WxFixBoot Developer Documentation

Release 3.0.0

Hamish McIntyre-Bhatty

Apr 30, 2019

CONTENTS:

Note: Most of the docstrings in this project don't follow any particular format at this time, and don't always provide helpful information. This will be remedied in due course.

CHAPTER

ONE

DOCUMENTATION FOR THE MAIN EXECUTABLE FILE (WXFIXBOOT.PY)

This is WxFixBoot's main executable. Execute the file to start the program. This file also contains most of the GUI-related code, with small portions being offloaded into Tools/dialogtools.py and Tools/notebookfunctions.py.

class wxfixboot.WxFixBoot.**BackendThread**(*parent_window*)

This is the backend thread that performs all of the bootloader (and other) operations in the background when the progress window is being shown.

generate_system_report()

Create a system report, containing various information helpful for debugging and fixing problems. It's pretty much like a bootinfo summary.

run()

Do setup, and call self.start_operations()

start_operations()

Start doing operations.

class wxfixboot.WxFixBoot.**BootloaderOptionsWindow**

The bootloader options window.

bind_events()

Bind all events for BootloaderOptionsWindow

create_buttons()

Create the buttons

create_checkboxes()

Create the check boxes

create_choiceboxes()

Create the choice boxes

create_other_widgets()

Create all other widgets

create_text()

Create the text

display_timeoutinfo_message()

Displays an informational message to the user if they only have 1 detected OS.

load_settings(*event=None*)

Load all settings for this OS into the checkboxes and choice boxes

on_advanced_options(*event=None*)

Show/Hide the advanced options, and rotate the arrow

on_backup_bootloader_checkbox(*event=None*)
Enable/Disable the bootloader timeout spinner, based on the value of the timeout checkbox.

on_backup_bootloader_choice(*event=None*)
Allow the user to select a config file to backup the bootloader to

on_basic_options(*event=None*)
Hide/Show the basic options, and rotate the arrow

on_close(*event=None*)
Save settings and GUI state, and then close BootloaderOptionsWindow

on_install_new_bootloader_checkbox(*event=None*)
Enable/Disable options, based on the value of the new bootloader checkbox.

on_kerneloptions_checkbox(*event=None*)
Enable/Disable the kernel options text ctrl, based on the value of the kernel options checkbox.

on_new_bootloader_choice(*event=None*)
Warn user about issues chaging bootloaders if needed

on_oschoice_change(*event=None*, *startup=False*)
Save and load new GUI settings and states in accordance with the OS choice change

on_osinfo(*event=None*)
Hide/Show the OS info, and rotate the arrow

on_restore_bootloader_checkbox(*event=None*)
Enable/Disable options, based on the value of the timeout checkbox.

on_restore_bootloader_choice(*event=None*)
Allow the user to select a config file to restore the bootloader from

on_size(*event=None*)
Auto resize the ListCtrl columns

on_timeout_checkbox(*event=None*)
Enable/Disable the bootloader timeout spinner, based on the value of the timeout checkbox.

on_update_or_reinstall_checkbox(*event=None*)
Enable/Disable options, based on the value of the update/reinstall checkboxes.

save_gui_state(*event=None*, *_os=None*)
Save all the GUI element's states (enabled/disabled) for this _os

save_settings(*event=None*, *_os=None*)
Save all settings for this OS from the checkboxes and choice boxes

set_gui_state(*event=None*)
Set all the GUI element's states (enabled/disabled) for this OS

set_text_labels()
Set text labels for GUI elements

setup_for_restoring_bootloader(*config*)
Setup the window to use the configuration from the chosen bootloader config backup file.

setup_sizers()
Setup the sizers

system_info(*event=None*)
Start SystemInfoWindow

class wxfixboot.WxFixBoot.**GetDiskInformation**(*parent*)
Used to get disk information without blocking the GUI thread. Calls parent.receive_diskinfo when info has ben retrieved.

get_info()
Get disk information as a privileged user

run()
Get Disk Information and return it as a list with embedded lists

class wxfixboot.WxFixBoot.**InitThread**
This is the thread that does all of the heavy lifting and information collection during startup, while the splash screen is displayed.

main_code()
Create the temporary mount point folder and set some default settings.

receive_diskinfo(*info*)
Receive disk info

run()
Handle errors in the main thread code

class wxfixboot.WxFixBoot.**InitialPanel**(*parent*)
This is the panel shown when the GUI is initialising and collecting data.

