QtPass

1.4.0

Generated by Doxygen 1.9.6

Chapter 1

QtPass

1.1 Introduction

QtPass is a multi-platform GUI for pass, the standard unix password manager.

https://qtpass.org/

1.2 Installation

1.2.1 Dependencies

- QtPass requires Qt 5.2 or later.
- The Linguist package is required to compile the translations.
- For use of the fallback icons the SVG library is required.

At runtime the only real dependency is gpg2 but to make the most of it, you'll need git and pass too.

1.2.2 From source

On most ∗nix systems all you need is:

```
qmake && make && make install
```

Chapter 2

Changelog

Unreleased

Full Changelog

Fixed bugs:

- qtpass do not strip comments of gpg_id file #625
- Build Windows #598
- Please sign release 1.4.0-rc1 #594
- Create a new release 1.4.0 #567
- Crashes during free text search #147
- Program does not run in Windows 10 #123

Closed issues:

- Confusion about pass ecosystem #657
- No context menu #522

Merged pull requests:

- Fix build with Qt6. #656 (Vascom)

v1.4.0 (2023-09-17)

Full Changelog

Fixed bugs:

- Update website to reflect new brew command syntax #601
- Qtpass - not asking for password #585
- Missing menu #574

Merged pull requests:

- Initial Korean from Weblate #655 (annejan)
- Natural language fixes #654 (annejan)
- Translations update from Hosted Weblate #650 (weblate)
- Added Serbian and Estonian to project file #649 (annejan)
- Translations update from Hosted Weblate #648 (weblate)
- Translations update from Hosted Weblate #647 (weblate)
- Version bump and cleanup #646 (annejan)

1.4.0-rc2 (2023-08-31)

Full Changelog

Fixed bugs:

- OTP function stopped working #630
- Cannot decrypt own passwords; No secret key #580
- gpg not found on macOS #575
- Installation is failed using latest Homebrew in macOS #564
- Deleting a directory sometimes deletes the entire password store including git repositories #556

Closed issues:

- [Windows] Git repository not working #638
- support PASSWORD_STORE_SIGNING_KEY with profiles #624
- Support multiple branches via "Profiles" feature #545

Merged pull requests:

- clang-format -i src/∗.cpp src/∗.h #645 (annejan)
- Translations update from Hosted Weblate #644 (weblate)
- Fix taborder and add buddies in keygen dialog #643 (svuorela)
- Restore licensing info for QProgressIndicator #642 (svuorela)
- Clazy cleanup and other minor fixes #641 (svuorela)
- fix the unintended "running" of the entropy window in the keygen dial... #640 (lherschi)
- Translations update from Hosted Weblate #636 (weblate)
- Add pass store signing key feature #634 (timegrid)
- Translations update from Hosted Weblate #633 (weblate)
- Translations update from Hosted Weblate #632 (weblate)
- Translations update from Hosted Weblate #629 (weblate)
- Translations update from Hosted Weblate #628 (weblate)
- Translations update from Hosted Weblate #627 (weblate)
- Translations update from Hosted Weblate #626 (weblate)
- Translations update from Hosted Weblate #622 (weblate)
- markdownlint –fix && textlint –fix #621 (annejan)
- Document "Using profiles" #619 (buepro)
- Translations update from Hosted Weblate #618 (weblate)
- Translations update from Hosted Weblate #617 (weblate)
- super-linter ENV variables in shared location for local and automated #616 (annejan)
- fix bug => clipboard was not cleared when using primary selection #615 (pythcoiner)
- Translations update from Hosted Weblate #614 (weblate)
- Translations update from Hosted Weblate #613 (weblate)
- Removed travis (no longer free) and lgtm (migrated to Github) #612 (annejan)
- Translations update from Hosted Weblate #611 (weblate)
- Super Linter added and fixing findings #610 (annejan)
- New Transifex integration yml #609 (annejan)
- Install QT in codeql workflow #608 (annejan)
- Translations update from Hosted Weblate #607 (weblate)
- Translation cleanup #606 (annejan)
- translations updated #605 (annejan)
- Fix accidental deletion of entire passwordstore #604 (FSMaxB)
- Add more options for the password displaying #587 (l3u)
- Delete context menu after exec #578 (fasked)
- Translations update from Hosted Weblate #576 (weblate)

