
158             {
159                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
    CurrentUserSession::currentUser->username+".xml", "User", "PublicSearchVisibility", "Invisible");
160                 break;
161             }
162             default:
163             {
164                 break;
165             }
166         }
167     }
```

Here is the call graph for this function:



12.17.3.5 OnRbTorSelected()

```
void DimensionlessFrames::SettingsFrame::OnRbTorSelected (
            wxCommandEvent & event )  [protected], [virtual]
```

This function saves the appropriate Tor setting to XML when the radio button selected is changed.

Parameters

in	*event*	A wxCommandEvent object that is posted after a Tor radio button selection is changed.

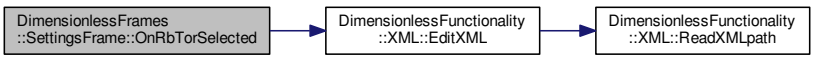
Reimplemented from SettingsFrameBase.

Definition at line 173 of file SettingsFrame.cpp.

References DimensionlessFunctionality::XML::EditXML(), and SettingsFrameBase::rbTor.

```
174     {
175         switch(rbTor->GetSelection()){
176             case 0:
177             {
178                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
    CurrentUserSession::currentUser->username+".xml", "User", "Tor", "Enabled-Proxy-Hidden-Service");
179                 break;
180             }
181             case 1:
182             {
183                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
    CurrentUserSession::currentUser->username+".xml", "User", "Tor", "Enabled-Hidden-Service");
184                 break;
185             }
186             case 2:
187             {
188                 EditXML("./Users/"+CurrentUserSession::currentUser->username+"/data/"+
    CurrentUserSession::currentUser->username+".xml", "User", "Tor", "Disabled");
189                 break;
190             }
191             default:
192             {
193                 break;
194             }
195         }
196     }
```

Here is the call graph for this function:



12.17.4 Member Data Documentation

12.17.4.1 settingsFrameThread

`std::thread DimensionlessFrames::SettingsFrame::settingsFrameThread [static], [private]`

This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Definition at line 70 of file SettingsFrame.h.

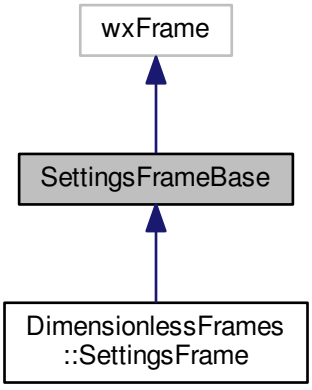
The documentation for this class was generated from the following files:

- DimensionlessClient/SettingsFrame.h
- DimensionlessClient/SettingsFrame.cpp

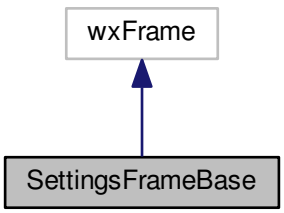
12.18 SettingsFrameBase Class Reference

`#include <Dimensionless.h>`

Inheritance diagram for SettingsFrameBase:



Collaboration diagram for SettingsFrameBase:



Public Member Functions

- wxRadioBox * GetRbDefaultWebBrowser ()
- wxRadioBox * GetRbPublicSearchVisibility ()
- wxRadioBox * GetRbTor ()
- wxStaticText * GetStFQDN ()
- wxTextCtrl * GetTcFQDN ()
- wxStaticText * GetStAPIKey ()
- wxTextCtrl * GetTcAPIKey ()
- wxStaticText * GetStUpdatePeriod ()
- wxSpinCtrl * GetScUpdatePeriod ()
- wxStaticText * GetStFreeDNSTestResult ()
- wxButton * GetBtnTestAndSaveDomain ()
- wxButton * GetBtnCheckForClientUpdates ()
- SettingsFrameBase (wxWindow *parent, wxWindowID id=wxID_ANY, const wxString &title=_("Settings"), const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(650, 550), long style=wxFRAME_F↩ LOAT_ON_PARENT|wxCAPTION|wxCLOSE_BOX)
- virtual ~SettingsFrameBase ()

Protected Member Functions

- virtual void OnRbDefaultWebBrowserSelected (wxCommandEvent &event)
- virtual void OnRbPublicSearchVisibilitySelected (wxCommandEvent &event)
- virtual void OnRbTorSelected (wxCommandEvent &event)
- virtual void OnBtnTestAndSaveDomainClick (wxCommandEvent &event)
- virtual void OnBtnCheckForClientUpdatesClick (wxCommandEvent &event)

Protected Attributes

- wxRadioBox * rbDefaultWebBrowser
- wxRadioBox * rbPublicSearchVisibility
- wxRadioBox * rbTor
- wxStaticText * stFQDN
- wxTextCtrl * tcFQDN
- wxStaticText * stAPIKey
- wxTextCtrl * tcAPIKey
- wxStaticText * stUpdatePeriod
- wxSpinCtrl * scUpdatePeriod
- wxStaticText * stFreeDNSTestResult
- wxButton * btnTestAndSaveDomain
- wxButton * btnCheckForClientUpdates

12.18.1 Detailed Description

Definition at line 215 of file Dimensionless.h.

12.18.2 Constructor & Destructor Documentation

12.18.2.1 SettingsFrameBase()

```
SettingsFrameBase::SettingsFrameBase (
            wxWindow * parent,
            wxWindowID id = wxID_ANY,
            const wxString & title = _("Settings"),
            const wxPoint & pos = wxDefaultPosition,
            const wxSize & size = wxSize(650,550),
            long style = wxFRAME_FLOAT_ON_PARENT|wxCAPTION|wxCLOSE_BOX )
```

Definition at line 515 of file Dimensionless.cpp.

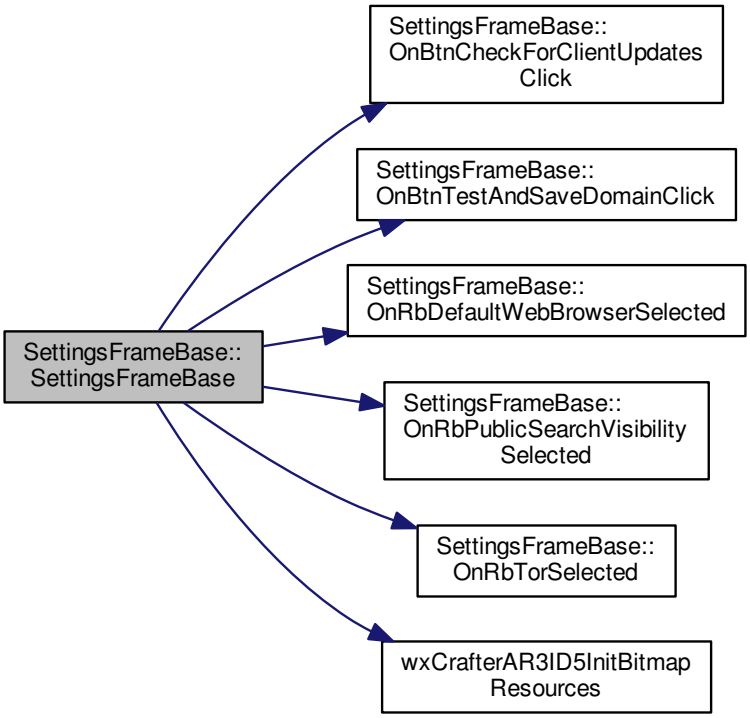
References bBitmapLoaded, btnCheckForClientUpdates, btnTestAndSaveDomain, OnBtnCheckForClient↩UpdatesClick(), OnBtnTestAndSaveDomainClick(), OnRbDefaultWebBrowserSelected(), OnRbPublicSearch↩VisibilitySelected(), OnRbTorSelected(), rbDefaultWebBrowser, rbPublicSearchVisibility, rbTor, scUpdatePeriod, stAPIKey, stFQDN, stFreeDNSTestResult, stUpdatePeriod, tcAPIKey, tcFQDN, WXC_FROM_DIP, and wxCrafter↩AR3ID5InitBitmapResources().

```
516     : wxFrame(parent, id, title, pos, size, style)
517 {
518     if ( !bBitmapLoaded ) {
519         // We need to initialise the default bitmap handler
520         wxXmlResource::Get()->AddHandler(new wxBitmapXmlHandler);
521         wxCrafterAR3ID5InitBitmapResources();
522         bBitmapLoaded = true;
523     }
524
525     wxBoxSizer* bszrSettings = new wxBoxSizer(wxVERTICAL);
526     this->SetSizer(bszrSettings);
527
528     wxBoxSizer* bszrDefaultWebBrowser = new wxBoxSizer(wxHORIZONTAL);
529
530     bszrSettings->Add(bszrDefaultWebBrowser, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
531
532     wxArrayString rbDefaultWebBrowserArr;
533     rbDefaultWebBrowserArr.Add(_("GNU IceCat"));
534     rbDefaultWebBrowserArr.Add(_("System Default"));
535     rbDefaultWebBrowser = new wxRadioBox(this, wxID_ANY, _("Default Web Browser"),
    wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), rbDefaultWebBrowserArr, 1, 0);
536     rbDefaultWebBrowser->SetSelection(0);
537
538     bszrDefaultWebBrowser->Add(rbDefaultWebBrowser, 1, wxALL,
    WXC_FROM_DIP(5));
539
540     wxBoxSizer* bszrPublicSearchVisibility = new wxBoxSizer(wxHORIZONTAL);
541
542     bszrSettings->Add(bszrPublicSearchVisibility, 0, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
543
544     wxArrayString rbPublicSearchVisibilityArr;
545     rbPublicSearchVisibilityArr.Add(_("Visible"));
546     rbPublicSearchVisibilityArr.Add(_("Invisible"));
547     rbPublicSearchVisibility = new wxRadioBox(this, wxID_ANY, _("Public Search
     Visibility"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), rbPublicSearchVisibilityArr, 1, 0);
548     rbPublicSearchVisibility->SetSelection(0);
549
550     bszrPublicSearchVisibility->Add(rbPublicSearchVisibility, 1, wxALL,
    WXC_FROM_DIP(5));
551
552     wxBoxSizer* bszrTor = new wxBoxSizer(wxHORIZONTAL);
553
554     bszrSettings->Add(bszrTor, 0, wxALL|wxEXPAND, WXC_FROM_DIP(5));
555
556     wxArrayString rbTorArr;
557     rbTorArr.Add(_("Enabled (Proxy and Hidden Service)"));
558     rbTorArr.Add(_("Enabled (Hidden Service Only)"));
559     rbTorArr.Add(_("Disabled"));
560     rbTor = new wxRadioBox(this, wxID_ANY, _("Tor"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)
    ), rbTorArr, 1, 0);
561     rbTor->SetSelection(1);
562
563     bszrTor->Add(rbTor, 1, wxALL, WXC_FROM_DIP(5));
564
565     wxStaticBoxSizer* sbszrFreeDNS = new wxStaticBoxSizer( new wxStaticBox(this, wxID_ANY, _("
    Freedns.afraid.org Domain")), wxVERTICAL);
566
567     bszrSettings->Add(sbszrFreeDNS, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
```

```
568
569     wxBoxSizer* bszrFreeDNS = new wxBoxSizer(wxVERTICAL);
570
571     sbszrFreeDNS->Add(bszrFreeDNS, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
572
573     wxBoxSizer* bszrFreeDNSFQDN = new wxBoxSizer(wxHORIZONTAL);
574
575     bszrFreeDNS->Add(bszrFreeDNSFQDN, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
576
577     stFQDN = new wxStaticText(this, wxID_ANY, _("Fully Qualified Domain Name:"), wxDefaultPosition,
    wxDLG_UNIT(this, wxSize(-1,-1)), 0);
578
579     bszrFreeDNSFQDN->Add(stFQDN, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|wxALIGN_CENTER_VERTICAL,
    WXC_FROM_DIP(5));
580     stFQDN->SetMinSize(wxSize(190,-1));
581
582     tcFQDN = new wxTextCtrl(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1
    )), 0);
583     #if wxVERSION_NUMBER >= 3000
584     tcFQDN->SetHint(_("(for examples see: http://freedns.afraid.org/domain/registry/)"));
585     #endif
586
587     bszrFreeDNSFQDN->Add(tcFQDN, 1, wxALL|wxALIGN_CENTER_HORIZONTAL|wxALIGN_CENTER_VERTICAL,
    WXC_FROM_DIP(5));
588
589     wxBoxSizer* bszrFreeDNSAPIKey = new wxBoxSizer(wxHORIZONTAL);
590
591     bszrFreeDNS->Add(bszrFreeDNSAPIKey, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
592
593     stAPIKey = new wxStaticText(this, wxID_ANY, _("Update API Key for Domain:"), wxDefaultPosition,
     wxDLG_UNIT(this, wxSize(-1,-1)), 0);
594
595     bszrFreeDNSAPIKey->Add(stAPIKey, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|wxALIGN_CENTER_VERTICAL,
    WXC_FROM_DIP(5));
596     stAPIKey->SetMinSize(wxSize(190,-1));
597
598     tcAPIKey = new wxTextCtrl(this, wxID_ANY, wxT(""), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-
    1,-1)), 0);
599     #if wxVERSION_NUMBER >= 3000
600     tcAPIKey->SetHint(_("(see quick cron here: http://freedns.afraid.org/dynamic/)"));
601     #endif
602
603     bszrFreeDNSAPIKey->Add(tcAPIKey, 1, wxALL|wxALIGN_CENTER_HORIZONTAL|wxALIGN_CENTER_VERTICAL,
    WXC_FROM_DIP(5));
604
605     wxBoxSizer* bszrFreeDNSUpdatePeriod = new wxBoxSizer(wxHORIZONTAL);
606
607     bszrFreeDNS->Add(bszrFreeDNSUpdatePeriod, 1, wxALL|wxEXPAND, WXC_FROM_DIP(5));
608
609     stUpdatePeriod = new wxStaticText(this, wxID_ANY, _("Update Period (in minutes)"),
    wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
610
611     bszrFreeDNSUpdatePeriod->Add(stUpdatePeriod, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|
    wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
612     stUpdatePeriod->SetMinSize(wxSize(190,-1));
613
614     scUpdatePeriod = new wxSpinCtrl(this, wxID_ANY, wxT("120"), wxDefaultPosition, wxDLG_UNIT
    (this, wxSize(-1,-1)), wxSP_ARROW_KEYS);
615     scUpdatePeriod->SetRange(1, 1440);
616     scUpdatePeriod->SetValue(120);
617
618     bszrFreeDNSUpdatePeriod->Add(scUpdatePeriod, 1, wxALL|wxALIGN_CENTER_HORIZONTAL|
    wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
619
620     stFreeDNSTestResult = new wxStaticText(this, wxID_ANY, _("Success! Domain IP has
     been updated and saved."), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
621     stFreeDNSTestResult->Hide();
622
623     bszrFreeDNS->Add(stFreeDNSTestResult, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|
    wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
624
625     btnTestAndSaveDomain = new wxButton(this, wxID_ANY, _("Test and Save Domain"),
    wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
626
627     bszrFreeDNS->Add(btnTestAndSaveDomain, 0, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
628
629     btnCheckForClientUpdates = new wxButton(this, wxID_ANY, _("Check for Client
     Updates"), wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,-1)), 0);
630
631     bszrSettings->Add(btnCheckForClientUpdates, 0, wxALL|wxEXPAND,
    WXC_FROM_DIP(5));
632
633     SetName(wxT("SettingsFrameBase"));
634     SetMinClientSize(wxSize(650,550));
635     SetSize(650,550);
636     if (GetSizer()) {
```

```
637          GetSizer()->Fit(this);
638     }
639     if(GetParent()) {
640         CentreOnParent(wxBOTH);
641     } else {
642         CentreOnScreen(wxBOTH);
643     }
644 #if wxVERSION_NUMBER >= 2900
645     if(!wxPersistenceManager::Get().Find(this)) {
646         wxPersistenceManager::Get().RegisterAndRestore(this);
647     } else {
648         wxPersistenceManager::Get().Restore(this);
649     }
650 #endif
651     // Connect events
652     rbDefaultWebBrowser->Connect(wxEVT_COMMAND_RADIOBOX_SELECTED, wxCommandEventHandler(
      SettingsFrameBase::OnRbDefaultWebBrowserSelected), NULL,
      this);
653     rbPublicSearchVisibility->Connect(wxEVT_COMMAND_RADIOBOX_SELECTED,
      wxCommandEventHandler(SettingsFrameBase::OnRbPublicSearchVisibilitySelected
      ), NULL, this);
654     rbTor->Connect(wxEVT_COMMAND_RADIOBOX_SELECTED, wxCommandEventHandler(
      SettingsFrameBase::OnRbTorSelected), NULL, this);
655     btnTestAndSaveDomain->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      SettingsFrameBase::OnBtnTestAndSaveDomainClick), NULL, this);
656     btnCheckForClientUpdates->Connect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(SettingsFrameBase::OnBtnCheckForClientUpdatesClick)
      , NULL, this);
657
658 }
```

Here is the call graph for this function:



12.18.2.2 ~SettingsFrameBase()

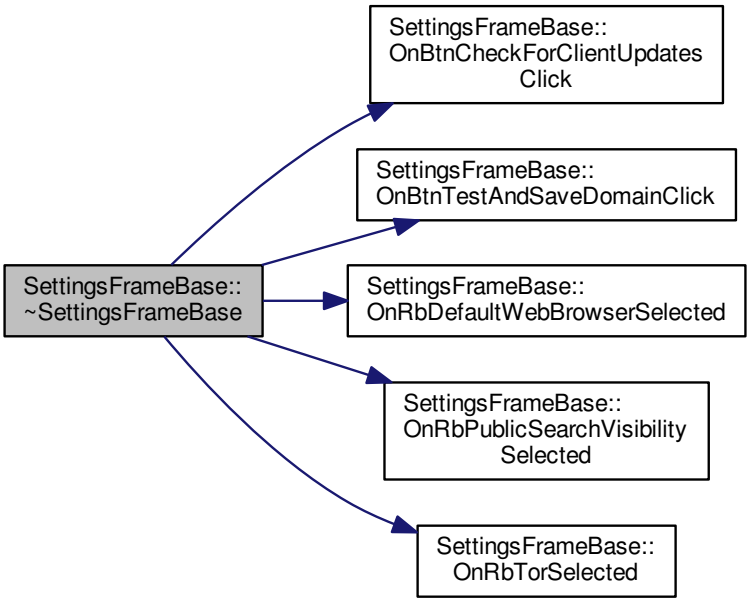
```
SettingsFrameBase::~SettingsFrameBase ( )  [virtual]
```

Definition at line 660 of file Dimensionless.cpp.

References btnCheckForClientUpdates, btnTestAndSaveDomain, OnBtnCheckForClientUpdatesClick(), OnBtnTestAndSaveDomainClick(), OnRbDefaultWebBrowserSelected(), OnRbPublicSearchVisibilitySelected(), OnRbTorSelected(), rbDefaultWebBrowser, rbPublicSearchVisibility, and rbTor.

```
661 {
662     rbDefaultWebBrowser->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
      wxCommandEventHandler(SettingsFrameBase::OnRbDefaultWebBrowserSelected), NULL,
      this);
663     rbPublicSearchVisibility->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED,
      wxCommandEventHandler(SettingsFrameBase::OnRbPublicSearchVisibilitySelected
      ), NULL, this);
664     rbTor->Disconnect(wxEVT_COMMAND_RADIOBOX_SELECTED, wxCommandEventHandler(
      SettingsFrameBase::OnRbTorSelected), NULL, this);
665     btnTestAndSaveDomain->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(SettingsFrameBase::OnBtnTestAndSaveDomainClick), NULL, this)
      ;
666     btnCheckForClientUpdates->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED,
      wxCommandEventHandler(SettingsFrameBase::OnBtnCheckForClientUpdatesClick
      ), NULL, this);
667
668 }
```

Here is the call graph for this function:



12.18.3 Member Function Documentation

12.18.3.1 GetBtnCheckForClientUpdates()

```
wxButton* SettingsFrameBase::GetBtnCheckForClientUpdates ( )  [inline]
```

Definition at line 250 of file Dimensionless.h.

```
250 { return btnCheckForClientUpdates; }
```

12.18.3.2 GetBtnTestAndSaveDomain()

```
wxButton* SettingsFrameBase::GetBtnTestAndSaveDomain ( )  [inline]
```

Definition at line 249 of file Dimensionless.h.

```
249 { return btnTestAndSaveDomain; }
```

12.18.3.3 GetRbDefaultWebBrowser()

```
wxRadioBox* SettingsFrameBase::GetRbDefaultWebBrowser ( )  [inline]
```

Definition at line 239 of file Dimensionless.h.

```
239 { return rbDefaultWebBrowser; }
```

12.18.3.4 GetRbPublicSearchVisibility()

```
wxRadioBox* SettingsFrameBase::GetRbPublicSearchVisibility ( )  [inline]
```

Definition at line 240 of file Dimensionless.h.

```
240 { return rbPublicSearchVisibility; }
```

12.18.3.5 GetRbTor()

```
wxRadioBox* SettingsFrameBase::GetRbTor ( )  [inline]
```

Definition at line 241 of file Dimensionless.h.

```
241 { return rbTor; }
```

12.18.3.6 GetScUpdatePeriod()

```
wxSpinCtrl* SettingsFrameBase::GetScUpdatePeriod ( )  [inline]
```

Definition at line 247 of file Dimensionless.h.

```
247 { return scUpdatePeriod; }
```

12.18.3.7 GetStAPIKey()

```
wxStaticText* SettingsFrameBase::GetStAPIKey ( )  [inline]
```

Definition at line 244 of file Dimensionless.h.

```
244 { return stAPIKey; }
```

12.18.3.8 GetStFQDN()

```
wxStaticText* SettingsFrameBase::GetStFQDN ( )  [inline]
```

Definition at line 242 of file Dimensionless.h.