on_erase_background(*event*)
Redraw the background image when needed

class wxfixboot.WxFixBoot.**InitialWindow**
This is the splash screen that displays on startup with the progress bar and text displaying startup progress.

create_progress_bar_and_text()
Create a progressbar and some progress text

finished_startup(*event=None*)
Starts MainWindow, called when StartupScripts are finished

set_progress_text(*message*)
Update the progress text with the given string

setup_sizers()
Setup sizers for InitialWindow

update_progress_bar(*value*)
Update the progress bar with the given value

update_progress_text(*message*)
Call the text handler thread to distract the user

class wxfixboot.WxFixBoot.**MainWindow**
This is WxFixBoot's main window, where the main options can be selected, and you can navigate to the Bootloader Options window.

bind_events()
Bind all mainwindow events

bootloader_options(*event=None*)
Show the Bootloader Options Window

check_for_updates(*event=None*, *starting_up=False*)
Check for updates using the plist-formatted update file on my website. If some startup, only display info to the user if there was an update. Otherwise (aka requested by user), always display the information.

count_operations()
Count the number of operations to do. Called by self.MainMainOpts()

create_buttons()
Create the buttons

create_checkboxes()
Create the checkboxes

create_menus()
Create the menus

create_text()
Create the text

make_status_bar()
Create the status bar

on_about(*event=None*)
Shows the About Box

on_checkbox(*event=None*)
Called when one of the checkboxes is checked/unchecked, to make sure the options stay valid

on_exit(*event=None*)
Shut down.

progress_window(*event=None*)
Starts Progress Window

refresh_main_window()
Refresh the main window to reflect changes in the options, or after a restart.

save_main_options()
Save all options

setup_sizers()
Setup sizers for MainWindow

show_privacypolicy(*event=None*)
Show PrivPolWindow

system_info(*event=None*)
Start SystemInfoWindow

update_output_box(*line*, *show_output*)
Ignored, temporarily here for compatibility with coretools.start_process

class wxfixboot.WxFixBoot.**PrivPolWindow**
The privacy policy window where all of WxFixBoot's privacy information may be found.

bind_events()
Bind events so we can close this window.

create_button()
Create the close buton.

load_page()
Load the privacy policy web page (locally stored)

on_close(*event=None*)
Close PrivPolWindow

setup_sizers()
Set up sizers for PrivPolWindow

class wxfixboot.WxFixBoot.**ProgressTextHandlerThread**
A separate thread to handle the text on the splash screen to indicate the program is still running during slow startup routines.

run()
Distract the user with some text effects on the Initial Window. For 10 seconds of the same message, make dots build up to 3 dots and back after the text. After 10 seconds of same message, state that WxFixBoot is still starting up, and to be patient.

class wxfixboot.WxFixBoot.**ProgressWindow**
This is the window that displays with the two progress bars during bootloader (and other) operations.

add_line_to_output_box(*line*, *_type*)
Adds the given line to the outtput box, also handles carriage returns and backspaces

backend_thread_finished()
Called when the BackendThread is finished, enables self.restart_button and self.exit_button

backspace()
Handles backspaces in output

bind_events()
Bind events for Progress Window

carriagereturn()
Handles carriage returns in output

create_buttons()
Create buttons.

create_progressbars()
Create both progress bars

create_text()
Create the Text

focus_on_output_button(*event=None*)
Focus on the show output button instead of the TextCtrl, and reset the insertion point back after 30 milliseconds, preventing the user from changing the insertion point and messing the formatting up.

on_exit(*event=None*)
Exits the programs, and sorts out log file saving/deleting stuff

restart_wxfixboot(*event=None*)
Restart WxFixBoot

session_ending(*event*)
Attempt to veto e.g. a shutdown/logout event if recovering data.

setup_sizers()
Setup sizers for Progress Window

show_output(*event=None*)
Show and Hide the output box in ProgressWindow()

update_current_operation_text(*message*)
Keep the current operations status text up to date.

update_current_progress(*msg*)
Update the progress of the current progress progress bar

update_output_box(*line*, *show_output=True*)
Update the output box, and add lines to the list

update_total_progress()
Update the progress of the overall progress progress bar

class wxfixboot.WxFixBoot.**SystemInfoPage1**(*notebook*, *systeminfo_window*)
The first notebook page of the System Information Window.