1.4.0-rc1 (2021-09-22)

Full Changelog

Implemented enhancements:

- Set correct WM_CLASS for the qr-code popup #506

Fixed bugs:

- QtPass does not detect current $GNUPGHOME and causes it to fail decryption #569
- <tt> ... </tt> included in password text #542
- Markup tags are left in password and clipboard #533
- Renaming passwords and directories fail #487
- Will not run on Windows 10 1903 b18362.418 #486

Closed issues:

- Hide results on search #551
- QtPass 1.3.2 freezes on macOS 10.15.6 when trying to display password #544
- Icons are blurry when fractional scaling is enabled #525
- [Request] clear search password when change profile #524
- Copying not possible on Ubuntu 20.04 #521
- UI can't handle passwords with periods in their name #520
- Display passwords in mono space font #514
- QtPass 1.3.2 for Ubuntu 19.10 (eoan) #512
- Default password visibility #511
- Consider mentioning export abilities in migration docs, if any are present #505
- Enable out-of-source (shadow) builds. #501
- password visiblity can't be fully hidden #496
- Translations need updating and checking #488
- Frontend doesn't work well with HiDPI screen #464
- How to let QtPass use the real "pass" on windows #458
- Fresh install of Antergos with Deepin - High DPI scaling is not working #417
- Strange behavior when clearing filter #402
- Tray icon remains after quitting program #401
- QtPass doesn't work will pass in WSL #375
- UI is blurry on HiDPI screens on macOS (retina) since 1.2.x #355
- No prompt for passphrase for git key on windows. #317

- Config dialog's Password Generation field got crowded between 1.1.3 and 1.1.6 #278

Merged pull requests:

- Translations update from Weblate #573 (weblate)
- Fix keys created/expires dates in the users dialog window (fix: 571) #572 (nfetisov)
- Correct a typo in pass.cpp #570 (felixonmars)
- Fix installation instructions in README.md #565 (kawarimidoll)
- Translations update from Weblate #563 (weblate)
- Translations update from Weblate #562 (weblate)
- Translations update from Weblate #560 (weblate)
- Keep suffices when moving (to) a directory while imitiating pass #559 (ichthyosaurus)
- Explicitly only remove ".gpg" when renaming files #558 (ichthyosaurus)
- Translations update from Weblate #554 (weblate)
- Translations update from Weblate #553 (weblate)
- Translations update from Weblate #552 (weblate)
- Translations update from Weblate #548 (weblate)
- Move MainWindow to the screen the cursor is on #547 (inhinias)
- Translations update from Weblate #541 (weblate)
- Translations update from Weblate #535 (weblate)
- Fix issues with renaming passwords and moving folders #532 (ChaoticEnigma)
- Translations update from Weblate #531 (weblate)
- Translations update from Weblate #530 (weblate)
- Clear search on profile change #529 (cmol)
- #514 Show password with a monospace font #528 (cmol)
- Update minimum Qt version #527 (cmol)
- Fix blurry icons when fractional scaling is enabled #526 (mthw0)
- Spelling: Git pull, Git push #516 (comradekingu)
- Enable ubuntu, windows and macOS based builds for CI #508 (boppybibbles)
- Enable out-of-source build #503 (boppybibbles)
- Use new stable version of `install-qt-action`. #502 (boppybibbles)
- Don't base pass-otp availability decision on hardcoded /usr/lib #499 (nh2)
- Spelling: Search for users, , #495 (comradekingu)
- Spelling: Keylist missing, Could not fetch, GPG #493 (comradekingu)
- Spelling: Git, GPG, PWGen, etc. #492 (comradekingu)
- Don't use a deprecated method #491 (amarsman)
- Issue #402: 'deselect()' on clearing filter #490 (petr-nehez)

v1.3.2 (2019-10-09)

Full Changelog

Fixed bugs:

- QtPass could not run on Windows7 thin #485
- Segfault on application startup (macOS) #481
- Application crashes on empty password store #466
- App is completely broken #423

Closed issues:

- Edit window on Gnome has no padding around #484
- Buttons width on RHEL 8 #483
- 'Start minimized' no longer works #471
- Editor doesn't wait for PGP key to decrypt #470
- v1.3.0 Data Not Showing #465
- Hangs on macOS after Security Update 2019-003 10.12.6 #461
- No public key #308

Merged pull requests:

- Don't call QtPass::setup() from QtPass class constructor (should fix #466) #482 (maciejsszmigiero)

v1.3.1 (2019-10-01)

Full Changelog

Implemented enhancements:

- Renaming password #463
- [Feature Request] Edit main title field #446

Fixed bugs:

- build: dependency issue #467
- is running but no gui #451

Closed issues:

- Additional lines (notes) are not shown #474
- Bundle ID is literally $(PRODUCT_BUNDLE_IDENTIFIER) #448

Merged pull requests:

- Add license scan report and status #480 (fossabot)
- Build tooling related fixes #479 (maciejsszmigiero)
- Add missing overrides #478 (amarsman)
- Main window entry details improvements #477 (maciejsszmigiero)
- Fix HTML links color and NL translation building error #476 (a-andreyev)
- Restore directories-first order of passwords tree view on non-Mac platforms #475 (maciejsszmigiero)
- Add missing finishedShow() signal connection in PasswordDialog constructor (fixes the "Edit password" function) #473 (maciejsszmigiero)
- Sorted profiles dropdown as in #404 #472 (Noettore)
- Add support for passwords and directories renaming as requested in #463 #469 (Noettore)
- Fix missing app ID and icon on Wayland. #468 (lightbulbjim)

v1.3.0 (2019-08-20)

Full Changelog

Implemented enhancements:

- Localization makes commits absolutely unreadable #405
- Add otp (two factor authentication) support #327
- Open specific entry from command-line parameter #32

Fixed bugs:

- Windows sigsev issues #326
- Access to the / (root) directory form within the application window on macOS #302
- PRNG seeding is done totally wrong #238
- Context menu on transparent fields is transparent too . . #227

Closed issues:

- various issues with Info.plist file on macOS #457
- Can not add new passwords for some reason #454
- GnuPG not found on Linux Mint #433

- How to clean up the app #429
- LAN sync request #427
- Profiles can not be removed #415
- Compilation error in (K)ubuntu 16.04.5 with sources tar.gz from version 1.2.3 #408
- Prevent from removing whole password-store directory and hidden directories and files #400
- Version information string/s #398
- We should select a C++ std too #372
- We should select a minimum Qt version #371
- Problem with GNUpg not found on macOS #362
- Compiling for Linux Mint 18 Ubuntu 16 #357
- make qtpass portable in windows #356
- Unable to see main application window (applicationn runs minimized to tray only) #286
- Startup variables and parameters #212
- [macOS] Password input dialog suddenly stopped popping up #191
- MainWindow is a giant monolithic mess #107

Merged pull requests:

- Use key fingerprint as ID instead of “long” ID. #452 (Natureshadow)
- Typo: dialouge to dialogue. #444 (georgjaehnig)
- Scripts and logic specific to Windows Store releases #439 (rdoeffinger)
- For config check, check that the selected binary is available. #438 (rdoeffinger)
- Fix character encoding issues for non-UTF-8 locales. #435 (rdoeffinger)
- Fixes and improvments for config dialog #432 (rdoeffinger)
- Support for using WSL binaries on Windows #431 (rdoeffinger)
- Bugfixes and Windows compatibility improvements #430 (mrsch)
- Semi-automatic code cleanup #425 (annejan)
- Update to prevent the installer requesting admin #424 (hughwilliams94)
- Display passwords as QR codes #421 (frawi)
- Tested working on macOS HS with pinentry-mac #419 (riccardocossu)
- Dutch (nl) translation improvements #418 (equaeghe)
- Bugfixes #413 (rdoeffinger)
- pwgen: fix inverted "Generate ... less secure passwords" checkbox #409 (ahippo)
- Continuing refactoring #407 (FiloSpaTeam)
- #390 make box cheched when opening a folder users panel #403 (kenji21)

v1.2.3 (2018-06-04)