```
242 { return stFQDN; }
```

12.18.3.9 GetStFreeDNSTestResult()

```
wxStaticText* SettingsFrameBase::GetStFreeDNSTestResult ( )  [inline]
```

Definition at line 248 of file Dimensionless.h.

```
248 { return stFreeDNSTestResult; }
```

12.18.3.10 GetStUpdatePeriod()

```
wxStaticText* SettingsFrameBase::GetStUpdatePeriod ( )  [inline]
```

Definition at line 246 of file Dimensionless.h.

```
246 { return stUpdatePeriod; }
```

12.18.3.11 GetTcAPIKey()

```
wxTextCtrl* SettingsFrameBase::GetTcAPIKey ( )  [inline]
```

Definition at line 245 of file Dimensionless.h.

```
245 { return tcAPIKey; }
```

12.18.3.12 GetTcFQDN()

```
wxTextCtrl* SettingsFrameBase::GetTcFQDN ( )  [inline]
```

Definition at line 243 of file Dimensionless.h.

```
243 { return tcFQDN; }
```

12.18.3.13 OnBtnCheckForClientUpdatesClick()

```
virtual void SettingsFrameBase::OnBtnCheckForClientUpdatesClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

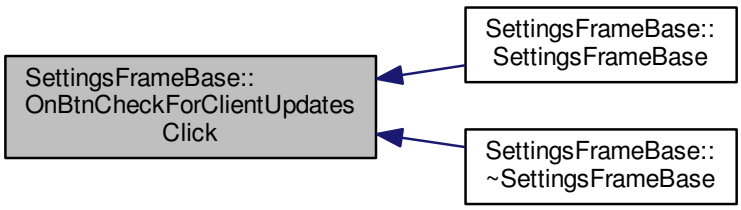
Reimplemented in DimensionlessFrames::SettingsFrame.

Definition at line 236 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

```
236 { event.Skip(); }
```

Here is the caller graph for this function:



12.18.3.14 OnBtnTestAndSaveDomainClick()

```
virtual void SettingsFrameBase::OnBtnTestAndSaveDomainClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

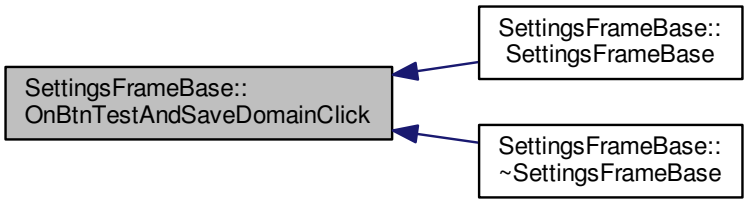
Reimplemented in DimensionlessFrames::SettingsFrame.

Definition at line 235 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

```
235 { event.Skip(); }
```

Here is the caller graph for this function:



12.18.3.15 OnRbDefaultWebBrowserSelected()

```
virtual void SettingsFrameBase::OnRbDefaultWebBrowserSelected (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

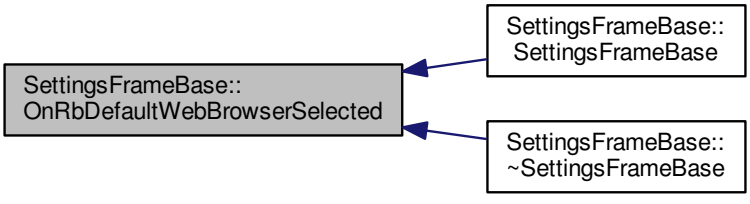
Reimplemented in DimensionlessFrames::SettingsFrame.

Definition at line 232 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

```
232 { event.Skip(); }
```

Here is the caller graph for this function:



12.18.3.16 OnRbPublicSearchVisibilitySelected()

```
virtual void SettingsFrameBase::OnRbPublicSearchVisibilitySelected (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

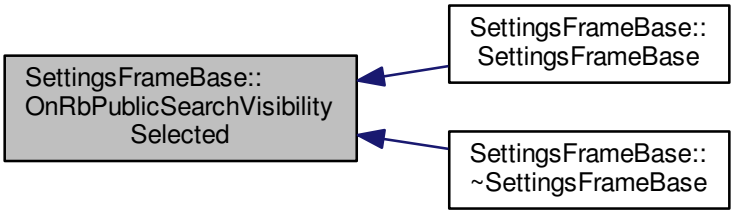
Reimplemented in DimensionlessFrames::SettingsFrame.

Definition at line 233 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

```
233 { event.Skip(); }
```

Here is the caller graph for this function:



12.18.3.17 OnRbTorSelected()

```
virtual void SettingsFrameBase::OnRbTorSelected (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

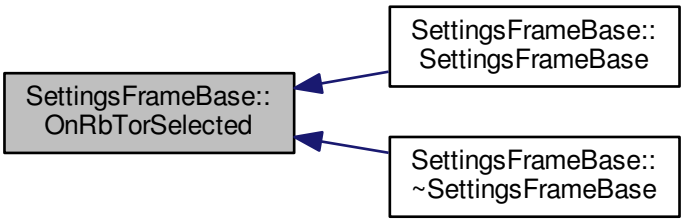
Reimplemented in DimensionlessFrames::SettingsFrame.

Definition at line 234 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

```
234 { event.Skip(); }
```

Here is the caller graph for this function:



12.18.4 Member Data Documentation

12.18.4.1 btnCheckForClientUpdates

```
wxButton* SettingsFrameBase::btnCheckForClientUpdates  [protected]
```

Definition at line 229 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

12.18.4.2 btnTestAndSaveDomain

```
wxButton* SettingsFrameBase::btnTestAndSaveDomain  [protected]
```

Definition at line 228 of file Dimensionless.h.

Referenced by SettingsFrameBase(), and ~SettingsFrameBase().

12.18.4.3 rbDefaultWebBrowser

```
wxRadioBox* SettingsFrameBase::rbDefaultWebBrowser  [protected]
```

Definition at line 218 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnRbDefaultWebBrowserSelected(), SettingsFrameBase(), and ~SettingsFrameBase().

12.18.4.4 rbPublicSearchVisibility

```
wxRadioBox* SettingsFrameBase::rbPublicSearchVisibility  [protected]
```

Definition at line 219 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnRbPublicSearchVisibilitySelected(), SettingsFrame↩Base(), and ~SettingsFrameBase().

12.18.4.5 rbTor

```
wxRadioBox* SettingsFrameBase::rbTor  [protected]
```

Definition at line 220 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnRbTorSelected(), SettingsFrameBase(), and ~Settings↩FrameBase().

12.18.4.6 scUpdatePeriod

```
wxSpinCtrl* SettingsFrameBase::scUpdatePeriod  [protected]
```

Definition at line 226 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick(), DimensionlessFrames↩::SettingsFrame::SettingsFrame(), and SettingsFrameBase().

12.18.4.7 stAPIKey

```
wxStaticText* SettingsFrameBase::stAPIKey  [protected]
```

Definition at line 223 of file Dimensionless.h.

Referenced by SettingsFrameBase().

12.18.4.8 stFQDN

```
wxStaticText* SettingsFrameBase::stFQDN  [protected]
```

Definition at line 221 of file Dimensionless.h.

Referenced by SettingsFrameBase().

12.18.4.9 stFreeDNSTestResult

```
wxStaticText* SettingsFrameBase::stFreeDNSTestResult  [protected]
```

Definition at line 227 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick(), and SettingsFrame↩Base().

12.18.4.10 stUpdatePeriod

```
wxStaticText* SettingsFrameBase::stUpdatePeriod  [protected]
```

Definition at line 225 of file Dimensionless.h.

Referenced by SettingsFrameBase().

12.18.4.11 tcAPIKey

```
wxTextCtrl* SettingsFrameBase::tcAPIKey  [protected]
```

Definition at line 224 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick(), DimensionlessFrames↩
::SettingsFrame::SettingsFrame(), and SettingsFrameBase().

12.18.4.12 tcFQDN

```
wxTextCtrl* SettingsFrameBase::tcFQDN  [protected]
```

Definition at line 222 of file Dimensionless.h.

Referenced by DimensionlessFrames::SettingsFrame::OnBtnTestAndSaveDomainClick(), DimensionlessFrames↩
::SettingsFrame::SettingsFrame(), and SettingsFrameBase().

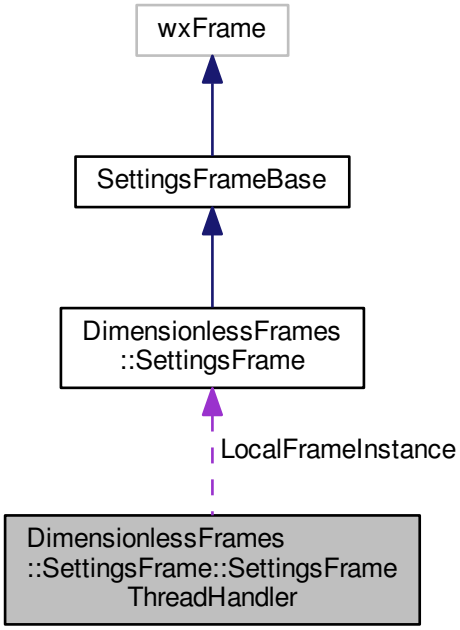
The documentation for this class was generated from the following files:

- DimensionlessClient/Dimensionless.h
- DimensionlessClient/Dimensionless.cpp

12.19 DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler Struct Reference

This struct handles events posted from our settingsFrameThread.

Collaboration diagram for DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler:



Public Member Functions

- SettingsFrameThreadHandler (SettingsFrame *frame)
- void operator() (wxThreadEvent &evt)

Private Attributes

- SettingsFrame * LocalFrameInstance = nullptr

  A raw pointer reference to a SettingsFrame to handle events for.

12.19.1 Detailed Description

This struct handles events posted from our settingsFrameThread.

Definition at line 40 of file SettingsFrame.cpp.

12.19.2 Constructor & Destructor Documentation

12.19.2.1 SettingsFrameThreadHandler()

```
DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler::SettingsFrameThreadHandler (
            SettingsFrame * frame )  [inline]
```

This constructor initializes a new SettingsFrameThreadHandler object with a pointer to the current SettingsFrame instance so it may be modified as required.

Parameters

in	*frame*	The raw pointer to a SettingsFrame instance.

Definition at line 47 of file SettingsFrame.cpp.

References LocalFrameInstance.

```
48             {
49                 // Set the local SettingsFrame reference to that passed.
50                 LocalFrameInstance = frame;
51             }
```

12.19.3 Member Function Documentation

12.19.3.1 operator()()

```
void DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler::operator() (
            wxThreadEvent & evt )  [inline]
```

This function remaps the () operator of SettingsFrameThreadHandler to handle settingsFrameThread events.

Parameters

in	*event*	A wxThreadEvent object posted by our settingsFrameThread.

Definition at line 57 of file SettingsFrame.cpp.

```
58             {
59                 switch(evt.GetId()){
60                     case wxID_ANY:
61                     {
62                         break;
63                     }
64                     default:
65                     {
66                         break;
67                     }
68                 }
69             }
```

12.19.4 Member Data Documentation

12.19.4.1 LocalFrameInstance

```
SettingsFrame* DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler::LocalFrame↩
Instance = nullptr  [private]
```

A raw pointer reference to a SettingsFrame to handle events for.

Definition at line 72 of file SettingsFrame.cpp.

Referenced by SettingsFrameThreadHandler().

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

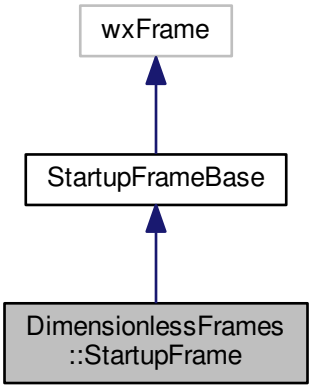
- DimensionlessClient/SettingsFrame.cpp

12.20 DimensionlessFrames::StartupFrame Class Reference

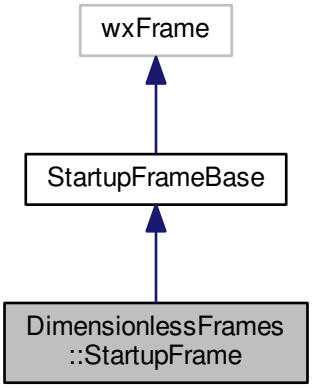
This class creates a StartupFrame that is the first frame displayed when the application is started.

`#include <StartupFrame.h>`

Inheritance diagram for DimensionlessFrames::StartupFrame:



Collaboration diagram for DimensionlessFrames::StartupFrame:



Classes

- struct StartupFrameThreadHandler

  This struct handles events posted from our startupFrameThread.

Public Member Functions

- StartupFrame (wxWindow *parent)
- virtual ~StartupFrame ()

Protected Member Functions

- virtual void OnTCPasswordEnterPressed (wxCommandEvent &event)
- virtual void OnTCRepeatPasswordEnterPressed (wxCommandEvent &event)
- virtual void OnTCUsernameEnterPressed (wxCommandEvent &event)
- virtual void OnBtnRegisterClicked (wxCommandEvent &event)
- virtual void OnBtnLoginClick (wxCommandEvent &event)
- virtual void OnAbout (wxCommandEvent &event)
- virtual void OnClose (wxCloseEvent &event)
- virtual void OnQuit (wxCommandEvent &event)

Private Types

- enum LoginState { LoginState::Login, LoginState::Register }

  This enumeration decides the state of the StartupFrame.

Private Member Functions

- void ThreadEntryCheckLogin (const string &username, const string &password)

Private Attributes

- LoginState currentLoginState = LoginState::Login

  This variable stores the current state of this StartupFrame instance.
- bool UsersExist = false

  This boolean decides if there exist any pre-existing users for our application.

Static Private Attributes

- static std::thread startupFrameThread

  This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Additional Inherited Members

12.20.1 Detailed Description

This class creates a StartupFrame that is the first frame displayed when the application is started.

Definition at line 44 of file StartupFrame.h.

12.20.2 Member Enumeration Documentation

12.20.2.1 LoginState

```
enum DimensionlessFrames::StartupFrame::LoginState  [strong], [private]
```

This enumeration decides the state of the StartupFrame.

Enumerator

Login	Denotes that the StartupFrame is in the "login an existing user" state.
Register	Denotes that the StartupFrame is in the "register a new user" state.

Definition at line 84 of file StartupFrame.h.

```
84                                  {
85                                    Login,
86                                    Register
87                                  };
```

12.20.3 Constructor & Destructor Documentation

12.20.3.1 StartupFrame()

```
DimensionlessFrames::StartupFrame::StartupFrame (
            wxWindow * parent )
```

This constructor initializes a new StartupFrame.

Parameters

in	*parent*	The parent window of the one to be created.

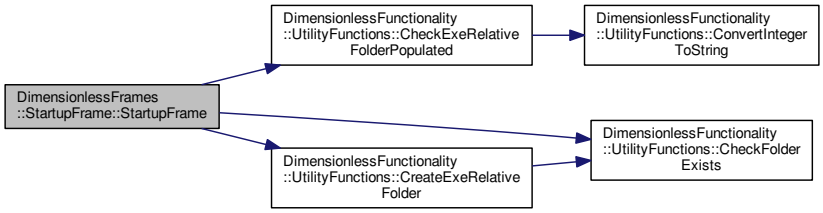
Definition at line 116 of file StartupFrame.cpp.

References StartupFrameBase::btnLogin, DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated(), DimensionlessFunctionality::UtilityFunctions::CheckFolderExists(), DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFolder(), currentLoginState, Register, StartupFrameBase::stRegistrationInfo, StartupFrameBase::stRepeatPassword, StartupFrameBase::tcRepeatPassword, and UsersExist.

```
116                                         : StartupFrameBase(parent)
117     {
118         this->Bind(wxEVT_THREAD, StartupFrame::StartupFrameThreadHandler(this), wxID_ANY);
119         if(CheckFolderExists("Users") &&
      CheckExeRelativeFolderPopulated("Users")){
120             wxLogMessage("Users directory exists and is populated.");
```

```
121             UsersExist = true;
122         } else {
123             wxLogMessage("Users directory is empty or does not exist.");
124             // Probably the first time the user starts the application.
125             CreateExeRelativeFolder("Users");
126             currentLoginState = LoginState::Register;
127             btnLogin->SetLabel("Go Back to Login");
128             btnLogin->Enable(false);
129             stRepeatPassword->Show();
130             tcRepeatPassword->Show();
131             stRegistrationInfo->Show();
132         }
133     }
```

Here is the call graph for this function:



12.20.3.2 ~StartupFrame()

```
DimensionlessFrames::StartupFrame::~StartupFrame ( )  [virtual]
```

This destructor unbinds any thread events from our StartupFrame.

Definition at line 138 of file StartupFrame.cpp.

```
139     {
140         this->Unbind(wxEVT_THREAD, StartupFrame::StartupFrameThreadHandler(this), wxID_ANY);
141     }
```

12.20.4 Member Function Documentation

12.20.4.1 OnAbout()

```
void DimensionlessFrames::StartupFrame::OnAbout (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "About" item is clicked on a StartupFrame's header menu.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "About" option is selected under the "Help" header menu.

Reimplemented from StartupFrameBase.

Definition at line 147 of file StartupFrame.cpp.

```
148     {
149         wxAboutDialogInfo* info = new wxAboutDialogInfo();
150         info->AddDeveloper(_("Eclectic Diemensions - http://eclecticdimensions.com"));
151         info->SetCopyright(wxString::FromUTF8("© Eclectic Dimensions"));
152         info->SetDescription(_("A fully decentralized social platform."));
153         info->SetName(_("Dimensionless"));
154         info->SetWebSite(_("http://eclecticdimensions.com"),_("Eclectic Dimensions Website"));
155         wxAboutBox(*info);
156     }
```

12.20.4.2 OnBtnLoginClick()

```
void DimensionlessFrames::StartupFrame::OnBtnLoginClick (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Log In" button is clicked on a StartupFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Log In" button is clicked.

Reimplemented from StartupFrameBase.

Definition at line 180 of file StartupFrame.cpp.

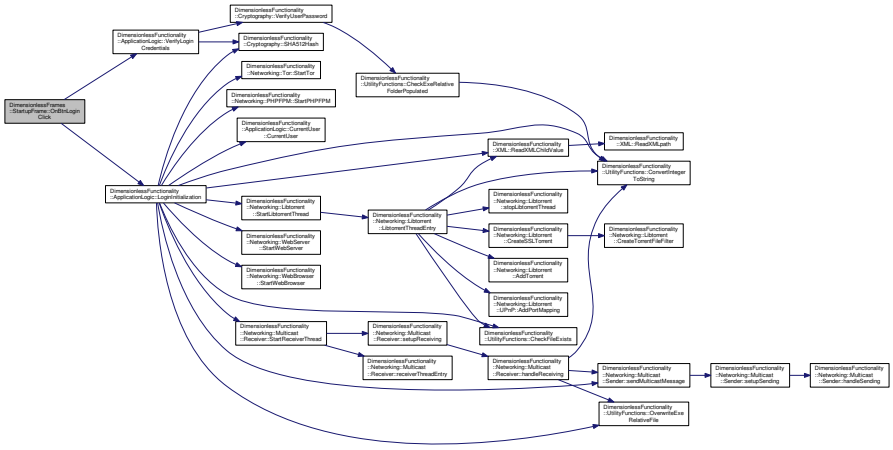
References StartupFrameBase::btnLogin, currentLoginState, Login, DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), StartupFrameBase::stRegistrationInfo, StartupFrameBase::stRepeatPassword, StartupFrameBase::tcPassword, StartupFrameBase::tcRepeatPassword, StartupFrameBase::tcUsername, and DimensionlessFunctionality::ApplicationLogic::VerifyLoginCredentials().

Referenced by OnTCPasswordEnterPressed(), OnTCRepeatPasswordEnterPressed(), and OnTCUsernameEnterPressed().

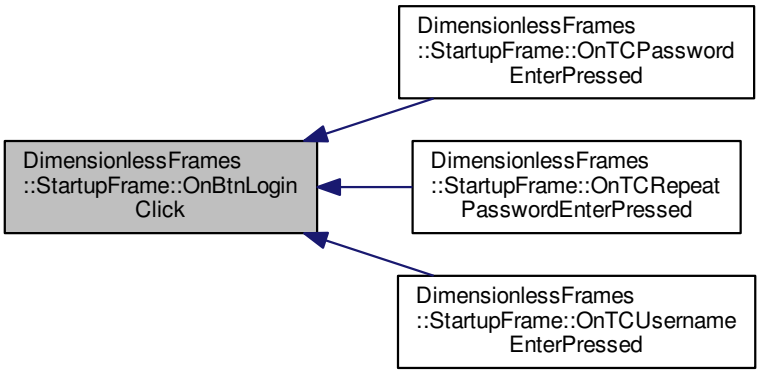
```
181     {
182         if(currentLoginState == LoginState::Login){
183             if(tcPassword->GetValue().ToStdString().length() > 0 &&
    VerifyLoginCredentials(tcUsername->GetValue().ToStdString(),
    tcPassword->GetValue().ToStdString())){
184                 MainFrame *mainFrame = new MainFrame(this, tcUsername->GetValue().ToStdString());
185                 LoginInitialization(tcUsername->GetValue().ToStdString(),
    tcPassword->GetValue().ToStdString());
186                 tcUsername->Clear();
187                 tcPassword->Clear();
188                 tcRepeatPassword->Clear();
189                 stRegistrationInfo->Hide();
190                 mainFrame->Show();
191                 this->Hide();
192             } else {
```

```
193                 stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
194                 stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
195                 stRegistrationInfo->SetLabel("Bad username or password.");
196                 stRegistrationInfo->Wrap(400);
197                 stRegistrationInfo->Show();
198                 this->Layout();
199             }
200         } else {
201             currentLoginState = LoginState::Login;
202             stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
203             stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
204             stRegistrationInfo->SetLabel("No users detected, you must register before
    proceeding.");
205             stRegistrationInfo->Hide();
206             btnLogin->SetLabel("Login");
207             stRepeatPassword->Hide();
208             tcRepeatPassword->Hide();
209             this->Layout();
210         }
211     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.20.4.3 OnBtnRegisterClicked()

```
void DimensionlessFrames::StartupFrame::OnBtnRegisterClicked (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Register" button is clicked on a StartupFrame instance.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Register" button is clicked.