on_size(*event=None*)
Auto resize the ListCtrl columns

class wxfixboot.WxFixBoot.**SystemInfoPage2**(*notebook*, *systeminfo_window*)
The first notebook page of the System Information Window.

on_size(*event=None*)
Auto resize the ListCtrl columns

class wxfixboot.WxFixBoot.**SystemInfoPage3**(*notebook*, *systeminfo_window*)
The first notebook page of the System Information Window.

on_size(*event=None*)
Auto resize the ListCtrl columns

class wxfixboot.WxFixBoot.**SystemInfoPage4**(*notebook*, *systeminfo_window*)
The first notebook page of the System Information Window.

on_size(*event=None*)
Auto resize the ListCtrl columns

class wxfixboot.WxFixBoot.**SystemInfoPage5**(*notebook*, *systeminfo_window*)
The first notebook page of the System Information Window.

on_size(*event=None*)
Auto resize the ListCtrl columns

class wxfixboot.WxFixBoot.**SystemInfoPage6**(*notebook*, *systeminfo_window*)
The first notebook page of the System Information Window.

on_size(*event=None*)
Auto resize the ListCtrl columns

class wxfixboot.WxFixBoot.**SystemInfoWindow**
The System Information Window, complete with all of the notebook pages that hold the information.

bind_events()
Bind all events for SystemInfoWindow

on_exit(*event=None*)
Exit SystemInfoWindow

class wxfixboot.WxFixBoot.**WxFixBoot**(*redirect=False*, *filename=None*, *useBestVisual=False*, *clearSigInt=True*)
This class is used to start the GUI.

OnInit()
Starts InitialWindow()

wxfixboot.WxFixBoot.**usage**()
Prints usage information to the command line

CHAPTER

TWO

DOCUMENTATION FOR THE UNIT TESTS EXECUTABLE (TESTS.PY)

This module contains the code used to run the test suites for WxFixBoot.

CHAPTER

THREE

DOCUMENTATION FOR THE UNIT TESTS PACKAGE

CHAPTER

FOUR

DOCUMENTATION FOR THE TOOLS PACKAGE

CHAPTER

FIVE

DOCUMENTATION FOR THE CORE TOOLS IN THE TOOLS PACKAGE

This module contains the "core tools" used in various parts of WxFixBoot.

wxfixboot.Tools.coretools.**any_mounted**(*partitions*)
Checks if any of the given partitions are mounted. partitions are the given partitions to check.

Return boolean True/False.

wxfixboot.Tools.coretools.**emergency_exit**(*message*)
Handle emergency exits. Warn the user, log, and exit to terminal with the given message

wxfixboot.Tools.coretools.**get_helper**(*cmd*)
Figure out which helper script to use.

wxfixboot.Tools.coretools.**get_mount_point_of**(*partition*)
Returns the mountpoint of the given partition, if any. Otherwise, return None

wxfixboot.Tools.coretools.**get_partition_mounted_at**(*mount_point*)
Returns the partition mounted at the given mountpoint, if any. Otherwise, return None

wxfixboot.Tools.coretools.**is_mounted**(*partition*, *mount_point=None*)
Checks if the given partition is mounted. partition is the given partition to check. If mount_point is specified, check if the partition is mounted there, rather than just if it's mounted.

Return boolean True/False.

wxfixboot.Tools.coretools.**mount_partition**(*partition*, *mount_point*, *options=''*)
Mounts the given partition. partition is the partition to mount. mount_point is where you want to mount the partition. options is non-mandatory and contains whatever options you want to pass to the mount command. The default value for options is an empty string.

wxfixboot.Tools.coretools.**read**(*cmd*, *testing=False*)
Read the cmd's output char by char, but do as little processing as possible to improve startup performance

wxfixboot.Tools.coretools.**read_and_send_output**(*cmd*, *show_output*)
Read the cmd's output char by char, and send the output to the output box

wxfixboot.Tools.coretools.**read_privileged_file**(*filename*)
Uses start_process() and a helper script to read privileged files and returns the content as a string.

wxfixboot.Tools.coretools.**remount_partition**(*partition*, *mode='rw'*)
Remounts the given partition. partition is the partition to remount. mode is non-mandatory and is either rw or ro for read-write or read-only respectively. The default value for mode is rw.

wxfixboot.Tools.coretools.**send_notification**(*msg*)
Send a notification, created to reduce clutter in the rest of the code.

wxfixboot.Tools.coretools.**setup_chroot**(*mount_point*)
Set up a chroot for the given mountpoint.