Reimplemented from StartupFrameBase.

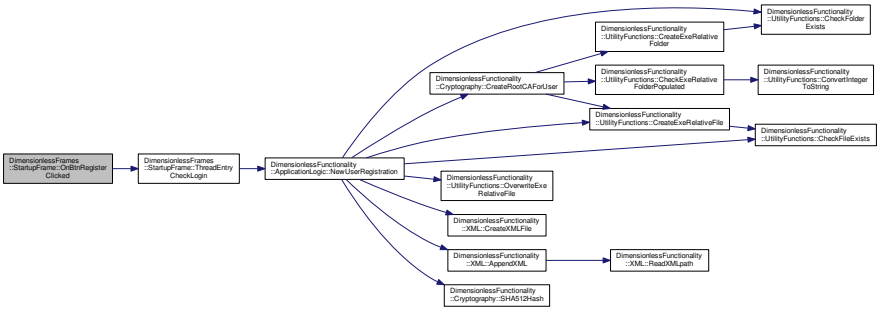
Definition at line 217 of file StartupFrame.cpp.

References StartupFrameBase::btnLogin, StartupFrameBase::btnRegister, currentLoginState, Login, Register, startupFrameThread, StartupFrameBase::stRegistrationInfo, StartupFrameBase::stRepeatPassword, StartupFrameBase::tcPassword, StartupFrameBase::tcRepeatPassword, StartupFrameBase::tcUsername, and ThreadEntryCheckLogin().

```
218     {
219         if(currentLoginState == LoginState::Login){
220             currentLoginState = LoginState::Register;
221             btnLogin->SetLabel("Go Back to Login");
222             stRepeatPassword->Show();
223             tcRepeatPassword->Show();
224             this->Layout();
225         } else {
226             stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(0,204,0)")));
227             stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(0,204,0)")));
228             stRegistrationInfo->SetLabel("Registering a new user, this can take a while
    (many minutes or a few seconds, secure parameter generation takes time - please be patient).");
229             stRegistrationInfo->Wrap(400);
230             stRegistrationInfo->Show();
231             this->Layout();
232
233             if(tcPassword->GetValue().Length() < 4 || tcUsername->GetValue().Length() <
    4){
234                 stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
235                 stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
236                 stRegistrationInfo->SetLabel("Error! Username or password too short, at
    minimum 4 characters are needed.");
237                 stRegistrationInfo->Wrap(400);
238                 stRegistrationInfo->Show();
239                 this->Layout();
240                 return;
241             } else if(tcUsername->GetValue().Length() > 512) {
242                 stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
243                 stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
244                 stRegistrationInfo->SetLabel("Error! Username too long, the maximum
    username length is 512 characters.");
245                 stRegistrationInfo->Wrap(400);
246                 stRegistrationInfo->Show();
247                 this->Layout();
248                 return;
249             }
250
251             std::string username = tcUsername->GetValue().ToStdString();
252             std::string password = "";
253
254             if(tcPassword->GetValue() == tcRepeatPassword->GetValue()){
255                 btnLogin->Enable(false);
256                 btnRegister->Enable(false);
257                 password = tcPassword->GetValue().ToStdString();
258                 std::thread::id nothread;
259                 if(startupFrameThread.get_id() != nothread){
260                     startupFrameThread.~thread();
261                 }
262                 startupFrameThread = std::thread(&
    StartupFrame::ThreadEntryCheckLogin, this, username, password);
263                 startupFrameThread.detach();
264             } else {
265                 stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
266                 stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
```

```
267                 stRegistrationInfo->SetLabel("The enterred passwords do not match.");
268                 stRegistrationInfo->Show();
269                 this->Layout();
270                 return;
271             }
272         }
273     }
```

Here is the call graph for this function:



12.20.4.4 OnClose()

```
void DimensionlessFrames::StartupFrame::OnClose (
            wxCloseEvent & event )  [protected], [virtual]
```

This function manages what happens after the StartupFrame is closed.

Parameters

in	*event*	A wxCloseEvent object that is posted when the StartupFrame must be closed.

Reimplemented from StartupFrameBase.

Definition at line 162 of file StartupFrame.cpp.

```
163     {
164         this->Destroy();
165     }
```

12.20.4.5 OnQuit()

```
void DimensionlessFrames::StartupFrame::OnQuit (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the "Quit" item is clicked on a StartupFrame's header menu.

Parameters

in	*event*	A wxCommandEvent object that is posted after the "Quit" option is selected under the "File" header menu.

Reimplemented from StartupFrameBase.

Definition at line 171 of file StartupFrame.cpp.

```
172     {
173         this->Close(true);
174     }
```

12.20.4.6 OnTCPasswordEnterPressed()

```
void DimensionlessFrames::StartupFrame::OnTCPasswordEnterPressed (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the enter keyboard button is pressed in the field where you enter a login/registration password.

Parameters

in	*event*	A wxCommandEvent object that is posted after the enter keyboard button is pressed.

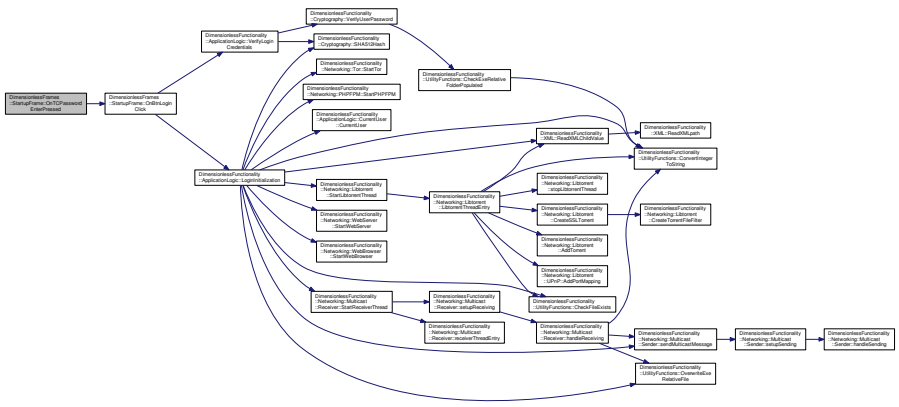
Reimplemented from StartupFrameBase.

Definition at line 288 of file StartupFrame.cpp.

References OnBtnLoginClick().

```
289     {
290         this->OnBtnLoginClick(event);
291     }
```

Here is the call graph for this function:



12.20.4.7 OnTCRepeatPasswordEnterPressed()

```
void DimensionlessFrames::StartupFrame::OnTCRepeatPasswordEnterPressed (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the enter keyboard button is pressed in the field where you enter a registration password again.

Parameters

in	*event*	A wxCommandEvent object that is posted after the enter keyboard button is pressed.

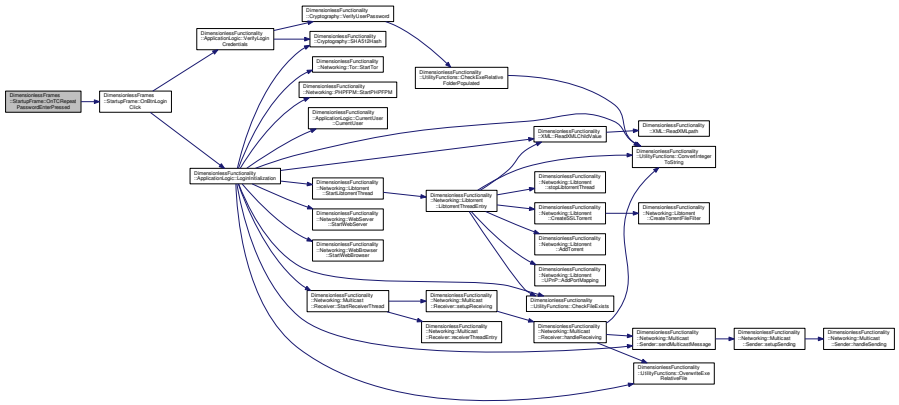
Reimplemented from StartupFrameBase.

Definition at line 297 of file StartupFrame.cpp.

References OnBtnLoginClick().

```
298     {
299         this->OnBtnLoginClick(event);
300     }
```

Here is the call graph for this function:



12.20.4.8 OnTCUsernameEnterPressed()

```
void DimensionlessFrames::StartupFrame::OnTCUsernameEnterPressed (
            wxCommandEvent & event )  [protected], [virtual]
```

This function manages what happens after the enter keyboard button is pressed in the field where you enter a login/registration username.

Parameters

in	*event*	A wxCommandEvent object that is posted after the enter keyboard button is pressed.

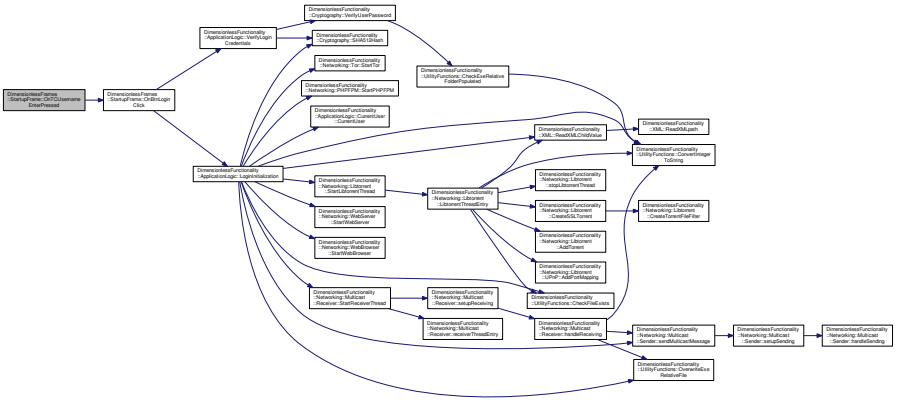
Reimplemented from StartupFrameBase.

Definition at line 279 of file StartupFrame.cpp.

References OnBtnLoginClick().

```
280     {
281         this->OnBtnLoginClick(event);
282     }
```

Here is the call graph for this function:



12.20.4.9 ThreadEntryCheckLogin()

```
void DimensionlessFrames::StartupFrame::ThreadEntryCheckLogin (
            const string & username,
            const string & password )  [private]
```

This function is called by a startupFrameThread when the "Login" button is clicked.

Parameters

in	*username*	A string containing the username of the current user attempting to login.
in	*password*	A string containing the password of the current user attempting to login.

Definition at line 307 of file StartupFrame.cpp.

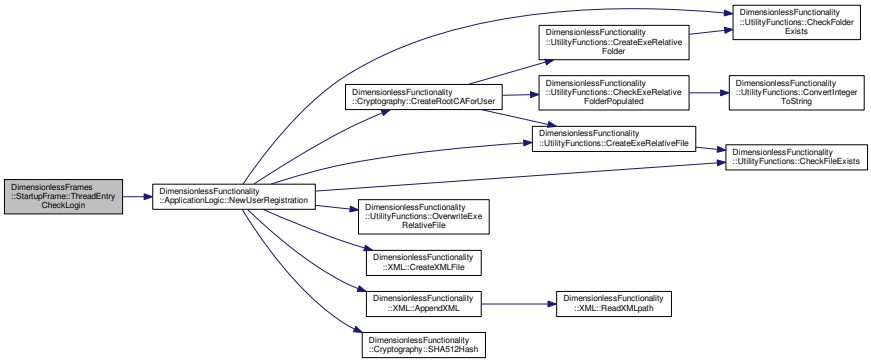
References DimensionlessFunctionality::ApplicationLogic::NewUserRegistration().

Referenced by OnBtnRegisterClicked().

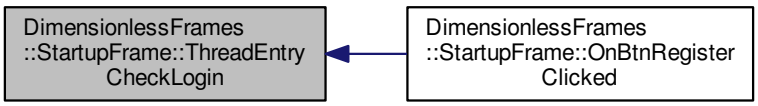
```
308     {
309         wxThreadEvent event(wxEVT_THREAD, wxID_ANY);
310 
311         try{
312             if(NewUserRegistration(username, password)){
```

```
313                 event.SetString(username);
314                 event.SetInt(0);
315             } else {
316                 event.SetString(username);
317                 event.SetInt(1);
318             }
319         } catch (std::exception& e){
320             event.SetString(e.what());
321             event.SetInt(2);
322         }
323 
324         wxPostEvent(this, event);
325     }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.20.5 Member Data Documentation

12.20.5.1 currentLoginState

```
LoginState DimensionlessFrames::StartupFrame::currentLoginState = LoginState::Login  [private]
```

This variable stores the current state of this StartupFrame instance.

Definition at line 90 of file StartupFrame.h.

Referenced by OnBtnLoginClick(), OnBtnRegisterClicked(), and StartupFrame().

12.20.5.2 startupFrameThread

```
std::thread DimensionlessFrames::StartupFrame::startupFrameThread  [static], [private]
```

This variable stores the thread instance that handles any functions called in DimensionlessFunctionality.

Definition at line 78 of file StartupFrame.h.

Referenced by OnBtnRegisterClicked().

12.20.5.3 UsersExist

```
bool DimensionlessFrames::StartupFrame::UsersExist = false  [private]
```

This boolean decides if there exist any pre-existing users for our application.

Definition at line 93 of file StartupFrame.h.

Referenced by DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::operator()(), and StartupFrame().

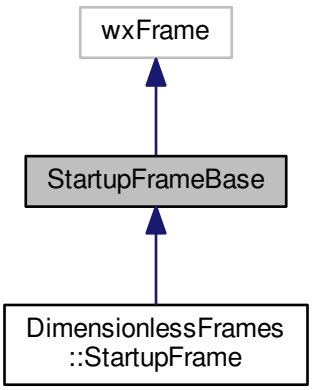
The documentation for this class was generated from the following files:

- DimensionlessClient/StartupFrame.h
- DimensionlessClient/StartupFrame.cpp

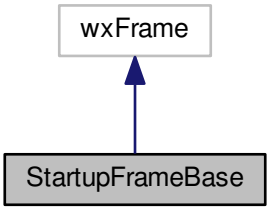
12.21 StartupFrameBase Class Reference

```
#include <Dimensionless.h>
```

Inheritance diagram for StartupFrameBase:



Collaboration diagram for StartupFrameBase:



Public Types

- enum {
  idbtnLogin = 10001, idmiFileQuit = 10002, idmiHelpAbout = 10003, idtcPassword = 10004,
  idtcUsername = 10005 }

Public Member Functions

- wxMenuBar * GetMbHeader ()
- wxStaticBitmap * GetBmpDimensionlessLogo ()
- wxStaticText * GetStRegistrationInfo ()
- wxStaticText * GetStUsername ()
- wxTextCtrl * GetTcUsername ()
- wxStaticText * GetStPassword ()
- wxTextCtrl * GetTcPassword ()
- wxStaticText * GetStRepeatPassword ()
- wxTextCtrl * GetTcRepeatPassword ()
- wxButton * GetBtnLogin ()
- wxButton * GetBtnRegister ()
- wxPanel * GetPnlLoginBackground ()
- wxPanel * GetPnlBackground ()
- StartupFrameBase (wxWindow *parent, wxWindowID id=wxID_ANY, const wxString &title=_("Welcome to Dimensionless!"), const wxPoint &pos=wxDefaultPosition, const wxSize &size=wxSize(1000, 600), long style=wxDEFAULT_FRAME_STYLE|wxTAB_TRAVERSAL)
- virtual ~StartupFrameBase ()

Protected Member Functions

- virtual void OnClose (wxCloseEvent &event)
- virtual void OnQuit (wxCommandEvent &event)
- virtual void OnAbout (wxCommandEvent &event)
- virtual void OnTCUsernameEnterPressed (wxCommandEvent &event)
- virtual void OnTCPasswordEnterPressed (wxCommandEvent &event)
- virtual void OnTCRepeatPasswordEnterPressed (wxCommandEvent &event)
- virtual void OnBtnLoginClick (wxCommandEvent &event)
- virtual void OnBtnRegisterClicked (wxCommandEvent &event)

Protected Attributes

- wxMenuBar * mbHeader
- wxMenu * mFile
- wxMenuItem * miFileQuit
- wxMenu * mHelp
- wxMenuItem * miHelpAbout
- wxPanel * pnlBackground
- wxPanel * pnlLoginBackground
- wxStaticBitmap * bmpDimensionlessLogo
- wxStaticText * stRegistrationInfo
- wxStaticText * stUsername
- wxTextCtrl * tcUsername
- wxStaticText * stPassword
- wxTextCtrl * tcPassword
- wxStaticText * stRepeatPassword
- wxTextCtrl * tcRepeatPassword
- wxButton * btnLogin
- wxButton * btnRegister

12.21.1 Detailed Description

Definition at line 50 of file Dimensionless.h.

12.21.2 Member Enumeration Documentation

12.21.2.1 anonymous enum

```
anonymous enum
```

Enumerator

idbtnLogin	
idmiFileQuit	
idmiHelpAbout	
idtcPassword	
idtcUsername	

Definition at line 53 of file Dimensionless.h.

```
53          {
54         idbtnLogin = 10001,
55         idmiFileQuit = 10002,
56         idmiHelpAbout = 10003,
57         idtcPassword = 10004,
58         idtcUsername = 10005,
59     };
```

12.21.3 Constructor & Destructor Documentation

12.21.3.1 StartupFrameBase()

```
StartupFrameBase::StartupFrameBase (
            wxWindow * parent,
            wxWindowID id = wxID_ANY,
            const wxString & title = _("Welcome to Dimensionless!"),
            const wxPoint & pos = wxDefaultPosition,
            const wxSize & size = wxSize(1000,600),
            long style = wxDEFAULT_FRAME_STYLE|wxTAB_TRAVERSAL )
```

Definition at line 16 of file Dimensionless.cpp.

References bBitmapLoaded, bmpDimensionlessLogo, btnLogin, btnRegister, idbtnLogin, idmiFileQuit, idmiHelpAbout, idtcPassword, idtcUsername, mbHeader, mFile, mHelp, miFileQuit, miHelpAbout, OnAbout(), OnBtnLoginClick(), OnBtnRegisterClicked(), OnClose(), OnQuit(), OnTCPasswordEnterPressed(), OnTCRepeatPasswordEnterPressed(), OnTCUsernameEnterPressed(), pnlBackground, pnlLoginBackground, stPassword, stRegistrationInfo, stRepeatPassword, stUsername, tcPassword, tcRepeatPassword, tcUsername, WXC_FROM_DIP, and wxCrafterAR3ID5InitBitmapResources().

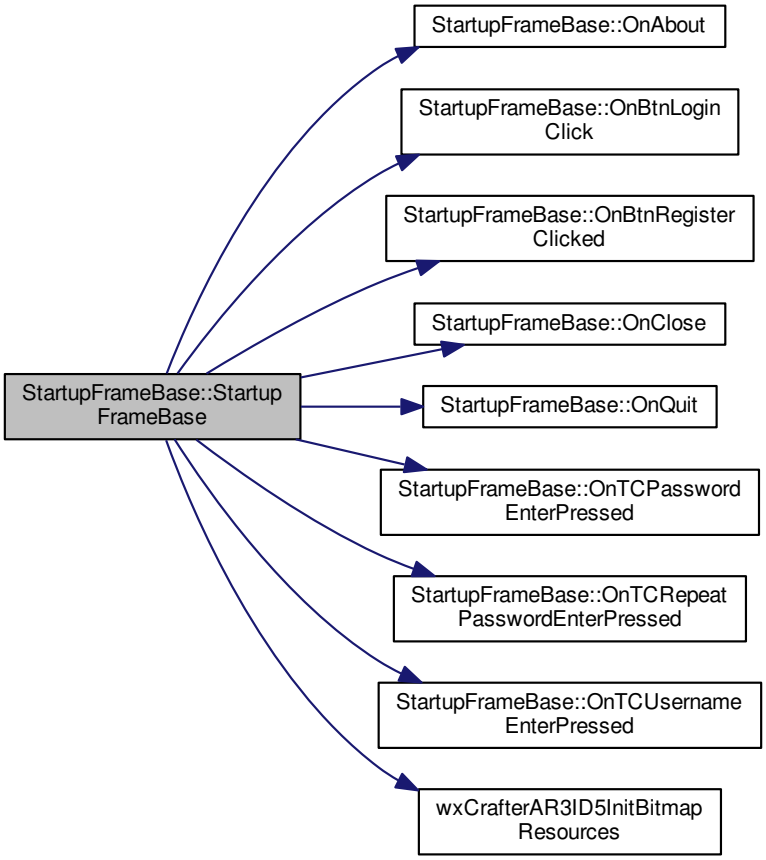
Referenced by GetPnlBackground().

```
17     : wxFrame(parent, id, title, pos, size, style)
18 {
19     if ( !bBitmapLoaded ) {
20         // We need to initialise the default bitmap handler
21         wxXmlResource::Get()->AddHandler(new wxBitmapXmlHandler);
22         wxCrafterAR3ID5InitBitmapResources();
23         bBitmapLoaded = true;
24     }
25
26     mbHeader = new wxMenuBar(0);
27     this->SetMenuBar(mbHeader);
28
29     mFile = new wxMenu();
30     mbHeader->Append(mFile, _("&File"));
31
32     miFileQuit = new wxMenuItem(mFile, idmiFileQuit, _("&Quit\tAlt+F4"), _("Quit
       the application"), wxITEM_NORMAL);
33     mFile->Append(miFileQuit);
34
35     mHelp = new wxMenu();
36     mbHeader->Append(mHelp, _("&Help"));
37
38     miHelpAbout = new wxMenuItem(mHelp, idmiHelpAbout, _("&About\tF1"), _("
      Show info about this application"), wxITEM_NORMAL);
39     mHelp->Append(miHelpAbout);
40
41     wxGridSizer* gszrStartup = new wxGridSizer(0, 1, 0, 0);
42     this->SetSizer(gszrStartup);
43
44     pnlBackground = new wxPanel(this, wxID_ANY, wxDefaultPosition, wxDLG_UNIT(this, wxSize(-1,
       -1)), wxTAB_TRAVERSAL);
45
46     gszrStartup->Add(pnlBackground, 1, wxALIGN_CENTER_HORIZONTAL|wxALIGN_CENTER_VERTICAL,
      WXC_FROM_DIP(0));
47
48     wxGridSizer* gszrInner = new wxGridSizer(0, 1, 0, 0);
49     pnlBackground->SetSizer(gszrInner);
50
51     pnlLoginBackground = new wxPanel(pnlBackground, wxID_ANY,
      wxDefaultPosition, wxDLG_UNIT(pnlBackground, wxSize(-1,-1)), wxTAB_TRAVERSAL);
52
53     gszrInner->Add(pnlLoginBackground, 0, wxALL|wxALIGN_CENTER_HORIZONTAL|
      wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
54
55     wxBoxSizer* bszrLogin = new wxBoxSizer(wxVERTICAL);
56     pnlLoginBackground->SetSizer(bszrLogin);
```

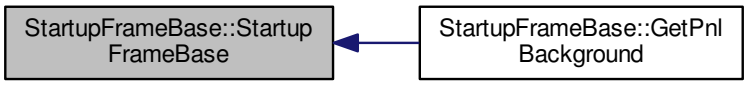
```
57
58      bmpDimensionlessLogo = new wxStaticBitmap(
       pnlLoginBackground, wxID_ANY, wxXmlResource::Get()->LoadBitmap(wxT("DimensionlessLogo")),
        wxDefaultPosition, wxDLG_UNIT(pnlLoginBackground, wxSize(250,100)), 0 );
59
60      bszrLogin->Add(bmpDimensionlessLogo, 0, wxALL|wxALIGN_CENTER_HORIZONTAL,
       WXC_FROM_DIP(5));
61      bmpDimensionlessLogo->SetMinSize(wxSize(250,100));
62
63      wxBoxSizer* bszrRegistrationInfo = new wxBoxSizer(wxHORIZONTAL);
64
65      bszrLogin->Add(bszrRegistrationInfo, 0, wxALL|wxALIGN_CENTER_HORIZONTAL,
       WXC_FROM_DIP(5));
66
67      stRegistrationInfo = new wxStaticText(pnlLoginBackground, wxID_ANY,
        _("No users detected, you must register before proceeding."), wxDefaultPosition, wxDLG_UNIT(
       pnlLoginBackground, wxSize(-1,-1)), wxALIGN_CENTRE);
68      stRegistrationInfo->Wrap(400);
69      stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
70      stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
71      stRegistrationInfo->Hide();
72
73      bszrRegistrationInfo->Add(stRegistrationInfo, 0, wxALL|wxALIGN_CENTER|
       wxALIGN_CENTER_HORIZONTAL|wxALIGN_CENTER_VERTICAL|wxRESERVE_SPACE_EVEN_IF_HIDDEN, WXC_FROM_DIP(5));
74
75      wxFlexGridSizer* fgszrUsernameAndPassword = new wxFlexGridSizer(3, 2, 0, 0);
76      fgszrUsernameAndPassword->SetFlexibleDirection( wxBOTH );
77      fgszrUsernameAndPassword->SetNonFlexibleGrowMode( wxFLEX_GROWMODE_SPECIFIED );
78
79      bszrLogin->Add(fgszrUsernameAndPassword, 0, wxALL|wxALIGN_CENTER|wxALIGN_CENTER_HORIZONTAL|
       wxALIGN_CENTER_VERTICAL, WXC_FROM_DIP(5));
80
81      stUsername = new wxStaticText(pnlLoginBackground, wxID_ANY, _("Username:"),
        wxDefaultPosition, wxDLG_UNIT(pnlLoginBackground, wxSize(-1, -1)), 0);
82
83      fgszrUsernameAndPassword->Add(stUsername, 0, wxALL|wxALIGN_CENTER,
       WXC_FROM_DIP(5));
84
85      tcUsername = new wxTextCtrl(pnlLoginBackground,
       idtcUsername, wxT(""), wxDefaultPosition, wxDLG_UNIT(
       pnlLoginBackground, wxSize(290,35)), wxTE_RICH2|wxTE_PROCESS_ENTER);
86      #if wxVERSION_NUMBER >= 3000
87      tcUsername->SetHint(wxT(""));
88      #endif
89
90      fgszrUsernameAndPassword->Add(tcUsername, 0, wxALL|wxALIGN_CENTER,
       WXC_FROM_DIP(5));
91      tcUsername->SetMinSize(wxSize(290,35));
92
93      stPassword = new wxStaticText(pnlLoginBackground, wxID_ANY, _("Password:"),
        wxDefaultPosition, wxDLG_UNIT(pnlLoginBackground, wxSize(-1, -1)), 0);
94
95      fgszrUsernameAndPassword->Add(stPassword, 0, wxALL|wxALIGN_CENTER,
       WXC_FROM_DIP(5));
96
97      tcPassword = new wxTextCtrl(pnlLoginBackground,
       idtcPassword, wxT(""), wxDefaultPosition, wxDLG_UNIT(
       pnlLoginBackground, wxSize(290,35)), wxTE_RICH2|wxTE_PROCESS_ENTER|wxTE_PASSWORD);
98      #if wxVERSION_NUMBER >= 3000
99      tcPassword->SetHint(wxT(""));
100     #endif
101
102     fgszrUsernameAndPassword->Add(tcPassword, 0, wxALL|wxALIGN_CENTER,
       WXC_FROM_DIP(5));
103     tcPassword->SetMinSize(wxSize(290,35));
104
105     stRepeatPassword = new wxStaticText(pnlLoginBackground, wxID_ANY, _("
       Repeat Password:"), wxDefaultPosition, wxDLG_UNIT(pnlLoginBackground, wxSize(-1, -1)), 0);
106     stRepeatPassword->Hide();
107
108     fgszrUsernameAndPassword->Add(stRepeatPassword, 0, wxALL|wxALIGN_CENTER,
       WXC_FROM_DIP(5));
109
110     tcRepeatPassword = new wxTextCtrl(pnlLoginBackground,
       idtcPassword, wxT(""), wxDefaultPosition, wxDLG_UNIT(
       pnlLoginBackground, wxSize(290,35)), wxTE_RICH2|wxTE_PROCESS_ENTER|wxTE_PASSWORD);
111     tcRepeatPassword->Hide();
112     #if wxVERSION_NUMBER >= 3000
113     tcRepeatPassword->SetHint(wxT(""));
114     #endif
115
116     fgszrUsernameAndPassword->Add(tcRepeatPassword, 0, wxALL|wxALIGN_CENTER,
       WXC_FROM_DIP(5));
117     tcRepeatPassword->SetMinSize(wxSize(290,35));
118
119     wxBoxSizer* bszrLoginRegisterButtons = new wxBoxSizer(wxHORIZONTAL);
120
```

```
121     bszrLogin->Add(bszrLoginRegisterButtons, 0, wxALIGN_CENTER_HORIZONTAL,
      WXC_FROM_DIP(5));
122
123     btnLogin = new wxButton(pnlLoginBackground,
      idbtnLogin, _("Login"), wxDefaultPosition, wxDLG_UNIT(
      pnlLoginBackground, wxSize(-1, -1)), 0);
124
125     bszrLoginRegisterButtons->Add(btnLogin, 0, wxALL, WXC_FROM_DIP(5));
126
127     btnRegister = new wxButton(pnlLoginBackground,
      idbtnLogin, _("Register"), wxDefaultPosition, wxDLG_UNIT(
      pnlLoginBackground, wxSize(-1, -1)), 0);
128
129     bszrLoginRegisterButtons->Add(btnRegister, 0, wxALL, WXC_FROM_DIP(5));
130
131     SetName(wxT("StartupFrameBase"));
132     SetMinClientSize(wxSize(1000,600));
133     SetSize(1000,600);
134     if (GetSizer()) {
135          GetSizer()->Fit(this);
136     }
137     if(GetParent()) {
138         CentreOnParent();
139     } else {
140         CentreOnScreen();
141     }
142 #if wxVERSION_NUMBER >= 2900
143     if(!wxPersistenceManager::Get().Find(this)) {
144         wxPersistenceManager::Get().RegisterAndRestore(this);
145     } else {
146         wxPersistenceManager::Get().Restore(this);
147     }
148 #endif
149     // Connect events
150     this->Connect(wxEVT_CLOSE_WINDOW, wxCloseEventHandler(
      StartupFrameBase::OnClose), NULL, this);
151     this->Connect(miFileQuit->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
      StartupFrameBase::OnQuit), NULL, this);
152     this->Connect(miHelpAbout->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
      StartupFrameBase::OnAbout), NULL, this);
153     tcUsername->Connect(wxEVT_COMMAND_TEXT_ENTER, wxCommandEventHandler(
      StartupFrameBase::OnTCUsernameEnterPressed), NULL, this);
154     tcPassword->Connect(wxEVT_COMMAND_TEXT_ENTER, wxCommandEventHandler(
      StartupFrameBase::OnTCPasswordEnterPressed), NULL, this);
155     tcRepeatPassword->Connect(wxEVT_COMMAND_TEXT_ENTER, wxCommandEventHandler(
      StartupFrameBase::OnTCRepeatPasswordEnterPressed), NULL,
      this);
156     btnLogin->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      StartupFrameBase::OnBtnLoginClick), NULL, this);
157     btnRegister->Connect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
      StartupFrameBase::OnBtnRegisterClicked), NULL, this);
158
159 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.21.3.2 ~StartupFrameBase()

```
StartupFrameBase::~StartupFrameBase ( )  [virtual]
```

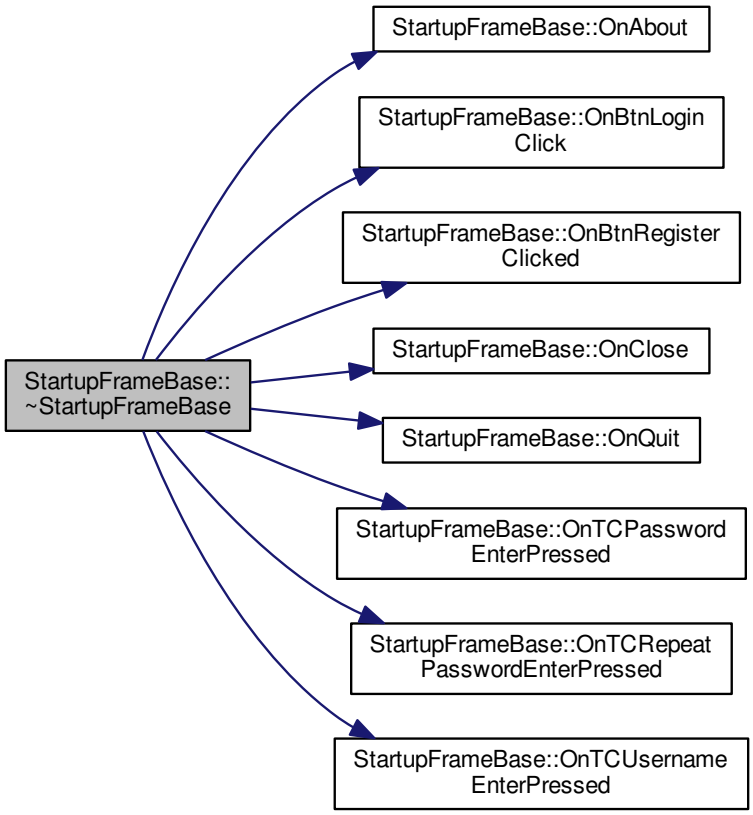
Definition at line 161 of file Dimensionless.cpp.

References btnLogin, btnRegister, miFileQuit, miHelpAbout, OnAbout(), OnBtnLoginClick(), OnBtnRegisterClicked(), OnClose(), OnQuit(), OnTCPasswordEnterPressed(), OnTCRepeatPasswordEnterPressed(), OnTCUsernameEnterPressed(), tcPassword, tcRepeatPassword, and tcUsername.

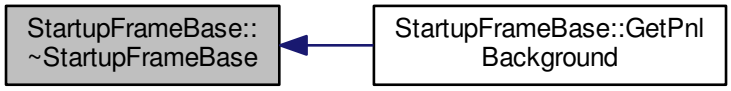
Referenced by GetPnlBackground().

```
162 {
163     this->Disconnect(wxEVT_CLOSE_WINDOW, wxCloseEventHandler(
    StartupFrameBase::OnClose), NULL, this);
164     this->Disconnect(miFileQuit->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
    StartupFrameBase::OnQuit), NULL, this);
165     this->Disconnect(miHelpAbout->GetId(), wxEVT_COMMAND_MENU_SELECTED, wxCommandEventHandler(
    StartupFrameBase::OnAbout), NULL, this);
166     tcUsername->Disconnect(wxEVT_COMMAND_TEXT_ENTER, wxCommandEventHandler(
    StartupFrameBase::OnTCUsernameEnterPressed), NULL, this);
167     tcPassword->Disconnect(wxEVT_COMMAND_TEXT_ENTER, wxCommandEventHandler(
    StartupFrameBase::OnTCPasswordEnterPressed), NULL, this);
168     tcRepeatPassword->Disconnect(wxEVT_COMMAND_TEXT_ENTER, wxCommandEventHandler(
    StartupFrameBase::OnTCRepeatPasswordEnterPressed), NULL,
    this);
169     btnLogin->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
    StartupFrameBase::OnBtnLoginClick), NULL, this);
170     btnRegister->Disconnect(wxEVT_COMMAND_BUTTON_CLICKED, wxCommandEventHandler(
    StartupFrameBase::OnBtnRegisterClicked), NULL, this);
171
172 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



12.21.4 Member Function Documentation

12.21.4.1 GetBmpDimensionlessLogo()

```
wxStaticBitmap* StartupFrameBase::GetBmpDimensionlessLogo ( )  [inline]
```

Definition at line 91 of file Dimensionless.h.

References bmpDimensionlessLogo.

```
91 { return bmpDimensionlessLogo; }
```

12.21.4.2 GetBtnLogin()

```
wxButton* StartupFrameBase::GetBtnLogin ( )  [inline]
```

Definition at line 99 of file Dimensionless.h.

References btnLogin.

```
99 { return btnLogin; }
```

12.21.4.3 GetBtnRegister()

```
wxButton* StartupFrameBase::GetBtnRegister ( )  [inline]
```

Definition at line 100 of file Dimensionless.h.

References btnRegister.

```
100 { return btnRegister; }
```

12.21.4.4 GetMbHeader()

```
wxMenuBar* StartupFrameBase::GetMbHeader ( )  [inline]
```

Definition at line 90 of file Dimensionless.h.

References mbHeader.

```
90 { return mbHeader; }
```

12.21.4.5 GetPnlBackground()

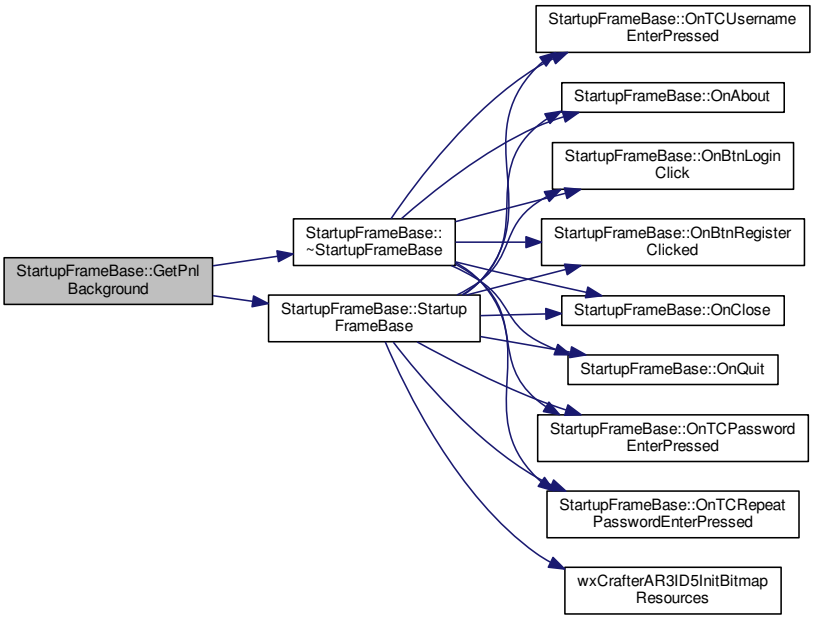
```
wxPanel* StartupFrameBase::GetPnlBackground ( )  [inline]
```

Definition at line 102 of file Dimensionless.h.

References pnlBackground, StartupFrameBase(), and ~StartupFrameBase().

```
102 { return pnlBackground; }
```

Here is the call graph for this function:



12.21.4.6 GetPnlLoginBackground()

```
wxPanel* StartupFrameBase::GetPnlLoginBackground ( )  [inline]
```

Definition at line 101 of file Dimensionless.h.

References pnlLoginBackground.

```
101 { return pnlLoginBackground; }
```

12.21.4.7 GetStPassword()

```
wxStaticText* StartupFrameBase::GetStPassword ( )  [inline]
```

Definition at line 95 of file Dimensionless.h.

References stPassword.

```
95 { return stPassword; }
```

12.21.4.8 GetStRegistrationInfo()

```
wxStaticText* StartupFrameBase::GetStRegistrationInfo ( )  [inline]
```

Definition at line 92 of file Dimensionless.h.

References stRegistrationInfo.

```
92 { return stRegistrationInfo; }
```

12.21.4.9 GetStRepeatPassword()

```
wxStaticText* StartupFrameBase::GetStRepeatPassword ( )  [inline]
```

Definition at line 97 of file Dimensionless.h.

References stRepeatPassword.

```
97 { return stRepeatPassword; }
```

12.21.4.10 GetStUsername()

```
wxStaticText* StartupFrameBase::GetStUsername ( )  [inline]
```

Definition at line 93 of file Dimensionless.h.

References stUsername.

```
93 { return stUsername; }
```

12.21.4.11 GetTcPassword()

```
wxTextCtrl* StartupFrameBase::GetTcPassword ( )  [inline]
```

Definition at line 96 of file Dimensionless.h.

References tcPassword.

```
96 { return tcPassword; }
```

12.21.4.12 GetTcRepeatPassword()

```
wxTextCtrl* StartupFrameBase::GetTcRepeatPassword ( )  [inline]
```

Definition at line 98 of file Dimensionless.h.

References tcRepeatPassword.

```
98 { return tcRepeatPassword; }
```

12.21.4.13 GetTcUsername()

```
wxTextCtrl* StartupFrameBase::GetTcUsername ( )  [inline]
```

Definition at line 94 of file Dimensionless.h.

References tcUsername.

```
94 { return tcUsername; }
```

12.21.4.14 OnAbout()

```
virtual void StartupFrameBase::OnAbout (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

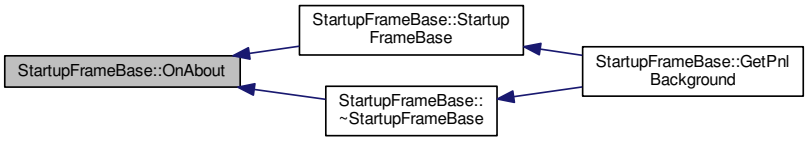
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 82 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
82 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.15 OnBtnLoginClick()

```
virtual void StartupFrameBase::OnBtnLoginClick (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

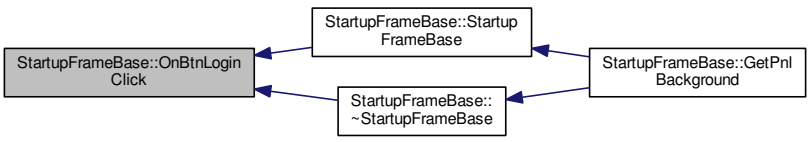
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 86 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
86 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.16 OnBtnRegisterClicked()

```
virtual void StartupFrameBase::OnBtnRegisterClicked (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

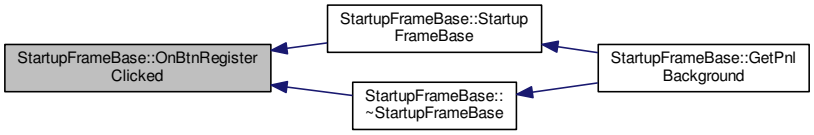
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 87 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
87 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.17 OnClose()

```
virtual void StartupFrameBase::OnClose (
            wxCloseEvent & event )  [inline], [protected], [virtual]
```

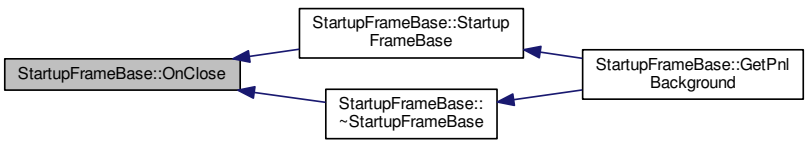
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 80 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
80 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.18 OnQuit()

```
virtual void StartupFrameBase::OnQuit (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

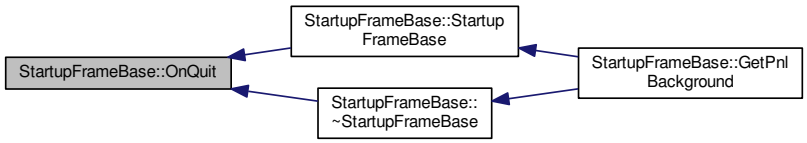
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 81 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
81 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.19 OnTCPasswordEnterPressed()

```
virtual void StartupFrameBase::OnTCPasswordEnterPressed (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

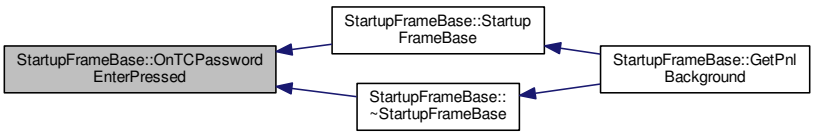
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 84 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
84 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.20 OnTCRepeatPasswordEnterPressed()

```
virtual void StartupFrameBase::OnTCRepeatPasswordEnterPressed (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

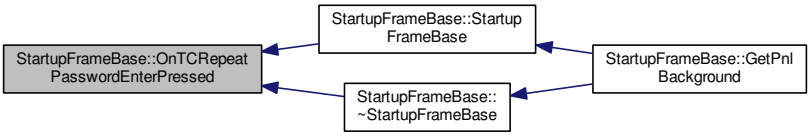
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 85 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
85 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.4.21 OnTCUsernameEnterPressed()

```
virtual void StartupFrameBase::OnTCUsernameEnterPressed (
            wxCommandEvent & event )  [inline], [protected], [virtual]
```

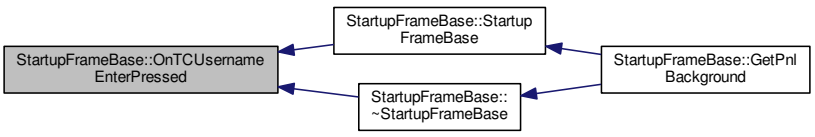
Reimplemented in DimensionlessFrames::StartupFrame.