wxfixboot.Tools.coretools.**start_process**(*exec_cmds*, *show_output=True*, *return_output=False*, *testing=False*, *privileged=False*)
Start a process given a string of commands to execute. show_output is boolean and specifies whether to show output in the outputbox (if exists) or not.

return_output is boolean and specifies whether to return the output back to the caller or not.

wxfixboot.Tools.coretools.**teardown_chroot**(*mount_point*)
Remove a chroot at the given mountpoint.

wxfixboot.Tools.coretools.**unmount**(*mount_point*)
Unmounts the given mountpoint. mount_point is the mountpoint to unmount. mount_point can also be a partition name (for example /dev/sda1).

wxfixboot.Tools.coretools.**update_chroot_mtab**(*mount_point*)
Update /etc/mtab inside a chroot, so the list of mounted filesystems is always right.

wxfixboot.Tools.coretools.**write_privileged_file**(*filename*, *file_contents*)
Uses start_process() and a helper script to write privileged files. This is an inherent security risk, but is needed to write bootloader configuration. So, there is a hardcoded whitelist of files we're allowed to write to here to mitigate at least some of that risk.

This isn't an exact match thing, because these files might be within mountpoints. We will nevertheless match as exactly as we can.

A few different checks are done to ensure that we are allowed to modify this file, just in case the program does something weird.

The helper script is run through polkit, for security.

CHAPTER

SIX

DOCUMENTATION FOR THE DIALOG TOOLS IN THE TOOLS PACKAGE

This module provides the standard dialogs and related control functions used by WxFixBoot.

This provides the extra "show_thread..." functions to enable calling dialogs from background threads - for example the thread that is used for backend operations when modifying the bootloader.

wxfixboot.Tools.dialogtools.**is_gui_thread**()
Used to determine if the current thread is the GUI thread. Performing this check enables us to use theese convenience functions in the GUI thread as well.

Args: None

Returns: boolean - Whether or not this is the GUI thread.

wxfixboot.Tools.dialogtools.**show_choice_dlg**(*message*, *title*, *choices*, *allow_cancel=False*)
Handle showing thread choice dialogs, reducing code duplication and complications and errors. It can be used like this:

DialogTools().show_choice_dlg(message=<message>, title=<title>, choices=<choices>)

message is whatever you want the dialog to say. title sets the title bar text on the dialog. choices is a list of choices you want the dialog to display.

wxfixboot.Tools.dialogtools.**show_msg_dlg**(*message*, *kind='info'*)
Handle showing thread message dialogs, reducing code duplication and complications and errors. It can be used like this: DialogTools().show_msg_dlg(kind=<kind>, message=<message>). kind is one of 'info', 'warning' or 'error'. message is whatever you want the dialog to say.

wxfixboot.Tools.dialogtools.**show_save_file_dlg**(*title='WxFixBoot - Select A File'*, *wildcard='All Files/Devices (*)|*'*)
Handle showing thread file dialogs, reducing code duplication and compilications and errors. It can be used like this: ShowFileDlg(title=<title>, wildcard=<wildcard>) message is whatever you want the dialog to say. wildcard is a | seperated list of file types to show, including their names as visible to the user.

wxfixboot.Tools.dialogtools.**show_text_entry_dlg**(*message*, *title='WxFixBoot - Text Entry'*)
Handle showing thread text entry dialogs, reducing code duplication, compilications and errors. It can be used like this: DialogTools().show_text_entry_dlg(message=<message>, title=<title>) message is whatever you want the dialog to say. title sets the title bar text on the dialog.

wxfixboot.Tools.dialogtools.**show_thread_choice_dlg**(*msg*, *choices*, *title='WxFixBoot - Select an Option'*)
Shows a choice dialog from a thread upon instruction

wxfixboot.Tools.dialogtools.**show_thread_msg_dlg**(*msg*, *kind='info'*)
Shows a message dialog from a thread upon instruction

wxfixboot.Tools.dialogtools.**show_thread_save_file_dlg**(*title='WxFixBoot - Select A File'*, *wildcard='All Files/Devices (*)|*'*)

Shows a save file choice dialog from a thread upon instruction

wxfixboot.Tools.dialogtools.**show_thread_text_entry_dlg**(*msg*, *title='WxFixBoot - Text Entry'*)

Shows a text entry dialog from a thread upon instruction

wxfixboot.Tools.dialogtools.**show_thread_yes_no_dlg**(*msg*, *title='WxFixBoot - Question'*, *buttons=(None, None)*)

Shows a yes/no dialog from a thread upon instruction

wxfixboot.Tools.dialogtools.**show_yes_no_dlg**(*message*, *title='WxFixBoot - Question'*, *buttons=(None, None)*)

Handle showing thread yes/no dialogs, reducing code duplication and compilications and errors. It can be used like this: DialogTools().show_yes_no_dlg(message=<message>, title=<title>) message is whatever you want the dialog to say. buttons is the text for the button labels. title sets the title bar text on the dialog.