Definition at line 83 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

```
83 { event.Skip(); }
```

Here is the caller graph for this function:



12.21.5 Member Data Documentation

12.21.5.1 bmpDimensionlessLogo

```
wxStaticBitmap* StartupFrameBase::bmpDimensionlessLogo  [protected]
```

Definition at line 68 of file Dimensionless.h.

Referenced by GetBmpDimensionlessLogo(), MainFrameBase::GetBmpDimensionlessLogo(), and StartupFrameBase().

12.21.5.2 btnLogin

```
wxButton* StartupFrameBase::btnLogin  [protected]
```

Definition at line 76 of file Dimensionless.h.

Referenced by GetBtnLogin(), DimensionlessFrames::StartupFrame::OnBtnLoginClick(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::operator()(), DimensionlessFrames::StartupFrame::StartupFrame(), StartupFrameBase(), and ~StartupFrameBase().

12.21.5.3 btnRegister

```
wxButton* StartupFrameBase::btnRegister  [protected]
```

Definition at line 77 of file Dimensionless.h.

Referenced by GetBtnRegister(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::operator()(), StartupFrameBase(), and ~StartupFrameBase().

12.21.5.4 mbHeader

```
wxMenuBar* StartupFrameBase::mbHeader  [protected]
```

Definition at line 61 of file Dimensionless.h.

Referenced by GetMbHeader(), MainFrameBase::GetMbHeader(), and StartupFrameBase().

12.21.5.5 mFile

```
wxMenu* StartupFrameBase::mFile  [protected]
```

Definition at line 62 of file Dimensionless.h.

Referenced by StartupFrameBase().

12.21.5.6 mHelp

```
wxMenu* StartupFrameBase::mHelp  [protected]
```

Definition at line 64 of file Dimensionless.h.

Referenced by StartupFrameBase().

12.21.5.7 miFileQuit

```
wxMenuItem* StartupFrameBase::miFileQuit  [protected]
```

Definition at line 63 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

12.21.5.8 miHelpAbout

```
wxMenuItem* StartupFrameBase::miHelpAbout  [protected]
```

Definition at line 65 of file Dimensionless.h.

Referenced by StartupFrameBase(), and ~StartupFrameBase().

12.21.5.9 pnlBackground

```
wxPanel* StartupFrameBase::pnlBackground  [protected]
```

Definition at line 66 of file Dimensionless.h.

Referenced by GetPnlBackground(), and StartupFrameBase().

12.21.5.10 pnlLoginBackground

```
wxPanel* StartupFrameBase::pnlLoginBackground  [protected]
```

Definition at line 67 of file Dimensionless.h.

Referenced by GetPnlLoginBackground(), and StartupFrameBase().

12.21.5.11 stPassword

```
wxStaticText* StartupFrameBase::stPassword  [protected]
```

Definition at line 72 of file Dimensionless.h.

Referenced by GetStPassword(), and StartupFrameBase().

12.21.5.12 stRegistrationInfo

```
wxStaticText* StartupFrameBase::stRegistrationInfo  [protected]
```

Definition at line 69 of file Dimensionless.h.

Referenced by GetStRegistrationInfo(), DimensionlessFrames::StartupFrame::OnBtnLoginClick(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::operator()(), DimensionlessFrames::StartupFrame::StartupFrame(), and StartupFrameBase().

12.21.5.13 stRepeatPassword

```
wxStaticText* StartupFrameBase::stRepeatPassword  [protected]
```

Definition at line 74 of file Dimensionless.h.

Referenced by GetStRepeatPassword(), DimensionlessFrames::StartupFrame::OnBtnLoginClick(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), DimensionlessFrames::StartupFrame::StartupFrame(), and StartupFrameBase().

12.21.5.14 stUsername

```
wxStaticText* StartupFrameBase::stUsername  [protected]
```

Definition at line 70 of file Dimensionless.h.

Referenced by GetStUsername(), and StartupFrameBase().

12.21.5.15 tcPassword

```
wxTextCtrl* StartupFrameBase::tcPassword  [protected]
```

Definition at line 73 of file Dimensionless.h.

Referenced by GetTcPassword(), DimensionlessFrames::StartupFrame::OnBtnLoginClick(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), StartupFrameBase(), and ~StartupFrameBase().

12.21.5.16 tcRepeatPassword

```
wxTextCtrl* StartupFrameBase::tcRepeatPassword  [protected]
```

Definition at line 75 of file Dimensionless.h.

Referenced by GetTcRepeatPassword(), DimensionlessFrames::StartupFrame::OnBtnLoginClick(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), DimensionlessFrames::StartupFrame::StartupFrame(), StartupFrameBase(), and ~StartupFrameBase().

12.21.5.17 tcUsername

```
wxTextCtrl* StartupFrameBase::tcUsername  [protected]
```

Definition at line 71 of file Dimensionless.h.

Referenced by GetTcUsername(), DimensionlessFrames::StartupFrame::OnBtnLoginClick(), DimensionlessFrames::StartupFrame::OnBtnRegisterClicked(), StartupFrameBase(), and ~StartupFrameBase().

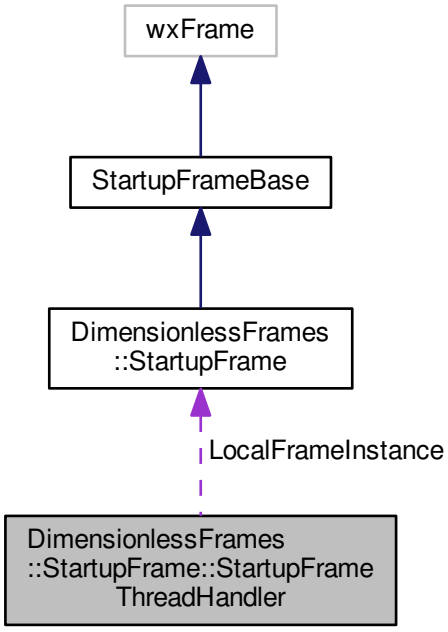
The documentation for this class was generated from the following files:

- DimensionlessClient/Dimensionless.h
- DimensionlessClient/Dimensionless.cpp

12.22 DimensionlessFrames::StartupFrame::StartupFrameThreadHandler Struct Reference

This struct handles events posted from our startupFrameThread.

Collaboration diagram for DimensionlessFrames::StartupFrame::StartupFrameThreadHandler:



Public Member Functions

- StartupFrameThreadHandler (StartupFrame *frame)
- void operator() (wxThreadEvent &evt)

Private Attributes

- StartupFrame * LocalFrameInstance = nullptr

  A raw pointer reference to a SearchForUsersFrame to handle events for.

12.22.1 Detailed Description

This struct handles events posted from our startupFrameThread.

Definition at line 41 of file StartupFrame.cpp.

12.22.2 Constructor & Destructor Documentation

12.22.2.1 StartupFrameThreadHandler()

```
DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::StartupFrameThreadHandler (
            StartupFrame * frame )  [inline]
```

This constructor initializes a new StartupFrameThreadHandler object with a pointer to the current StartupFrame instance so it may be modified as required.

Parameters

in	*frame*	The raw pointer to a StartupFrame instance.

Definition at line 48 of file StartupFrame.cpp.

References LocalFrameInstance.

```
49             {
50                 // Set the local StartupFrame reference to that passed.
51                 LocalFrameInstance = frame;
52             }
```

12.22.3 Member Function Documentation

12.22.3.1 operator()()

```
void DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::operator() (
            wxThreadEvent & evt )  [inline]
```

This function remaps the () operator of StartupFrameThreadHandler to handle startupFrameThread events.

Parameters

in	*event*	A wxThreadEvent object posted by our startupFrameThread.

Definition at line 58 of file StartupFrame.cpp.

References StartupFrameBase::btnLogin, StartupFrameBase::btnRegister, LocalFrameInstance, StartupFrameBase::stRegistrationInfo, and DimensionlessFrames::StartupFrame::UsersExist.

```
59             {
60                 switch(evt.GetId()){
61                     case wxID_ANY:
62                     {
63                         if(evt.GetInt() == 0){
```

```
64                              if(evt.GetString().length() > 50){
65                                  LocalFrameInstance->
    stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(0,204,0)")));
66                                  LocalFrameInstance->
    stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(0,204,0)")));
67                                  LocalFrameInstance->
    stRegistrationInfo->SetLabel("Success! New user: " + evt.GetString().substr(0,50) + "...
     created. Go back and login or register another user.");
68                                  LocalFrameInstance->
    stRegistrationInfo->Show();
69                              } else {
70                                  LocalFrameInstance->
    stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(0,204,0)")));
71                                  LocalFrameInstance->
    stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(0,204,0)")));
72                                  LocalFrameInstance->
    stRegistrationInfo->SetLabel("Success! New user: " + evt.GetString() + " created. Go back
     and login or register another user.");
73                                  LocalFrameInstance->
    stRegistrationInfo->Show();
74                              }
75                              LocalFrameInstance->
    stRegistrationInfo->Wrap(400);
76                              LocalFrameInstance->Layout();
77                              wxLogMessage(_("Success! New user: " + evt.GetString() + " created."));
78                              if(LocalFrameInstance->UsersExist == false){
79                                  LocalFrameInstance->UsersExist = true;
80                                  LocalFrameInstance->btnLogin->Enable(true);
81                              }
82                          } else if(evt.GetInt() == 1) {
83                              LocalFrameInstance->
    stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
84                              LocalFrameInstance->
    stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
85                              LocalFrameInstance->
    stRegistrationInfo->SetLabel("Error! Something went wrong creating new user: " + evt.
    GetString());
86                              LocalFrameInstance->
    stRegistrationInfo->Show();
87                              LocalFrameInstance->Layout();
88                          } else {
89                              LocalFrameInstance->
    stRegistrationInfo->SetBackgroundColour(wxColour(wxT("rgb(255,0,0)")));
90                              LocalFrameInstance->
    stRegistrationInfo->SetForegroundColour(wxColour(wxT("rgb(255,0,0)")));
91                              LocalFrameInstance->
    stRegistrationInfo->Show();
92                              LocalFrameInstance->
    stRegistrationInfo->SetLabel(evt.GetString());
93                              LocalFrameInstance->
    stRegistrationInfo->Wrap(400);
94                              LocalFrameInstance->Layout();
95                              wxLogMessage(evt.GetString());
96                          }
97                          LocalFrameInstance->btnLogin->Enable(true);
98                          LocalFrameInstance->btnRegister->Enable(true);
99                          break;
100                     }
101                     default:
102                     {
103                         break;
104                     }
105                 }
106             }
```

12.22.4 Member Data Documentation

12.22.4.1 LocalFrameInstance

```
StartupFrame* DimensionlessFrames::StartupFrame::StartupFrameThreadHandler::LocalFrameInstance
= nullptr  [private]
```

A raw pointer reference to a SearchForUsersFrame to handle events for.

Definition at line 109 of file StartupFrame.cpp.

Referenced by operator()(), and StartupFrameThreadHandler().

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

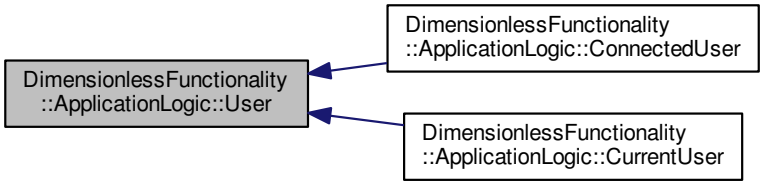
- DimensionlessClient/StartupFrame.cpp

12.23 DimensionlessFunctionality::ApplicationLogic::User Class Reference

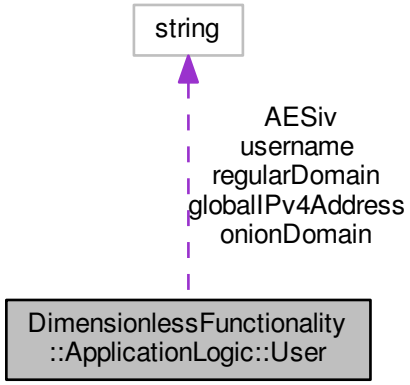
This class acts as a base class for ConnectedUser and CurrentUser, consisting of attributes and methods that should be common to both classes.

`#include <DimensionlessFunctionality.h>`

Inheritance diagram for DimensionlessFunctionality::ApplicationLogic::User:



Collaboration diagram for DimensionlessFunctionality::ApplicationLogic::User:



Public Types

- enum Platform { Platform::mobile, Platform::desktop, Platform::null }

  This enumeration decides the type of platform a User is on.
- enum Status {
  Status::online, Status::offline, Status::request, Status::rejected,
  Status::null }

  This enumeration decides the status of a User.

Public Member Functions

- User (const string &aesiv, const string &newusername, const string &globalipv4address, const string &onion-domain, const string &regulardomain)
- ~User ()
- string toMulticastString ()
- string toSourceTargetString ()
- string toFoundString ()

Public Attributes

- boost::posix_time::ptime lastPageUpdatedTime
- bool newPageAvailable = false
- bool newMessageAvailable = false
- string AESiv = ""
- string username = ""
- string globalIPv4Address = ""
- string onionDomain = ""
- string regularDomain = ""
- int libtorrentPort = 6881
- int webPort = 1337
- int OpenVPNPort = 7331

12.23.1 Detailed Description

This class acts as a base class for ConnectedUser and CurrentUser, consisting of attributes and methods that should be common to both classes.

Definition at line 749 of file DimensionlessFunctionality.h.

12.23.2 Member Enumeration Documentation

12.23.2.1 Platform

```
enum DimensionlessFunctionality::ApplicationLogic::User::Platform [strong]
```

This enumeration decides the type of platform a User is on.

Enumerator

mobile	Denotes that a User is on a mobile device.
desktop	Denotes that a User is on a desktop computer.
null	Denotes that a lack of a platform type has been selected.

Definition at line 758 of file DimensionlessFunctionality.h.

```
758                                     {
759                                        mobile,
760                                        desktop,
761                                        null
762                                      };
```

12.23.2.2 Status

```
enum DimensionlessFunctionality::ApplicationLogic::User::Status  [strong]
```

This enumeration decides the status of a User.

Enumerator

online	Denotes that a User is online and available to communicate.
offline	Denotes that a User is offline and not available to communicate.
request	Denotes that a connection request has been sent to a User.
rejected	Denotes that a User has rejected or been rejected a connection request.
null	Denotes that a lack of a status type has been selected.

Definition at line 765 of file DimensionlessFunctionality.h.

```
765                                    {
766                                       online,
767                                       offline,
768                                       request,
769                                       rejected,
770                                       null
771                                     };
```

12.23.3 Constructor & Destructor Documentation

12.23.3.1 User()

```
DimensionlessFunctionality::ApplicationLogic::User::User (
            const string & aesiv,
            const string & newusername,
            const string & globalipv4address,
            const string & oniondomain,
            const string & regulardomain )
```

This constructor initializes a new User object with the variable values we want. The User class acts as a base for the ConnectedUser and CurrentUser classes.

Parameters

in	*aesiv*	The AES IV of the User object to be created.
in	*newusername*	The username of the User object to be created.
in	*globalipv4address*	The IPv4 address of the User object to be created.
in	*oniondomain*	The onion domain of the User object to be created.
in	*regulardomain*	The regular FreeDNS domain of the User object to be created.

Definition at line 1798 of file DimensionlessFunctionality.cpp.

```
1799         {
1800             this->AESiv = aesiv;
1801             this->username = newusername;
1802             this->globalIPv4Address = globalipv4address;
1803             this->onionDomain = oniondomain;
1804             this->regularDomain = regulardomain;
1805         }
```

12.23.3.2 ~User()

```
DimensionlessFunctionality::ApplicationLogic::User::~User ( )
```

This destructor currently does nothing.

Definition at line 1810 of file DimensionlessFunctionality.cpp.

```
1811         {
1812         }
```

12.23.4 Member Function Documentation

12.23.4.1 toFoundString()

```
string DimensionlessFunctionality::ApplicationLogic::User::toFoundString ( )
```

This function provides a single string containing a User object's username, IPv4 address, onion domain, and regular domain (along with port number) in a comma separated format.

Returns

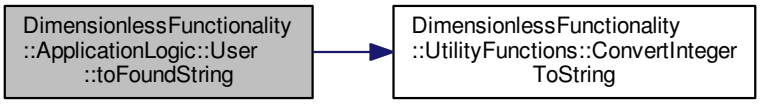
A comma separated string containing the data found by an Nmap scan for display in a SearchForUsersFrame instance.

Definition at line 1861 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString().

```
1862         {
1863             return username + ", " +
1864                    globalIPv4Address + ":" +
1865                    UtilityFunctions::ConvertIntegerToString(
      webPort) + ", " +
1866                    onionDomain + ":" +
1867                    UtilityFunctions::ConvertIntegerToString(
      webPort) + ", " +
1868                    regularDomain + ":" +
1869                    UtilityFunctions::ConvertIntegerToString(
      webPort);
1870         }
```

Here is the call graph for this function:



12.23.4.2 toMulticastString()

```
string DimensionlessFunctionality::ApplicationLogic::User::toMulticastString ( )
```

This function provides a single string containing a User object's username, IPv4 address, onion domain, regular domain, libtorrent port, web port and vpn port in a comma separated format.

Returns

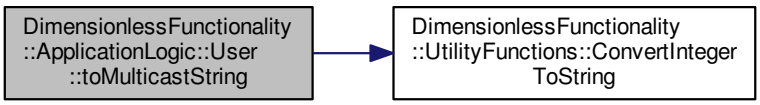
A comma separated string containing the data required to be sent over multicast for a particular user.

Definition at line 1818 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString().

```
1819         {
1820             string libport = UtilityFunctions::ConvertIntegerToString
     (libtorrentPort);
1821             string webport = UtilityFunctions::ConvertIntegerToString
     (webPort);
1822             string vpnport = UtilityFunctions::ConvertIntegerToString
     (OpenVPNPort);
1823             return UtilityFunctions::ConvertIntegerToString(
     username.length()) + "," +
1824                    UtilityFunctions::ConvertIntegerToString(
     globalIPv4Address.length()) + "," +
1825                    UtilityFunctions::ConvertIntegerToString(
     onionDomain.length()) + "," +
1826                    UtilityFunctions::ConvertIntegerToString(
     regularDomain.length()) + "," +
1827                    UtilityFunctions::ConvertIntegerToString(libport
     .length()) + "," +
1828                    UtilityFunctions::ConvertIntegerToString(webport
     .length()) + "," +
1829                    UtilityFunctions::ConvertIntegerToString(vpnport
     .length()) + "," +
1830                    username + "," +
1831                    globalIPv4Address + "," +
1832                    onionDomain + "," +
1833                    regularDomain + "," +
1834                    libport + "," +
1835                    webport + "," +
1836                    vpnport;
1837         }
```

Here is the call graph for this function:



12.23.4.3 toSourceTargetString()

```
string DimensionlessFunctionality::ApplicationLogic::User::toSourceTargetString ( )
```

This function provides a single string containing a User object's username, AES IV, IPv4 address, onion domain, and regular domain in a comma separated format.

Returns

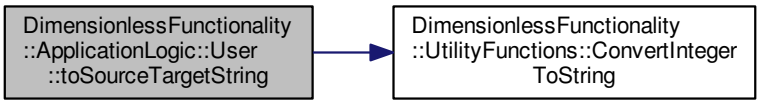
A comma separated string containing the data required for a request to be sent by PHP for a particular user.

Definition at line 1843 of file DimensionlessFunctionality.cpp.

References DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString().

```
1844         {
1845             return UtilityFunctions::ConvertIntegerToString(
     username.length()) + "," +
1846                    UtilityFunctions::ConvertIntegerToString(
     AESiv.length()) + "," +
1847                    UtilityFunctions::ConvertIntegerToString(
     globalIPv4Address.length()) + "," +
1848                    UtilityFunctions::ConvertIntegerToString(
     onionDomain.length()) + "," +
1849                    UtilityFunctions::ConvertIntegerToString(
     regularDomain.length()) + "," +
1850                    username + "," +
1851                    AESiv + "," +
1852                    globalIPv4Address + "," +
1853                    onionDomain + "," +
1854                    regularDomain;
1855         }
```

Here is the call graph for this function:



12.23.5 Member Data Documentation

12.23.5.1 AESiv

```
string DimensionlessFunctionality::ApplicationLogic::User::AESiv = ""
```

Definition at line 796 of file DimensionlessFunctionality.h.

12.23.5.2 globalIPv4Address

```
string DimensionlessFunctionality::ApplicationLogic::User::globalIPv4Address = ""
```

Definition at line 800 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString().

12.23.5.3 lastPageUpdatedTime

```
boost::posix_time::ptime DimensionlessFunctionality::ApplicationLogic::User::lastPageUpdated↩
Time
```

Definition at line 790 of file DimensionlessFunctionality.h.

12.23.5.4 libtorrentPort

```
int DimensionlessFunctionality::ApplicationLogic::User::libtorrentPort = 6881
```

Definition at line 806 of file DimensionlessFunctionality.h.

12.23.5.5 newMessageAvailable

```
bool DimensionlessFunctionality::ApplicationLogic::User::newMessageAvailable = false
```

Definition at line 794 of file DimensionlessFunctionality.h.

12.23.5.6 newPageAvailable

```
bool DimensionlessFunctionality::ApplicationLogic::User::newPageAvailable = false
```

Definition at line 792 of file DimensionlessFunctionality.h.

12.23.5.7 onionDomain

```
string DimensionlessFunctionality::ApplicationLogic::User::onionDomain = ""
```

Definition at line 802 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString().

12.23.5.8 OpenVPNPort

```
int DimensionlessFunctionality::ApplicationLogic::User::OpenVPNPort = 7331
```

Definition at line 810 of file DimensionlessFunctionality.h.

12.23.5.9 regularDomain

```
string DimensionlessFunctionality::ApplicationLogic::User::regularDomain = ""
```

Definition at line 804 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString().

12.23.5.10 username

```
string DimensionlessFunctionality::ApplicationLogic::User::username = ""
```

Definition at line 798 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::ApplicationLogic::NewUserRegistration(), DimensionlessFunctionality::ApplicationLogic::RefreshConnectedUsers(), DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest(), DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest(), and DimensionlessFunctionality::ApplicationLogic::CurrentUser::toDataString().

12.23.5.11 webPort

```
int DimensionlessFunctionality::ApplicationLogic::User::webPort = 1337
```

Definition at line 808 of file DimensionlessFunctionality.h.

Referenced by DimensionlessFunctionality::ApplicationLogic::LoginInitialization(), and DimensionlessFunctionality::ApplicationLogic::LogOutCleanup().

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

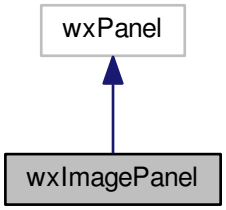
- DimensionlessClient/DimensionlessFunctionality.h
- DimensionlessClient/DimensionlessFunctionality.cpp

12.24 wxImagePanel Class Reference

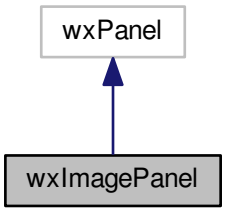
Class wxImagePanel.

```
#include <DimensionlessControls.h>
```

Inheritance diagram for wxImagePanel:



Collaboration diagram for wxImagePanel:



Classes

- struct handler

  This functor allows for events on the wxImagePanel to be handled.

Public Member Functions

- wxImagePanel (wxFrame ∗parent, wxString file, wxBitmapType format)
- wxImagePanel (wxFrame ∗parent, wxBitmap img)
- ∼wxImagePanel ()

Protected Attributes

- wxBitmap image

  Local instance of a bitmap object to be used as the background for the wxImagePanel.
- int w = -1

  A object wide variable that stores the width of the wxImagePanel at a given time.
- int h = -1

  A object wide variable that stores the height of the wxImagePanel at a given time.

12.24.1 Detailed Description

Class wxImagePanel.

The wxImagePanel class provides a control that implements wxPanel and adds the ability to provide a background image on the panel itself.

Definition at line 50 of file DimensionlessControls.h.

12.24.2 Constructor & Destructor Documentation

12.24.2.1 wxImagePanel() [1/2]

```
wxImagePanel::wxImagePanel (
            wxFrame * parent,
            wxString file,
            wxBitmapType format )
```

Description

This overload of the wxImagePanel constructor allows for a background bitmap to be loaded from a path provided by a string.

Constructor Body

Parameters

in	*parent*	A reference to the parent frame of this control.
in	*file*	A string that contains the file path of an image to display, supports all formats wxInitAllImageHandlers() turns on and supports.
in	*format*	The format of the image file passed (a wxBITMAP macro such as wxBITMAP_PNG for a .png file).

Definition at line 30 of file DimensionlessControls.cpp.

References image.

```
30                                                                          : wxPanel( parent ){
32      // Load the bitmap file.
33      image.LoadFile( file, format );
34
35      // Ensure bitmap is valid.
36      if( image.IsOk() != true )
37      {
38          // Display error if bitmap loading has had any errors.
39          wxMessageBox( _( "Cannot load background image." ), _( "Error" ), wxICON_ERROR );
40      }
41
42      // Bind events.
43      this->Bind( wxEVT_PAINT, handler( this ), this->GetId() );
44      this->Bind( wxEVT_SIZE, handler( this ), this->GetId() );
46  }
```

12.24.2.2 wxImagePanel() [2/2]

```
wxImagePanel::wxImagePanel (
              wxFrame * parent,
              wxBitmap bmp )
```

Description

This overload of the wxImagePanel constructor allows for a background bitmap to be loaded from a bitmap object.

Constructor Body

Parameters

in	*parent*	A reference to the parent frame of this control.
in	*bmp*	A wxBitmap object to be drawn on the panel.

Definition at line 56 of file DimensionlessControls.cpp.

References image.

```
56                                                          : wxPanel( parent ){
58      // Assign the passed bitmap image to the local instance.
59      image = bmp;
60
61      // Ensure bitmap is valid.
62      if( image.IsOk() != true )
63      {
64          // Display error if bitmap loading has had any errors.
65          wxMessageBox( _( "Cannot load background image." ), _( "Error" ), wxICON_ERROR );
```

```
66     }
67
68     // Bind events.
69     this->Bind( wxEVT_PAINT, handler( this ), this->GetId() );
70     this->Bind( wxEVT_SIZE, handler( this ), this->GetId() );
72 }
```

12.24.2.3 ~wxImagePanel()

```
wxImagePanel::~wxImagePanel ( )
```

Description

The destructor of the wxImagePanel simply unbinds events binded in its construction and ceases background drawing.

Destructor Body

Definition at line 80 of file DimensionlessControls.cpp.

References image.

```
81 {
83     // Run the bitmap object's destructor to ensure the LoginFrame can exit correctly if required.
84     image.~wxBitmap();
85
86     // Unbind events.
87     this->Unbind( wxEVT_PAINT, handler( this ), this->GetId() );
88     this->Unbind( wxEVT_SIZE, handler( this ), this->GetId() );
90 }
```

12.24.3 Member Data Documentation

12.24.3.1 h

```
int wxImagePanel::h = -1  [protected]
```

A object wide variable that stores the height of the wxImagePanel at a given time.

Definition at line 62 of file DimensionlessControls.h.

12.24.3.2 image

```
wxBitmap wxImagePanel::image  [protected]
```

Local instance of a bitmap object to be used as the background for the wxImagePanel.

Definition at line 56 of file DimensionlessControls.h.

Referenced by wxImagePanel(), and ~wxImagePanel().

12.24.3.3 w

```
int wxImagePanel::w = -1  [protected]
```

A object wide variable that stores the width of the wxImagePanel at a given time.

Definition at line 59 of file DimensionlessControls.h.

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

- DimensionlessClient/DimensionlessControls.h
- DimensionlessClient/DimensionlessControls.cpp

Chapter 13

File Documentation

13.1 DimensionlessClient/ChatFrame.cpp File Reference

```
#include "ChatFrame.h"
#include "MainFrame.h"
```

Include dependency graph for ChatFrame.cpp:

Classes

- struct DimensionlessFrames::ChatFrame::ChatFrameThreadHandler

  This struct handles events posted from our chatFrameThread.

Namespaces

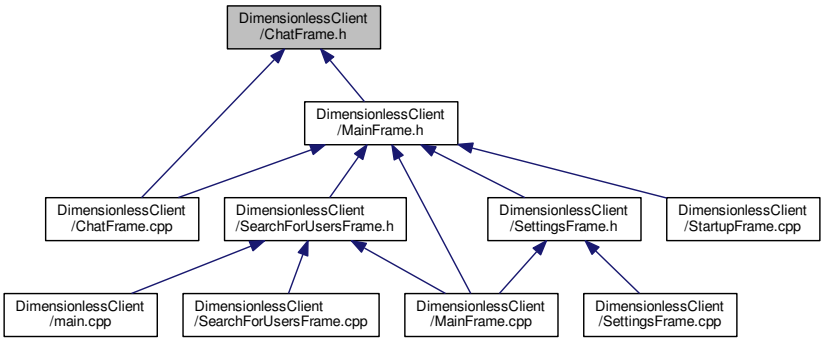
- DimensionlessFrames

13.2 DimensionlessClient/ChatFrame.h File Reference

```
#include "DimensionlessFunctionality.h"
#include "Dimensionless.h"
```

Include dependency graph for ChatFrame.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DimensionlessFrames::ChatUser

  This class stores a reference to a ConnectedUser that the current user can chat with in the ChatFrame.
- class DimensionlessFrames::ChatFrame

  This class creates a ChatFrame to chat with other ConnectedUsers.

Namespaces

- DimensionlessFrames

13.3 DimensionlessClient/Dimensionless.cpp File Reference

```
#include "Dimensionless.h"
```

Include dependency graph for Dimensionless.cpp:

Functions

- void wxCrafterAR3ID5InitBitmapResources ()

Variables

- static bool bBitmapLoaded = false

13.3.1 Function Documentation

13.3.1.1 wxCrafterAR3ID5InitBitmapResources()

```
void wxCrafterAR3ID5InitBitmapResources ( )
```

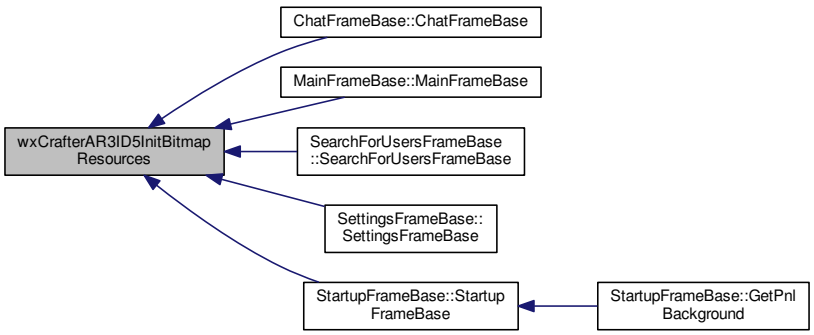
Definition at line 832 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

References xml_res_file_0, xml_res_file_1, xml_res_size_0, xml_res_size_1, and XRC_ADD_FILE.

Referenced by ChatFrameBase::ChatFrameBase(), MainFrameBase::MainFrameBase(), SearchForUsers↩FrameBase::SearchForUsersFrameBase(), SettingsFrameBase::SettingsFrameBase(), and StartupFrameBase::↩StartupFrameBase().

```
833 {
834
835     // Check for memory FS. If not present, load the handler:
836     {
837         wxMemoryFSHandler::AddFile(wxT("XRC_resource/dummy_file"), wxT("dummy one"));
838         wxFileSystem fsys;
839         wxFSFile *f = fsys.OpenFile(wxT("memory:XRC_resource/dummy_file"));
840         wxMemoryFSHandler::RemoveFile(wxT("XRC_resource/dummy_file"));
841         if (f) delete f;
842         else wxFileSystem::AddHandler(new wxMemoryFSHandlerBase);
843     }
844
845     XRC_ADD_FILE(wxT("
      XRC_resource/Dimensionless_dimensionlessclient_bitmaps.cpp$.._Resources_DimensionlessLogo.png"), xml_res_file_0, xml_res
846     XRC_ADD_FILE(wxT("
      XRC_resource/Dimensionless_dimensionlessclient_bitmaps.cpp$_home_ccc_Documents_Dimensionless_DimensionlessClient_Dimens
      xml_res_file_1, xml_res_size_1, wxT("text/xml"));
847     wxXmlResource::Get()->Load(wxT("
      memory:XRC_resource/Dimensionless_dimensionlessclient_bitmaps.cpp$_home_ccc_Documents_Dimensionless_DimensionlessClient_
848 }
```

Here is the caller graph for this function:



13.3.2 Variable Documentation

13.3.2.1 bBitmapLoaded

```
bool bBitmapLoaded = false  [static]
```

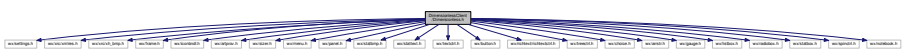
Definition at line 13 of file Dimensionless.cpp.

Referenced by ChatFrameBase::ChatFrameBase(), MainFrameBase::MainFrameBase(), SearchForUsers↩ FrameBase::SearchForUsersFrameBase(), SettingsFrameBase::SettingsFrameBase(), and StartupFrameBase::↩ StartupFrameBase().

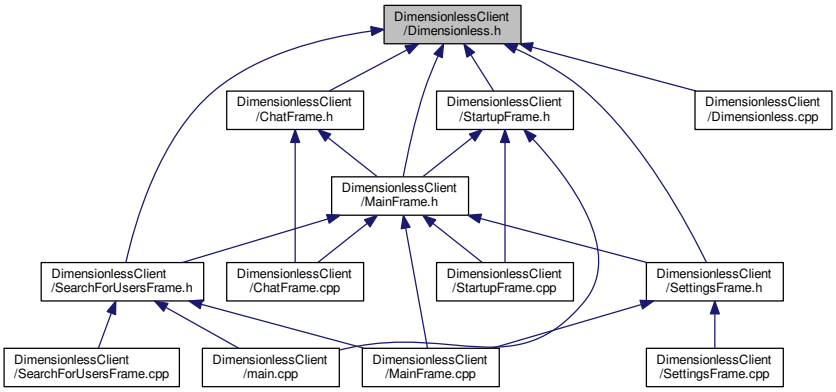
13.4 DimensionlessClient/Dimensionless.h File Reference

```
#include <wx/settings.h>
#include <wx/xrc/xmlres.h>
#include <wx/xrc/xh_bmp.h>
#include <wx/frame.h>
#include <wx/iconbndl.h>
#include <wx/artprov.h>
#include <wx/sizer.h>
#include <wx/menu.h>
#include <wx/panel.h>
#include <wx/statbmp.h>
#include <wx/stattext.h>
#include <wx/textctrl.h>
#include <wx/button.h>
#include <wx/richtext/richtextctrl.h>
#include <wx/treectrl.h>
#include <wx/choice.h>
#include <wx/arrstr.h>
#include <wx/gauge.h>
#include <wx/listbox.h>
#include <wx/radiobox.h>
#include <wx/statbox.h>
#include <wx/spinctrl.h>
#include <wx/notebook.h>
```

Include dependency graph for Dimensionless.h:



This graph shows which files directly or indirectly include this file:



Classes

- class StartupFrameBase
- class MainFrameBase
- class SearchForUsersFrameBase
- class SettingsFrameBase
- class ChatFrameBase

Macros

- #define WXC_FROM_DIP(x) x

13.4.1 Macro Definition Documentation

13.4.1.1 WXC_FROM_DIP

```
#define WXC_FROM_DIP(
              x ) x
```

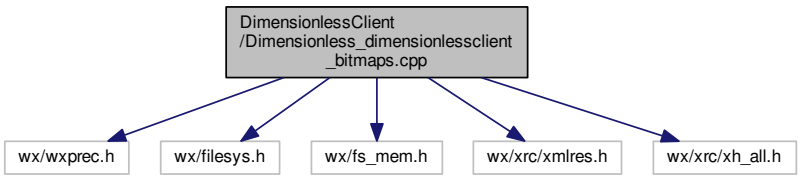
Definition at line 46 of file Dimensionless.h.

Referenced by DimensionlessFrames::ChatFrame::AddNewChat(), ChatFrameBase::ChatFrameBase(), MainFrameBase::MainFrameBase(), SearchForUsersFrameBase::SearchForUsersFrameBase(), SettingsFrameBase::SettingsFrameBase(), and StartupFrameBase::StartupFrameBase().

13.5 DimensionlessClient/Dimensionless_dimensionlessclient_bitmaps.cpp File Reference

```
#include <wx/wxprec.h>
#include <wx/filesys.h>
#include <wx/fs_mem.h>
#include <wx/xrc/xmlres.h>
#include <wx/xrc/xh_all.h>
```

Include dependency graph for Dimensionless_dimensionlessclient_bitmaps.cpp:



Macros

- #define XRC_ADD_FILE(name, data, size, mime) wxMemoryFSHandler::AddFile(name, data, size)

Functions

- void wxCrafterAR3ID5InitBitmapResources ()

Variables

- static size_t xml_res_size_0 = 15995
- static unsigned char xml_res_file_0 [ ]
- static size_t xml_res_size_1 = 279
- static unsigned char xml_res_file_1 [ ]

13.5.1 Macro Definition Documentation

13.5.1.1 XRC_ADD_FILE

```
#define XRC_ADD_FILE(
              name,
              data,
              size,
              mime ) wxMemoryFSHandler::AddFile(name, data, size)
```

Definition at line 20 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

Referenced by wxCrafterAR3ID5InitBitmapResources().

13.5.2 Function Documentation

13.5.2.1 wxCrafterAR3ID5InitBitmapResources()

```
void wxCrafterAR3ID5InitBitmapResources ( )
```

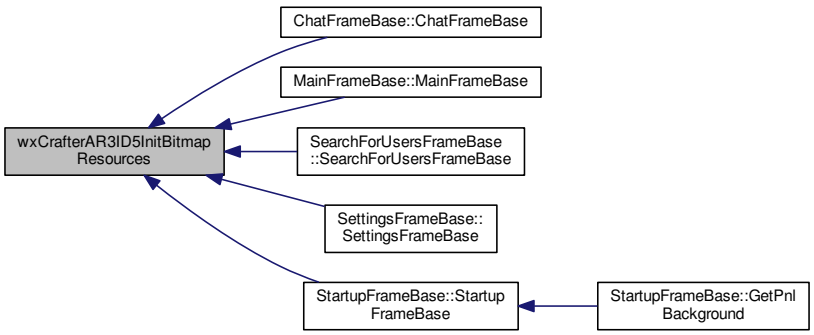
Definition at line 832 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

References xml_res_file_0, xml_res_file_1, xml_res_size_0, xml_res_size_1, and XRC_ADD_FILE.

Referenced by ChatFrameBase::ChatFrameBase(), MainFrameBase::MainFrameBase(), SearchForUsers↩FrameBase::SearchForUsersFrameBase(), SettingsFrameBase::SettingsFrameBase(), and StartupFrameBase::↩StartupFrameBase().

```
833 {
834
835     // Check for memory FS. If not present, load the handler:
836     {
837         wxMemoryFSHandler::AddFile(wxT("XRC_resource/dummy_file"), wxT("dummy one"));
838         wxFileSystem fsys;
839         wxFSFile *f = fsys.OpenFile(wxT("memory:XRC_resource/dummy_file"));
840         wxMemoryFSHandler::RemoveFile(wxT("XRC_resource/dummy_file"));
841         if (f) delete f;
842         else wxFileSystem::AddHandler(new wxMemoryFSHandlerBase);
843     }
844
845     XRC_ADD_FILE(wxT("
      XRC_resource/Dimensionless_dimensionlessclient_bitmaps.cpp$.._Resources_DimensionlessLogo.png"), xml_res_file_0, xml_res
846     XRC_ADD_FILE(wxT("
      XRC_resource/Dimensionless_dimensionlessclient_bitmaps.cpp$_home_ccc_Documents_Dimensionless_DimensionlessClient_Dimens
      xml_res_file_1, xml_res_size_1, wxT("text/xml"));
847     wxXmlResource::Get()->Load(wxT("
      memory:XRC_resource/Dimensionless_dimensionlessclient_bitmaps.cpp$_home_ccc_Documents_Dimensionless_DimensionlessClient_
848 }
```

Here is the caller graph for this function:



13.5.3 Variable Documentation

13.5.3.1 xml_res_file_0

```
unsigned char xml_res_file_0[]  [static]
```

Definition at line 25 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

Referenced by wxCrafterAR3ID5InitBitmapResources().

13.5.3.2 xml_res_file_1

```
unsigned char xml_res_file_1[]  [static]
```

Initial value:

```
= {
60,63,120,109,108,32,118,101,114,115,105,111,110,61,34,49,46,48,34,32,101,
110,99,111,100,105,110,103,61,34,85,84,70,45,56,34,63,62,10,60,114,101,
115,111,117,114,99,101,32,120,109,108,110,115,61,34,104,116,116,112,58,
47,47,119,119,119,46,119,120,119,105,100,103,101,116,115,46,111,114,103,
47,119,120,120,114,99,34,62,10,32,32,60,33,45,45,32,72,97,110,100,108,101,
114,32,71,101,110,101,114,97,116,105,111,110,32,105,115,32,79,78,32,45,
45,62,10,32,32,60,111,98,106,101,99,116,32,99,108,97,115,115,61,34,119,
120,66,105,116,109,97,112,34,32,110,97,109,101,61,34,68,105,109,101,110,
115,105,111,110,108,101,115,115,76,111,103,111,34,62,68,105,109,101,110,
115,105,111,110,108,101,115,115,95,100,105,109,101,110,115,105,111,110,
108,101,115,115,99,108,105,101,110,116,95,98,105,116,109,97,112,115,46,
99,112,112,36,46,46,95,82,101,115,111,117,114,99,101,115,95,68,105,109,
101,110,115,105,111,110,108,101,115,115,76,111,103,111,46,112,110,103,60,
47,111,98,106,101,99,116,62,10,60,47,114,101,115,111,117,114,99,101,62,
10}
```

Definition at line 815 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

Referenced by wxCrafterAR3ID5InitBitmapResources().