CHAPTER

SEVEN

DOCUMENTATION FOR THE DICTIONARY SETUP MODULE IN THE TOOLS PACKAGE

This module holds the shared dictionaries used by the rest of the program. All you need to do to gain access to the shared dictionaries is import this module.

CHAPTER

EIGHT

DOCUMENTATION FOR THE NOTEBOOK TOOLS IN THE TOOLS PACKAGE

This module contains functions used for generating the system info notebook.

wxfixboot.Tools.notebookfunctions.**bind_events**(*self*)
Bind all events for the caller

wxfixboot.Tools.notebookfunctions.**create_widgets**(*self*)
Create all widgets for the caller

wxfixboot.Tools.notebookfunctions.**setup_sizers**(*self*)
Set up the sizers for the caller

wxfixboot.Tools.notebookfunctions.**update_list_ctrl**(*self*, *event=None*, *headings=None*, *dictionary=None*)
Update the list control

CHAPTER

NINE

DOCUMENTATION FOR THE STARTUP TOOLS PACKAGE IN THE THE TOOLS PACKAGE

CHAPTER

TEN

DOCUMENTATION FOR THE CORE STARTUP TOOLS IN THE THE TOOLS PACKAGE

This module contains the core functions used during WxFixBoot's startup procedure.

wxfixboot.Tools.StartupTools.core.**ask_for_os_name**(*partition*, *is_current_os*)
Ask the user if an OS exists on the given partition.

wxfixboot.Tools.StartupTools.core.**determine_os_architecture**(*mount_point*)
Look for OS architecture on given partition.

wxfixboot.Tools.StartupTools.core.**determine_package_manager**(*apt_cmd*, *yum_cmd*)
Determine and return the package manager using the given command strings.

wxfixboot.Tools.StartupTools.core.**get_defaultoss_partition**(*the_os*)
Get the partition for the given OS's default OS to boot

wxfixboot.Tools.StartupTools.core.**get_fstab_info**(*mount_point*, *os_name*)
Get /etc/fstab info and related info (EFI Partition, /boot partition) for the given OS at the given mountpoint.

wxfixboot.Tools.StartupTools.core.**get_os_name_with_lsb**(*partition*, *mount_point*, *is_current_os*)
Attempt to get an OS's name using lsb_release -sd as a fallback.

wxfixboot.Tools.StartupTools.core.**has_windows_10**(*mount_point*)
Try to find a Windows 10 installation. Return True if found, False if not.

wxfixboot.Tools.StartupTools.core.**has_windows_7**(*mount_point*)
Try to find a Windows 7 installation. Return True if found, False if not.

wxfixboot.Tools.StartupTools.core.**has_windows_8**(*mount_point*)
Try to find a Windows 8/8.1 installation. Return True if found, False if not.

wxfixboot.Tools.StartupTools.core.**has_windows_9x**(*mount_point*)
Try to find a Windows 9X installation. Return True if found, False if not.

wxfixboot.Tools.StartupTools.core.**has_windows_vista**(*mount_point*)
Try to find a Windows Vista installation. Return True if found, False if not.

wxfixboot.Tools.StartupTools.core.**has_windows_xp**(*mount_point*)
Try to find a Windows XP installation. Return True if found, False if not.

wxfixboot.Tools.StartupTools.core.**look_for_bootloaders_on_partition**(*the_os*, *package_manager*, *mount_point*, *using_chroot*)
Look for bootloaders installed in the OS in the given mount point.

wxfixboot.Tools.StartupTools.core.**make_bootloaderinfo_entry_for_macos**(*the_os*)
Makes an entry in BOOTLOADER_INFO for macOS

wxfixboot.Tools.StartupTools.core.**make_bootloaderinfo_entry_for_windows**(*the_os*)
Makes an entry in BOOTLOADER_INFO for Windows

wxfixboot.Tools.StartupTools.core.**match_partition_to_os**(*the_os*)
Matches the default boot device (in a menu entry) to an OS in OS_INFO

CHAPTER

ELEVEN

DOCUMENTATION FOR THE BOOTLOADER CONFIGURATION OBTAINING STARTUP TOOLS IN THE THE TOOLS PACKAGE

This module contains tools used to get bootloader information during WxFixBoot's startup procedures.

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**assemble_grub2_menu_entry**(*menu_entries*, *menu_ids*, *menu_entries_f* *menu*, *line*, *en-* *try_counter*)

Assemble a menu entry in the dictionary for GRUB2 (BIOS and UEFI)

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**assemble_grublegacy_menu_entry**(*menu_e* *menu_e* *line*, *en-* *try_cou*

Assemble a menu entry in the dictionary for GRUB LEGACY

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**assemble_lilo_menu_entry**(*menu_entries*, *menu_entries_fil* *line*, *en-* *try_counter*)

Assemble a menu entry in the dictionary for LILO/ELILO

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**find_grub**(*os_partition*, *grub_version*)

Find GRUB for the given OS.

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**get_grub2_config**(*config_file_path*, *grubenv_file_path*, *menu_entries*)

Get important bits of config from grub2 (MBR or UEFI)

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**get_grublegacy_config**(*config_file_path*, *menu_entries*)

Get important bits of config from grub-legacy

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**get_lilo_config**(*config_file_path*, *_os*)

Get important bits of config from lilo and elilo

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**parse_grub2_menu_data**(*menu_data=''*, *mount_point=''*, *menu_entries=None*, *menu_name='MainM*, *menu_ids=None*, *menu_id=''*)

Find and parse GRUB2 (EFI and BIOS) menu entries in the given line list

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**parse_grublegacy_menu_entries**(*menu_ent*

Find and parse GRUB LEGACY menu entries.

wxfixboot.Tools.StartupTools.getbootloaderconfigtools.**parse_lilo_menu_entries**(*menu_entries_file_*

Find and parse LILO and ELILO menu entries.

CHAPTER

TWELVE

DOCUMENTATION FOR THE MAIN STARTUP TOOLS IN THE THE TOOLS PACKAGE

This module contains the main functions that form most of the cohesive whole of WxFixBoot's startup procedures.

wxfixboot.Tools.StartupTools.main.**check_depends**()
: Check dependencies, and show an error message and kill the app if the dependencies are not met.

wxfixboot.Tools.StartupTools.main.**check_filesystems**()
: Check all unmounted filesystems.

wxfixboot.Tools.StartupTools.main.**check_for_live_disk**()
: Try to determine if we're running on a live disk.

wxfixboot.Tools.StartupTools.main.**final_check**()
: Check for any conflicting options, and warn the user of any potential pitfalls.

wxfixboot.Tools.StartupTools.main.**get_bootloaders**()
: Find all bootloaders (for each OS), and gather some information about them

wxfixboot.Tools.StartupTools.main.**get_firmware_type**()
: Get the firmware type

wxfixboot.Tools.StartupTools.main.**get_oss**()
: Get the names of all OSs on the HDDs.

wxfixboot.Tools.StartupTools.main.**mount_core_filesystems**()
: Mount all core filsystems defined in the /etc/fstab of the current operating system.

CHAPTER

THIRTEEN

DOCUMENTATION FOR THE BACKEND TOOLS PACKAGE IN THE THE TOOLS PACKAGE

CHAPTER

FOURTEEN

DOCUMENTATION FOR THE ESSENTIAL BACKEND TOOLS IN THE THE TOOLS PACKAGE

This contains the essential backend tools operations. (internet connection test, file system checks).

These are called the essential tools because a failure here may cancel all other operations. For example, if a filesystem check fails, it's a bad idea to write to the disk by doing bootloader operations. Likewise, a new bootloader cannot be installed without an internet connection.

wxfixboot.Tools.BackendTools.essentials.**check_internet_connection**()
Check the internet connection.

wxfixboot.Tools.BackendTools.essentials.**filesystem_check**(*_type*, *manage_bootloader_function*)
Quickly check all filesystems.

wxfixboot.Tools.BackendTools.essentials.**handle_filesystem_check_return_values**(*exec_cmds*, *retval*, *partition*, *manage_bootloader_fu*
Handle Filesystem Checker return codes.