13.5.3.3 xml_res_size_0

```
size_t xml_res_size_0 = 15995  [static]
```

Definition at line 24 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

Referenced by wxCrafterAR3ID5InitBitmapResources().

13.5.3.4 xml_res_size_1

```
size_t xml_res_size_1 = 279  [static]
```

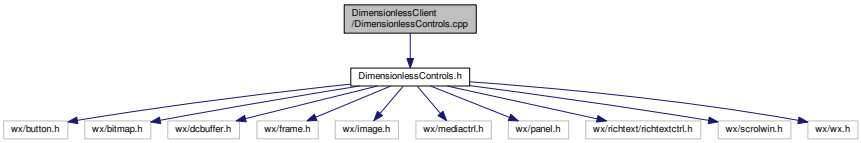
Definition at line 814 of file Dimensionless_dimensionlessclient_bitmaps.cpp.

Referenced by wxCrafterAR3ID5InitBitmapResources().

13.6 DimensionlessClient/DimensionlessControls.cpp File Reference

```
#include "DimensionlessControls.h"
```

Include dependency graph for DimensionlessControls.cpp:



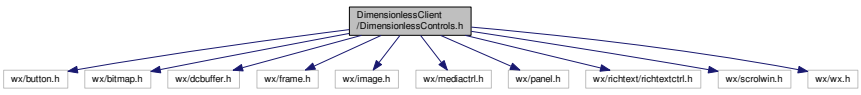
13.6.1 Detailed Description

Preprocessor Directives

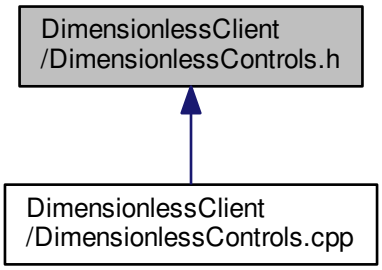
13.7 DimensionlessClient/DimensionlessControls.h File Reference

```
#include <wx/button.h>
#include <wx/bitmap.h>
#include <wx/dcbuffer.h>
#include <wx/frame.h>
#include <wx/image.h>
#include <wx/mediactrl.h>
#include <wx/panel.h>
#include <wx/richtext/richtextctrl.h>
#include <wx/scrolwin.h>
#include <wx/wx.h>
```

Include dependency graph for DimensionlessControls.h:



This graph shows which files directly or indirectly include this file:



Classes

- class wxImagePanel

  Class wxImagePanel.
- struct wxImagePanel::handler

  This functor allows for events on the wxImagePanel to be handled.

Macros

- #define DIMENSIONLESSCONTROLS_H

13.7.1 Detailed Description

Preprocessor Directives

13.7.2 Macro Definition Documentation

13.7.2.1 DIMENSIONLESSCONTROLS_H

```
#define DIMENSIONLESSCONTROLS_H
```

Header include guard used to prevent multiple include errors.

Definition at line 23 of file DimensionlessControls.h.

13.8 DimensionlessClient/DimensionlessFunctionality.cpp File Reference

```
#include "DimensionlessFunctionality.h"
```

Include dependency graph for DimensionlessFunctionality.cpp:

Namespaces

- DimensionlessFunctionality
- DimensionlessFunctionality::UtilityFunctions
- DimensionlessFunctionality::Compression
- DimensionlessFunctionality::Cryptography
- DimensionlessFunctionality::Networking
- DimensionlessFunctionality::Networking::Curl
- DimensionlessFunctionality::Networking::GeoIP
- DimensionlessFunctionality::Networking::Libtorrent
- DimensionlessFunctionality::Networking::Libtorrent::UPnP
- DimensionlessFunctionality::Networking::Multicast
- DimensionlessFunctionality::Networking::Multicast::Sender
- DimensionlessFunctionality::Networking::Multicast::Receiver
- DimensionlessFunctionality::Networking::Nmap
- DimensionlessFunctionality::Networking::OpenVPN
- DimensionlessFunctionality::Networking::PHPFPM
- DimensionlessFunctionality::Networking::Tor
- DimensionlessFunctionality::Networking::WebBrowser
- DimensionlessFunctionality::Networking::WebServer
- DimensionlessFunctionality::Validation
- DimensionlessFunctionality::XML
- DimensionlessFunctionality::ApplicationLogic

Functions

- string DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString (int integerToConvert)
- bool DimensionlessFunctionality::UtilityFunctions::CheckFileExists (const string &absoluteFilePath)
- bool DimensionlessFunctionality::UtilityFunctions::CheckFolderExists (const string &absoluteFolderPath)
- bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile (const string &exeRelativeFilePath)
- bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile (const string &exeRelativeFilePath, const string &fileContent)
- bool DimensionlessFunctionality::UtilityFunctions::OverwriteExeRelativeFile (const string &exeRelativeFile↩ Path, const string &fileContent)
- bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFolder (const string &exeRelative↩ FolderPath)
- bool DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated (const string &exe↩ RelativeFolderPath)
- string DimensionlessFunctionality::Compression::Compress (string dataToCompress)
- string DimensionlessFunctionality::Compression::Decompress (string dataToDecompress)
- string DimensionlessFunctionality::Cryptography::SHA512Hash (const string &dataToHash)
- bool DimensionlessFunctionality::Cryptography::VerifyUserPassword (const string &username, const string &password)

- bool DimensionlessFunctionality::Cryptography::CreateRootCAForUser (const string &username, const string &password)
- bool DimensionlessFunctionality::Cryptography::CreatePeerCertificateForUser (const string &username, const string &targetUsername, const string &aesiv, const string &password)
- string DimensionlessFunctionality::Networking::Curl::GetWebsiteTitle (const string &ipOrURL)
- string DimensionlessFunctionality::Networking::Curl::GetTorWebsiteTitle (const string &onionURL)
- string DimensionlessFunctionality::Networking::Curl::UpdateFreeDNSAfraidOrgDomain (const string &free↩ DNSAfraidOrgURL)
- void DimensionlessFunctionality::Networking::GeoIP::UpdateIPv4AddressList ()
- std::map< string, std::vector< string > > DimensionlessFunctionality::Networking::GeoIP::GetCity↩ CountryList ()
- std::vector< string > DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCityXorCountry (const string &cityxorcountry)
- std::vector< string > DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCountryAnd↩ City (const string &country, const string &city)
- bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::AddPortMapping (libtorrent::session_↩ handle::protocol_type protocolType, int externalPort, int internalPort)
- bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemovePortMapping (int index)
- bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemoveAllPortMappings ()
- vector< string > DimensionlessFunctionality::Networking::Libtorrent::ScanForTorrentFiles (const string &scanDirectoryRoot)
- bool DimensionlessFunctionality::Networking::Libtorrent::CreateTorrentFileFilter (std::string const &file)
- void DimensionlessFunctionality::Networking::Libtorrent::CreateTorrent (const string &username, const string &persistentAuthComment)
- void DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent (const string &username, const string &persistentAuthComment, const string &rootCertificatePath)
- void DimensionlessFunctionality::Networking::Libtorrent::AddTorrent (const string &torrentPath, const string &peerCertificatePath, const string &privateKeyPath, const string &dhParamsPath, const string &ourAESIV)
- void DimensionlessFunctionality::Networking::Libtorrent::RemoveTorrent (const string &targetUsername, const bool &deleteFiles)
- void DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry (const string &target↩ Username, const string &targetPassword)
- void DimensionlessFunctionality::Networking::Libtorrent::StartLibtorrentThread (const string &target↩ Username, const string &targetPassword)
- void DimensionlessFunctionality::Networking::Libtorrent::StopLibtorrentThread ()
- void DimensionlessFunctionality::Networking::Multicast::Sender::setupSending (boost::asio::io_service &senderIOService, const string &multicastAddress, const string &message)
- void DimensionlessFunctionality::Networking::Multicast::Sender::handleSending (const boost::system↩ ::error_code &error)
- void DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage (const string &multicastAddress, const string &message)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::setupReceiving (const string &listen↩ Address, const string &multicastAddress)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving (const boost::system↩ ::error_code &error, size_t bytesReceived)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::receiverThreadEntry ()
- void DimensionlessFunctionality::Networking::Multicast::Receiver::StartReceiverThread (const string &listenAddress, const string &multicastAddress)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::StopReceiverThread ()
- vector< string > DimensionlessFunctionality::Networking::Nmap::ScanIPAddressesWithPort (const string &username, const vector< string > &IPv4Addresses, const int &port)
- void DimensionlessFunctionality::Networking::OpenVPN::StartOpenVPNClient (const string &vpn↩ ConfigurationDirectory)
- void DimensionlessFunctionality::Networking::OpenVPN::StartOpenVPNServer (const string &vpn↩ ConfigurationDirectory)

- void DimensionlessFunctionality::Networking::OpenVPN::StopOpenVPNClient (const string &vpnConfigurationDirectory)
- void DimensionlessFunctionality::Networking::OpenVPN::StopOpenVPNServer (const string &vpnConfigurationDirectory)
- void DimensionlessFunctionality::Networking::PHPFPM::StartPHPFPM (const string &webDirectory)
- void DimensionlessFunctionality::Networking::PHPFPM::StopPHPFPM (const string &webDirectory)
- void DimensionlessFunctionality::Networking::Tor::StartTor (const string &webDirectory)
- void DimensionlessFunctionality::Networking::Tor::StopTor (const string &webDirectory)
- void DimensionlessFunctionality::Networking::WebBrowser::StartWebBrowser (const string &url)
- void DimensionlessFunctionality::Networking::WebBrowser::StopWebBrowser ()
- void DimensionlessFunctionality::Networking::WebServer::StartWebServer (const string &webDirectory, const string &password)
- void DimensionlessFunctionality::Networking::WebServer::StopWebServer (const string &webDirectory)
- bool DimensionlessFunctionality::XML::CreateXMLFile (string pathToCreateFile, string rootNodeName)
- bool DimensionlessFunctionality::XML::WriteXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributeValues)
- bool DimensionlessFunctionality::XML::AppendXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributeValues)
- bool DimensionlessFunctionality::XML::EditXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributesValues)
- vector< string > DimensionlessFunctionality::XML::ReadXMLValues (string pathToFile, string rootNode, vector< string > childNames)
- vector< string > DimensionlessFunctionality::XML::ReadXMLAttributeValues (string pathToFile, string rootNode, string childName, vector< string > attributeNames)
- string DimensionlessFunctionality::XML::ReadXMLChildValue (string pathToFile, string rootNode, string childName)
- string DimensionlessFunctionality::XML::ReadXMLAttribute (string pathToFile, string rootNode, string childName, string childAttribute)
- vector< string > DimensionlessFunctionality::XML::GetXMLElementsAndAttributes (string pathToFile, string rootNode, string childNameLike, vector< string > attributeNames)
- ptree DimensionlessFunctionality::XML::ReadXMLpath (string pathToFile)
- ptree DimensionlessFunctionality::XML::ReadXMLstr (string &data)
- void DimensionlessFunctionality::ApplicationLogic::RefreshConnectedUsers ()
- void DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest (const Request &request, const std::shared_ptr< ConnectedUser > &targetUser, const string &extradata)
- void DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest (std::atomic< unsigned int > &activeRequestHandlingUsers, const Request &request, const std::shared_ptr< ConnectedUser > &targetUser, const string &extradata)
- std::vector< std::shared_ptr< ConnectedUser > > DimensionlessFunctionality::ApplicationLogic::ProcessFoundUsers (const string &foundUsers)
- bool DimensionlessFunctionality::ApplicationLogic::LoginInitialization (const string &username, const string &password)
- bool DimensionlessFunctionality::ApplicationLogic::VerifyLoginCredentials (const string &username, const string &password)
- bool DimensionlessFunctionality::ApplicationLogic::NewUserRegistration (const string &username, const string &password)
- bool DimensionlessFunctionality::ApplicationLogic::LogOutCleanup (const string &username)

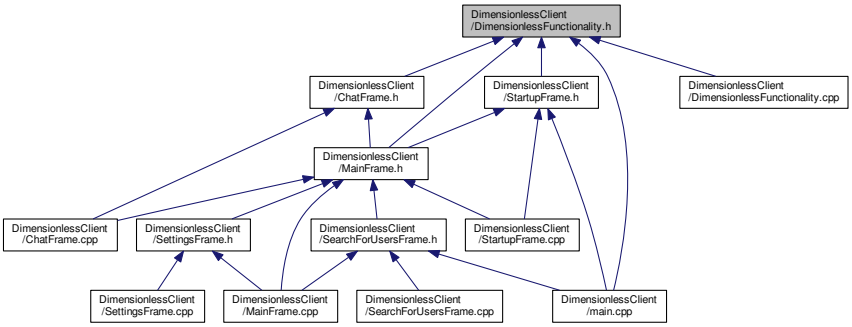
13.9 DimensionlessClient/DimensionlessFunctionality.h File Reference

```
#include <algorithm>
#include <deque>
#include <fstream>
#include <future>
```

```
#include <iostream>
#include <memory>
#include <sstream>
#include <thread>
#include <vector>
#include <boost/algorithm/string.hpp>
#include <boost/asio.hpp>
#include <boost/bind.hpp>
#include <boost/date_time/posix_time/posix_time.hpp>
#include <boost/date_time/posix_time/posix_time_types.hpp>
#include <boost/filesystem.hpp>
#include <boost/foreach.hpp>
#include <boost/iostreams/copy.hpp>
#include <boost/iostreams/filtering_streambuf.hpp>
#include <boost/iostreams/filter/zlib.hpp>
#include <boost/nondet_random.hpp>
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/xml_parser.hpp>
#include <boost/random/uniform_int.hpp>
#include <boost/random/variate_generator.hpp>
#include <boost/regex.hpp>
#include <libtorrent/add_torrent_params.hpp>
#include <libtorrent/alert_types.hpp>
#include <libtorrent/bencode.hpp>
#include <libtorrent/create_torrent.hpp>
#include <libtorrent/entry.hpp>
#include <libtorrent/error_code.hpp>
#include <libtorrent/file.hpp>
#include <libtorrent/file_pool.hpp>
#include <libtorrent/hasher.hpp>
#include <libtorrent/hex.hpp>
#include <libtorrent/session.hpp>
#include <libtorrent/storage.hpp>
#include <libtorrent/torrent_handle.hpp>
#include <libtorrent/torrent_info.hpp>
```

Include dependency graph for DimensionlessFunctionality.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DimensionlessFunctionality::ApplicationLogic::User

  This class acts as a base class for ConnectedUser and CurrentUser, consisting of attributes and methods that should be common to both classes.
- class DimensionlessFunctionality::ApplicationLogic::ConnectedUser

  This class denotes a user that is connected to the static CurrentUser that is logged into the application.
- class DimensionlessFunctionality::ApplicationLogic::CurrentUser

  This class denotes the User that is currently logged into the application.
- class DimensionlessFunctionality::ApplicationLogic::CurrentUserSession

  This singleton class stores a smart shared pointer reference to the currently logged in user and a collection of their connected users.

Namespaces

- DimensionlessFunctionality
- DimensionlessFunctionality::UtilityFunctions
- DimensionlessFunctionality::Compression
- DimensionlessFunctionality::Cryptography
- DimensionlessFunctionality::Networking
- DimensionlessFunctionality::Networking::Curl
- DimensionlessFunctionality::Networking::GeoIP
- DimensionlessFunctionality::Networking::Libtorrent
- DimensionlessFunctionality::Networking::Libtorrent::UPnP
- DimensionlessFunctionality::Networking::Multicast
- DimensionlessFunctionality::Networking::Multicast::Sender
- DimensionlessFunctionality::Networking::Multicast::Receiver
- DimensionlessFunctionality::Networking::Nmap
- DimensionlessFunctionality::Networking::OpenVPN
- DimensionlessFunctionality::Networking::PHPFPM
- DimensionlessFunctionality::Networking::Tor
- DimensionlessFunctionality::Networking::WebBrowser
- DimensionlessFunctionality::Networking::WebServer
- DimensionlessFunctionality::Validation
- DimensionlessFunctionality::XML
- DimensionlessFunctionality::ApplicationLogic

Enumerations

- enum DimensionlessFunctionality::Validation::ValidationType { DimensionlessFunctionality::Validation::ValidationType::Null, DimensionlessFunctionality::Validation::↩ ValidationType::DataType, DimensionlessFunctionality::Validation::ValidationType::Lookup, Dimensionless↩ Functionality::Validation::ValidationType::Format, DimensionlessFunctionality::Validation::ValidationType::Length, DimensionlessFunctionality::Validation::↩ ValidationType::Presence }

  This enumeration decides the type of validation to be performed.
- enum DimensionlessFunctionality::Validation::FormatType { DimensionlessFunctionality::Validation::↩ FormatType::Null, DimensionlessFunctionality::Validation::FormatType::ZlibCompressed }

  This enumeration decides the format of the data to be validated for the format check.

- enum DimensionlessFunctionality::ApplicationLogic::Request { DimensionlessFunctionality::ApplicationLogic::Request::SendConnectionRequest, DimensionlessFunctionality::ApplicationLogic::Request::AcceptConnectionRequest, DimensionlessFunctionality::ApplicationLogic::Request::RejectConnectionRequest, DimensionlessFunctionality::ApplicationLogic::Request::SendMessage, DimensionlessFunctionality::ApplicationLogic::Request::PerformSearch, DimensionlessFunctionality::ApplicationLogic::Request::GetOurRemoteIP, DimensionlessFunctionality::ApplicationLogic::Request::SetOurXML, DimensionlessFunctionality::ApplicationLogic::Request::SetOurTorrent, DimensionlessFunctionality::ApplicationLogic::Request::GetConnectedUserXML, DimensionlessFunctionality::ApplicationLogic::Request::GetConnectedUserTorrent, DimensionlessFunctionality::ApplicationLogic::Request::Null }

  This enumeration decides the type of request that our CurrentUser wants to send to their connectedUsers.

Functions

- string DimensionlessFunctionality::UtilityFunctions::ConvertIntegerToString (int integerToConvert)
- bool DimensionlessFunctionality::UtilityFunctions::CheckFileExists (const string &absoluteFilePath)
- bool DimensionlessFunctionality::UtilityFunctions::CheckFolderExists (const string &absoluteFolderPath)
- bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile (const string &exeRelativeFilePath)
- bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFile (const string &exeRelativeFilePath, const string &fileContent)
- bool DimensionlessFunctionality::UtilityFunctions::OverwriteExeRelativeFile (const string &exeRelativeFilePath, const string &fileContent)
- bool DimensionlessFunctionality::UtilityFunctions::CreateExeRelativeFolder (const string &exeRelativeFolderPath)
- bool DimensionlessFunctionality::UtilityFunctions::CheckExeRelativeFolderPopulated (const string &exeRelativeFolderPath)
- string DimensionlessFunctionality::Compression::Compress (string dataToCompress)
- string DimensionlessFunctionality::Compression::Decompress (string dataToDecompress)
- string DimensionlessFunctionality::Cryptography::SHA512Hash (const string &dataToHash)
- bool DimensionlessFunctionality::Cryptography::VerifyUserPassword (const string &username, const string &password)
- bool DimensionlessFunctionality::Cryptography::CreateRootCAForUser (const string &username, const string &password)
- bool DimensionlessFunctionality::Cryptography::CreatePeerCertificateForUser (const string &username, const string &targetUsername, const string &aesiv, const string &password)
- string DimensionlessFunctionality::Networking::Curl::GetWebsiteTitle (const string &ipOrURL)
- string DimensionlessFunctionality::Networking::Curl::GetTorWebsiteTitle (const string &onionURL)
- string DimensionlessFunctionality::Networking::Curl::UpdateFreeDNSAfraidOrgDomain (const string &freeDNSAfraidOrgURL)
- void DimensionlessFunctionality::Networking::GeoIP::UpdateIPv4AddressList ()
- std::map< string, std::vector< string > > DimensionlessFunctionality::Networking::GeoIP::GetCityCountryList ()
- std::vector< string > DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCityXorCountry (const string &cityxorcountry)
- std::vector< string > DimensionlessFunctionality::Networking::GeoIP::GetIPv4AddressesForCountryAndCity (const string &country, const string &city)
- bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::AddPortMapping (libtorrent::session_handle::protocol_type protocolType, int externalPort, int internalPort)
- bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemovePortMapping (int index)
- bool DimensionlessFunctionality::Networking::Libtorrent::UPnP::RemoveAllPortMappings ()
- static std::atomic< bool > DimensionlessFunctionality::Networking::Libtorrent::stopLibtorrentThread (false)

  This atomic boolean variable decides whether it is time to stop the libtorrent thread.
- vector< string > DimensionlessFunctionality::Networking::Libtorrent::ScanForTorrentFiles (const string &scanDirectoryRoot)

- bool DimensionlessFunctionality::Networking::Libtorrent::CreateTorrentFileFilter (std::string const &file)
- void DimensionlessFunctionality::Networking::Libtorrent::CreateTorrent (const string &username, const string &persistentAuthComment)
- void DimensionlessFunctionality::Networking::Libtorrent::CreateSSLTorrent (const string &username, const string &persistentAuthComment, const string &rootCertificatePath)
- void DimensionlessFunctionality::Networking::Libtorrent::AddTorrent (const string &torrentPath, const string &peerCertificatePath, const string &privateKeyPath, const string &dhParamsPath, const string &ourAESIV)
- void DimensionlessFunctionality::Networking::Libtorrent::RemoveTorrent (const string &targetUsername, const bool &deleteFiles)
- void DimensionlessFunctionality::Networking::Libtorrent::LibtorrentThreadEntry (const string &target↩ Username, const string &targetPassword)
- void DimensionlessFunctionality::Networking::Libtorrent::StartLibtorrentThread (const string &target↩ Username, const string &targetPassword)
- void DimensionlessFunctionality::Networking::Libtorrent::StopLibtorrentThread ()
- void DimensionlessFunctionality::Networking::Multicast::Sender::setupSending (boost::asio::io_service &senderIOService, const string &multicastAddress, const string &message)
- void DimensionlessFunctionality::Networking::Multicast::Sender::handleSending (const boost::system↩ ::error_code &error)
- void DimensionlessFunctionality::Networking::Multicast::Sender::sendMulticastMessage (const string &multicastAddress, const string &message)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::setupReceiving (const string &listen↩ Address, const string &multicastAddress)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::handleReceiving (const boost::system↩ ::error_code &error, size_t bytesReceived)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::receiverThreadEntry ()
- void DimensionlessFunctionality::Networking::Multicast::Receiver::StartReceiverThread (const string &listenAddress, const string &multicastAddress)
- void DimensionlessFunctionality::Networking::Multicast::Receiver::StopReceiverThread ()
- vector< string > DimensionlessFunctionality::Networking::Nmap::ScanIPAddressesWithPort (const string &username, const vector< string > &IPv4Addresses, const int &port)
- void DimensionlessFunctionality::Networking::OpenVPN::StartOpenVPNClient (const string &vpn↩ ConfigurationDirectory)
- void DimensionlessFunctionality::Networking::OpenVPN::StartOpenVPNServer (const string &vpn↩ ConfigurationDirectory)
- void DimensionlessFunctionality::Networking::OpenVPN::StopOpenVPNClient (const string &vpn↩ ConfigurationDirectory)
- void DimensionlessFunctionality::Networking::OpenVPN::StopOpenVPNServer (const string &vpn↩ ConfigurationDirectory)
- void DimensionlessFunctionality::Networking::PHPFPM::StartPHPFPM (const string &webDirectory)
- void DimensionlessFunctionality::Networking::PHPFPM::StopPHPFPM (const string &webDirectory)
- void DimensionlessFunctionality::Networking::Tor::StartTor (const string &webDirectory)
- void DimensionlessFunctionality::Networking::Tor::StopTor (const string &webDirectory)
- void DimensionlessFunctionality::Networking::WebBrowser::StartWebBrowser (const string &url)
- void DimensionlessFunctionality::Networking::WebBrowser::StopWebBrowser ()
- void DimensionlessFunctionality::Networking::WebServer::StartWebServer (const string &webDirectory, const string &password)
- void DimensionlessFunctionality::Networking::WebServer::StopWebServer (const string &webDirectory)
- template<typename Obtained , typename CheckableParam >
  bool DimensionlessFunctionality::Validation::validate (ValidationType typeOfValidation, Obtained obtained↩ Value, CheckableParam valueToValidateAgainst, FormatType format=FormatType::Null)
- template<>
  bool DimensionlessFunctionality::Validation::validate< string, int > (ValidationType typeOfValidation, string obtainedValue, int valueToValidateAgainst, FormatType format)
- template<>
  bool DimensionlessFunctionality::Validation::validate< string, string > (ValidationType typeOfValidation, string obtainedValue, string valueToValidateAgainst, FormatType format)

- bool DimensionlessFunctionality::XML::CreateXMLFile (string pathToCreateFile, string rootNodeName)
- bool DimensionlessFunctionality::XML::WriteXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributeValues)
- bool DimensionlessFunctionality::XML::AppendXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributeValues)
- bool DimensionlessFunctionality::XML::EditXML (string pathToFile, string rootNode, string childName, string childValue, vector< string > childAttributes, vector< string > childAttributesValues)
- vector< string > DimensionlessFunctionality::XML::ReadXMLValues (string pathToFile, string rootNode, vector< string > childNames)
- vector< string > DimensionlessFunctionality::XML::ReadXMLAttributeValues (string pathToFile, string root↩ Node, string childName, vector< string > attributeNames)
- string DimensionlessFunctionality::XML::ReadXMLChildValue (string pathToFile, string rootNode, string childName)
- string DimensionlessFunctionality::XML::ReadXMLAttribute (string pathToFile, string rootNode, string child↩ Name, string childAttribute)
- vector< string > DimensionlessFunctionality::XML::GetXMLElementsAndAttributes (string pathToFile, string rootNode, string childNameLike, vector< string > attributeNames)
- ptree DimensionlessFunctionality::XML::ReadXMLpath (string pathToFile)
- ptree DimensionlessFunctionality::XML::ReadXMLstr (string &data)
- void DimensionlessFunctionality::ApplicationLogic::RefreshConnectedUsers ()
- void DimensionlessFunctionality::ApplicationLogic::SendConnectedUserRequest (const Request &request, const std::shared_ptr< ConnectedUser > &targetUser, const string &extradata)
- void DimensionlessFunctionality::ApplicationLogic::SendMultithreadedConnectedUserRequest (std↩ ::atomic< unsigned int > &activeRequestHandlingUsers, const Request &request, const std::shared_ptr< ConnectedUser > &targetUser, const string &extradata)
- std::vector< std::shared_ptr< ConnectedUser > > DimensionlessFunctionality::ApplicationLogic::↩ ProcessFoundUsers (const string &foundUsers)
- bool DimensionlessFunctionality::ApplicationLogic::LoginInitialization (const string &username, const string &password)
- bool DimensionlessFunctionality::ApplicationLogic::VerifyLoginCredentials (const string &username, const string &password)
- bool DimensionlessFunctionality::ApplicationLogic::NewUserRegistration (const string &username, const string &password)
- bool DimensionlessFunctionality::ApplicationLogic::LogOutCleanup (const string &username)

Variables

- static std::vector< int > DimensionlessFunctionality::Networking::Libtorrent::UPnP::portMappings

  This integer vector stores a collection of reference to port mappings created by the AddPortMapping function.
- static std::thread DimensionlessFunctionality::Networking::Libtorrent::libtorrentThread

  This variable stores out libtorrent thread instance.
- static std::unique_ptr< session > DimensionlessFunctionality::Networking::Libtorrent::currentSession

  This smart unique pointer stores a reference to a libtorrent session.
- static std::unique_ptr< settings_pack > DimensionlessFunctionality::Networking::Libtorrent::currentSettings

  This smart unique pointer stores a reference to our current libtorrent session's settings.
- static std::map< boost::asio::ip::address, string > DimensionlessFunctionality::Networking::Multicast::local↩ UserMap

  This map of IP addresses and strings stores a reference of the multicast users discovered on our LAN.
- static std::thread DimensionlessFunctionality::Networking::Multicast::Receiver::receiverThread

  This variable stores out multicast receiver thread instance.
- static std::unique_ptr< boost::asio::io_service > DimensionlessFunctionality::Networking::Multicast::↩ Receiver::receiverIOService

  This smart unique pointer stores a reference to a receiver io_service instance.

- static boost::asio::ip::address DimensionlessFunctionality::Networking::Multicast::Receiver::currentMulticastAddress

  This variable stores the current multicast address group that our receiver thread is listening to.
- static boost::asio::ip::udp::endpoint DimensionlessFunctionality::Networking::Multicast::Receiver::senderEndpoint

  This variable stores the sender of the last received multicast message.
- static std::unique_ptr< boost::asio::ip::udp::socket > DimensionlessFunctionality::Networking::Multicast::Receiver::receiverSocket

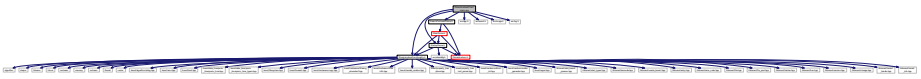
  This smart unique pointer stores a reference to a socket that listens for multicast messages.
- static std::vector< char > DimensionlessFunctionality::Networking::Multicast::Receiver::data = std::vector<char>(1024)

  This char vector stores the last received multicast message.

13.10 DimensionlessClient/main.cpp File Reference

```
#include <DimensionlessFunctionality.h>
#include "StartupFrame.h"
#include "SearchForUsersFrame.h"
#include <wx/app.h>
#include <wx/event.h>
#include <wx/image.h>
#include <wx/log.h>
```

Include dependency graph for main.cpp:



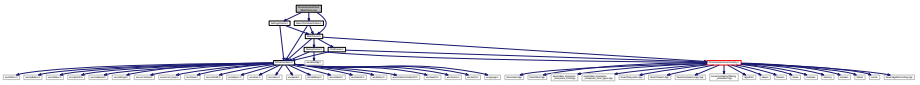
Classes

- class MainApp

13.11 DimensionlessClient/MainFrame.cpp File Reference

```
#include "MainFrame.h"
#include "SearchForUsersFrame.h"
#include "SettingsFrame.h"
```

Include dependency graph for MainFrame.cpp:



Classes

- struct DimensionlessFrames::MainFrame::MainFrameThreadHandler

  This struct handles events posted from our mainFrameThread.

Namespaces

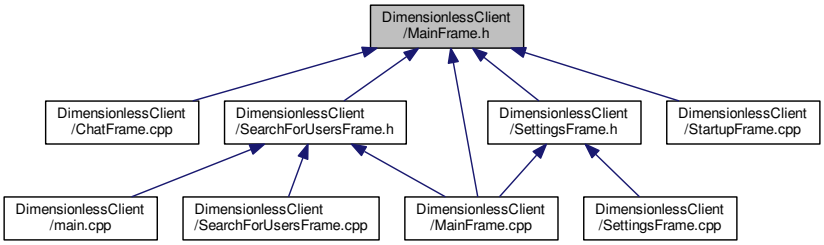
- DimensionlessFrames

13.12 DimensionlessClient/MainFrame.h File Reference

```
#include "ChatFrame.h"
#include "StartupFrame.h"
#include "Dimensionless.h"
#include "DimensionlessFunctionality.h"
```

Include dependency graph for MainFrame.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DimensionlessFrames::ConnectedUserIndex

  This class stores a reference to a ConnectedUser that the list of ConnectedUsers in the MainFrame.
- class DimensionlessFrames::MainFrame

  This class creates a MainFrame that is displayed after a user logs in.

Namespaces

- DimensionlessFrames

Macros

- #define idMainFrame 1000
- #define idContextMenuAcceptConnectionRequest 1001
- #define idContextMenuViewSiteOnline 1002
- #define idContextMenuViewSiteOffline 1003
- #define idContextMenuSendMessage 1004
- #define idContextMenuRejectConnectionRequest 1005

13.12.1 Macro Definition Documentation

13.12.1.1 idContextMenuAcceptConnectionRequest

```
#define idContextMenuAcceptConnectionRequest 1001
```

Definition at line 39 of file MainFrame.h.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick().

13.12.1.2 idContextMenuRejectConnectionRequest

```
#define idContextMenuRejectConnectionRequest 1005
```

Definition at line 43 of file MainFrame.h.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick().

13.12.1.3 idContextMenuSendMessage

```
#define idContextMenuSendMessage 1004
```

Definition at line 42 of file MainFrame.h.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick().

13.12.1.4 idContextMenuViewSiteOffline

```
#define idContextMenuViewSiteOffline 1003
```

Definition at line 41 of file MainFrame.h.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick().

13.12.1.5 idContextMenuViewSiteOnline

```
#define idContextMenuViewSiteOnline 1002
```

Definition at line 40 of file MainFrame.h.

Referenced by DimensionlessFrames::MainFrame::OnTcConnectedUsersContextMenuClick(), and DimensionlessFrames::MainFrame::OnTcConnectedUsersItemRightClick().

13.12.1.6 idMainFrame

```
#define idMainFrame 1000
```

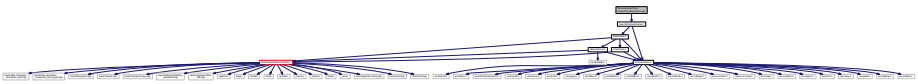
Definition at line 37 of file MainFrame.h.

Referenced by DimensionlessFrames::MainFrame::MainFrameThreadHandler::operator()().

13.13 DimensionlessClient/SearchForUsersFrame.cpp File Reference

```
#include "SearchForUsersFrame.h"
```

Include dependency graph for SearchForUsersFrame.cpp:



Classes

- struct DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler

  This struct handles events posted from our searchForUsersFrameThread.

Namespaces

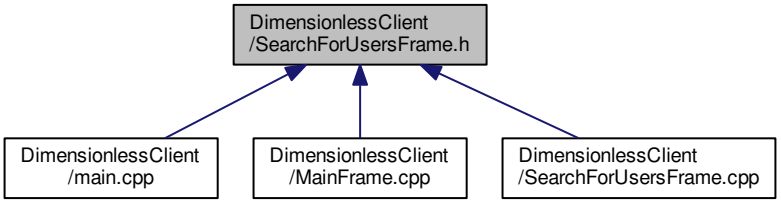
- DimensionlessFrames

13.14 DimensionlessClient/SearchForUsersFrame.h File Reference

```
#include "MainFrame.h"
#include "Dimensionless.h"
```

Include dependency graph for SearchForUsersFrame.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DimensionlessFrames::SearchForUsersFrame

  This class creates a SearchForUsersFrame that is displayed when the currently logged in user wishes to search for other users.

Namespaces

- DimensionlessFrames

Macros

- #define idThreadSearchForUsers 1000
- #define idThreadSearchForUser 1001
- #define idThreadSearchForTorUser 1002
- #define idThreadSendConnectionRequest 1003

13.14.1 Macro Definition Documentation

13.14.1.1 idThreadSearchForTorUser

```
#define idThreadSearchForTorUser 1002
```

Definition at line 37 of file SearchForUsersFrame.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), and DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForTorUser().

13.14.1.2 idThreadSearchForUser

```
#define idThreadSearchForUser 1001
```

Definition at line 36 of file SearchForUsersFrame.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), and DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUser().

13.14.1.3 idThreadSearchForUsers

```
#define idThreadSearchForUsers 1000
```

Definition at line 35 of file SearchForUsersFrame.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), and DimensionlessFrames::SearchForUsersFrame::ThreadEntrySearchForUsers().

13.14.1.4 idThreadSendConnectionRequest

```
#define idThreadSendConnectionRequest 1003
```

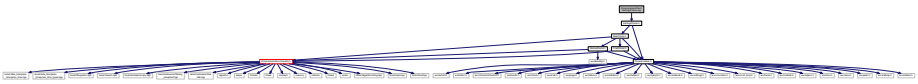
Definition at line 38 of file SearchForUsersFrame.h.

Referenced by DimensionlessFrames::SearchForUsersFrame::SearchForUsersFrameThreadHandler::operator()(), and DimensionlessFrames::SearchForUsersFrame::ThreadEntrySendConnectionRequest().

13.15 DimensionlessClient/SettingsFrame.cpp File Reference

```
#include "SettingsFrame.h"
```

Include dependency graph for SettingsFrame.cpp:



Classes

- struct DimensionlessFrames::SettingsFrame::SettingsFrameThreadHandler

  This struct handles events posted from our settingsFrameThread.

Namespaces

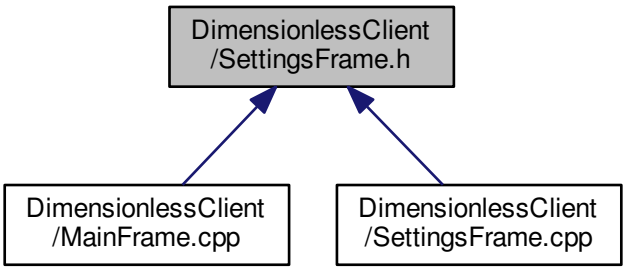
- DimensionlessFrames

13.16 DimensionlessClient/SettingsFrame.h File Reference

```
#include "MainFrame.h"
#include "Dimensionless.h"
```

Include dependency graph for SettingsFrame.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DimensionlessFrames::SettingsFrame

  This class creates a SettingsFrame that is displayed when the currently logged in user wishes to view or modify their settings.

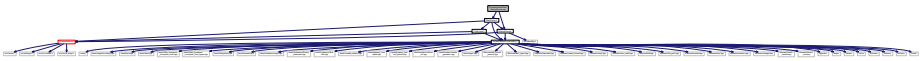
Namespaces

- DimensionlessFrames

13.17 DimensionlessClient/StartupFrame.cpp File Reference

```
#include "StartupFrame.h"
#include "MainFrame.h"
```

Include dependency graph for StartupFrame.cpp:



Classes

- struct DimensionlessFrames::StartupFrame::StartupFrameThreadHandler

  This struct handles events posted from our startupFrameThread.

Namespaces

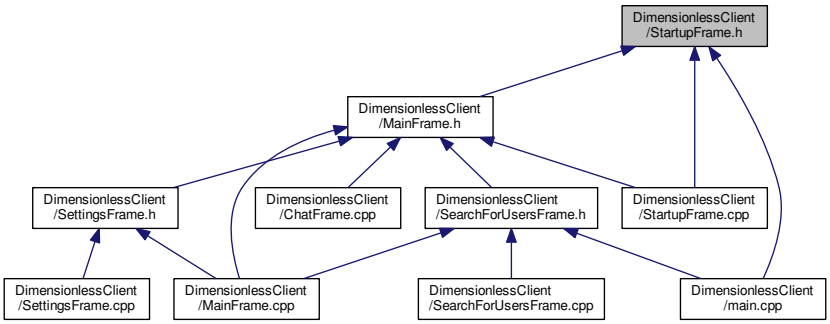
- DimensionlessFrames

13.18 DimensionlessClient/StartupFrame.h File Reference

```
#include "DimensionlessFunctionality.h"
#include "Dimensionless.h"
#include "wx/aboutdlg.h"
```

Include dependency graph for StartupFrame.h:

This graph shows which files directly or indirectly include this file:



Classes

- class DimensionlessFrames::StartupFrame

  This class creates a StartupFrame that is the first frame displayed when the application is started.

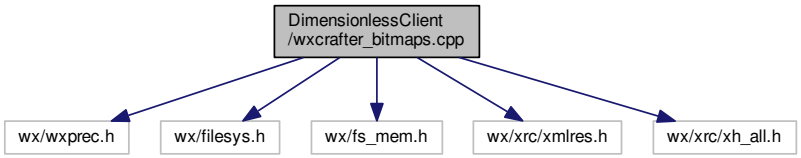
Namespaces

- DimensionlessFrames

13.19 DimensionlessClient/wxcrafter_bitmaps.cpp File Reference

```
#include <wx/wxprec.h>
#include <wx/filesys.h>
#include <wx/fs_mem.h>
#include <wx/xrc/xmlres.h>
#include <wx/xrc/xh_all.h>
```

Include dependency graph for wxcrafter_bitmaps.cpp:



Macros

- #define XRC_ADD_FILE(name, data, size, mime) wxMemoryFSHandler::AddFile(name, data, size)

Functions

- void wxC9ED9InitBitmapResources ()

Variables

- static size_t xml_res_size_0 = 137
- static unsigned char xml_res_file_0 [ ]

13.19.1 Macro Definition Documentation

13.19.1.1 XRC_ADD_FILE

```
#define XRC_ADD_FILE(
              name,
              data,
              size,
              mime ) wxMemoryFSHandler::AddFile(name, data, size)
```

Definition at line 20 of file wxcrafter_bitmaps.cpp.

Referenced by wxC9ED9InitBitmapResources().

13.19.2 Function Documentation

13.19.2.1 wxC9ED9InitBitmapResources()

```
void wxC9ED9InitBitmapResources ( )
```

Definition at line 34 of file wxcrafter_bitmaps.cpp.

References xml_res_file_0, xml_res_size_0, and XRC_ADD_FILE.

```
35 {
36
37     // Check for memory FS. If not present, load the handler:
38     {
39         wxMemoryFSHandler::AddFile(wxT("XRC_resource/dummy_file"), wxT("dummy one"));
40         wxFileSystem fsys;
41         wxFSFile *f = fsys.OpenFile(wxT("memory:XRC_resource/dummy_file"));
42         wxMemoryFSHandler::RemoveFile(wxT("XRC_resource/dummy_file"));
43         if (f) delete f;
44         else wxFileSystem::AddHandler(new wxMemoryFSHandlerBase);
45     }
46
47     XRC_ADD_FILE(wxT("
      XRC_resource/wxcrafter_bitmaps.cpp$_home_ccc_Documents_Dimensionless_DimensionlessClient_wxcrafter_bitmaps.xrc"), xml_re
      xml_res_size_0, wxT("text/xml"));
48     wxXmlResource::Get()->Load(wxT("
      memory:XRC_resource/wxcrafter_bitmaps.cpp$_home_ccc_Documents_Dimensionless_DimensionlessClient_wxcrafter_bitmaps.xrc"))
49 }
```

13.19.3 Variable Documentation

13.19.3.1 xml_res_file_0

```
unsigned char xml_res_file_0[]  [static]
```

Initial value:

```
= {
60,63,120,109,108,32,118,101,114,115,105,111,110,61,34,49,46,48,34,32,101,
110,99,111,100,105,110,103,61,34,85,84,70,45,56,34,63,62,10,60,114,101,
115,111,117,114,99,101,32,120,109,108,110,115,61,34,104,116,116,112,58,
47,47,119,119,119,46,119,120,119,105,100,103,101,116,115,46,111,114,103,
47,119,120,120,114,99,34,62,10,32,32,60,33,45,45,32,72,97,110,100,108,101,
114,32,71,101,110,101,114,97,116,105,111,110,32,105,115,32,79,78,32,45,
45,62,10,60,47,114,101,115,111,117,114,99,101,62,10}
```

Definition at line 25 of file wxcrafter_bitmaps.cpp.

Referenced by wxC9ED9InitBitmapResources().

13.19.3.2 xml_res_size_0

```
size_t xml_res_size_0 = 137  [static]
```

Definition at line 24 of file wxcrafter_bitmaps.cpp.

Referenced by wxC9ED9InitBitmapResources().

13.20 Documentation/Building.md File Reference

13.21 Documentation/Contributing.md File Reference

13.22 Documentation/Installation.md File Reference

13.23 Documentation/Internet Scanning.md File Reference

13.24 Documentation/Main Page.md File Reference

13.25 Documentation/Usage Tutorial.md File Reference