CHAPTER

FIFTEEN

DOCUMENTATION FOR THE HELPER BACKEND TOOLS IN THE THE TOOLS PACKAGE

This module contains the helper backend tools (functions) used when performing more complex operations with WxFixBoot. These are included here to reduce code duplication and attempt to keep everything more organised.

wxfixboot.Tools.BackendTools.helpers.**backup_uefi_files**(*mount_point*)
Backup some .efi files, just in case something goes wrong.

wxfixboot.Tools.BackendTools.helpers.**find_checkable_file_systems**()
Find all checkable filesystems, and then return them to EssentialBackendTools().filesystem_check()

wxfixboot.Tools.BackendTools.helpers.**find_missing_fsck_modules**()
Check for and return all missing fsck modules (fsck.vfat, fsck.minix, etc).

wxfixboot.Tools.BackendTools.helpers.**manage_uefi_files**(*_os*, *mount_point*)
Manage UEFI bootloader files.

wxfixboot.Tools.BackendTools.helpers.**partition_matches_os**(*partition*, *_os*)
Matches the given boot device to an OS, using the info we gathered at startup using the function above

wxfixboot.Tools.BackendTools.helpers.**wait_until_packagemanager_free**(*mount_point*, *package_manager*)
Check if the package manager is in use, and if so, wait until it is no longer in use.

wxfixboot.Tools.BackendTools.helpers.**write_fstab_entry_for_uefi_partition**(*_os*, *mount_point*)
Write an /etc/fstab entry for the UEFI System Partition, if there isn't already one. DISABLED***

CHAPTER

SIXTEEN

DOCUMENTATION FOR THE MAIN BACKEND TOOLS IN THE THE TOOLS PACKAGE

This module contains the functions for the main operations WxFixBoot performs: updating, removing, and installing/reinstalling bootloaders. It also includes high-level functions for setting bootloader configuration.

wxfixboot.Tools.BackendTools.main.**install_new_bootloader**(*_os*)
Install a new bootloader.

wxfixboot.Tools.BackendTools.main.**manage_bootloader**(*_os*)
Manage the installation and removal of each bootloader.

wxfixboot.Tools.BackendTools.main.**remove_old_bootloader**(*_os*)
Remove the currently installed bootloader.

wxfixboot.Tools.BackendTools.main.**set_new_bootloader_config**(*_os*)
Manage setting new bootloader config.

CHAPTER

SEVENTEEN

DOCUMENTATION FOR THE BOOTLOADER TOOLS PACKAGE IN THE BACKEND TOOLS PACKAGE IN THE THE TOOLS PACKAGE

CHAPTER

EIGHTEEN

DOCUMENTATION FOR THE BOOTLOADER CONFIG SETTING TOOLS IN THE BACKEND TOOLS PACKAGE IN THE THE TOOLS PACKAGE

This module contains the tools used to set the configuration of bootloaders when performing operations with WxFixBoot.

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**assemble_lilo_menu_entry**(*os_nam*
_os,
ker-
nel_opti
new_file

Create a LILO menu entry in the config file, and return it

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**install_elilo_to_partition**(*_os*,
pack-
*age_*
use_c
moun

Install ELILO to the EFI/UEFI Partition

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**install_grub2_to_efi_partition**

Install GRUB2 (EFI/UEFI version) into the EFI/UEFI partition

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**install_grub2_to_mbr**(*package_mana*
use_chroot,
mount_point,
de-
vice)

Install GRUB2 (BIOS version) into the MBR of the hard drive

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**install_lilo_to_mbr**(*use_chroot*,
mount_point)

Install LILO into the MBR.

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**make_lilo_os_entries**(*_os*,
file-
toopen,
mount_point,
ker-
nel_options)

Make OS Entries in the bootloader menu for LILO and ELILO, and then the default OS

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**set_grub2_config**(*_os*, *file-toopen*, *boot-loader_timeout*, *ker-nel_options*)

Set GRUB2 config.

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**set_lilo_config**(*_os*, *file-toopen*)

Set config for both LILO and ELILO

wxfixboot.Tools.BackendTools.BootloaderTools.setconfigtools.**update_grub2**(*_os*, *pack-age_manager*, *use_chroot*, *mount_point*)

Run 'update-grub' to update GRUB2's (BIOS and EFI/UEFI) configuration and bootloader menu

CHAPTER

NINETEEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

D

E

F

G

H

I

L

M

O

P

R

S

T

U

W