Full Changelog

Closed issues:

- Consider repology badges #396
- Unable to create new password #391
- Duplicate prefix in installation of tests directory in v1.2.2. #389
- Compilation error on FreeBSD member access into incomplete type #388
- No icons on macOS #377

Merged pull requests:

- Add support for OTP code generation on Linux as requested in #327 #394 (Noettore)
- Revert scroll bar changes #393 (destanyol)
- Fix High Dpi Support. Works now under Windows and KDE/Plasma. #392 (hgraeber)

v1.2.2 (2018-05-07)

Full Changelog

Implemented enhancements:

- Cleaning #includes #364 (FiloSpaTeam)

Fixed bugs:

- Insecure Password Generation #338
- Clipboard clearing timer is not reset when new passwords are copied to the clipboard #309
- Removal of files outside of password-store #300
- Some fixes and refactoring. #376 (FiloSpaTeam)
- Fix & make clearClipboard more robust #359 (lukedirtwalker)

Closed issues:

- Multiple question marks while trying to delete password #385
- No button icons and text in "menu bar" #383
- Cannot add a new password #380
- Tiny bit of regression #379

- Running qtPass remotelly not prompting for the GPG key passphrasse #374
- Entire program is huge on High DPI screen on Linux #369
- Two new issues since latest refactoring #368
- Chocolatey package outdated #366
- How do I change the language ? #352
- Parallel make issue in qtpass-1.2.1: ld: cannot find -lqtpass #350
- "copy" icon has disappeared in v1.2.1 #344
- No password entry prompt #343
- Can't install on macOS Sierra #337
- No icon on macOS #333
- Font and spacing used for URL links on right in main window absurdly large #329
- QtPass don't display all lines with templates #273

Merged pull requests:

- 2 simple fixes #386 (FiloSpaTeam)
- Should fix #383 #384 (FiloSpaTeam)
- Move connect action to main.cpp. Default search text as parameter of... #382 (FiloSpaTeam)
- fix #380 #381 (FiloSpaTeam)
- Small refactoring. #378 (FiloSpaTeam)
- Sorry for last error :) #370 (FiloSpaTeam)
- Optimizations :) #367 (FiloSpaTeam)
- Removed comment out #includes #365 (FiloSpaTeam)
- fix for #300 #363 (FiloSpaTeam)
- Translated all missing content to Italian, created Release of transla... #361 (FiloSpaTeam)
- Refactoring #360 (lukedirtwalker)
- Display all fields when using template setting, fixes #273 #358 (lukedirtwalker)
- Update CONTRIBUTING.md #354 (5bentz)
- Add two entries in FAQ about the language #353 (5bentz)
- Fix typo in french translation #349 (babolivier)
- New scroll bar on large files #347 (destanyol)
- Fix nested template argument list compile error #346 (martinburchell)
- Honor PREFIX during tests install #345 (SpiderX)

v1.2.1 (2018-01-04)

Full Changelog

Closed issues:

- Question: is it possible to mass import passes? #339
- Version 1.2.0 leaks passwords #334
- signed release files #332
- 2017 #330
- When importing settings from 1.1.5 or older clipboard settings revert to No Clipboard #232

Merged pull requests:

- Insecure password generation #342 (annejan)
- Add Catalan translation #336 (rbuj)

v1.2.0 (2017-11-08)

Full Changelog

Implemented enhancements:

- Icon tray from system icon theme #318
- Copy button for each custom field #291
- Feature Request: Use primary selection instead of clipboard #280
- Add primary selection as clipboard option #281 (annejan)
- Feature: CTRL/CMD + Q closes the mainwindow #258 #259 (YoshiMan)
- Feature/testing moved sources to src added tests #257 (annejan)
- enabled drag and drop support for passwords and passwordfolders #245 (YoshiMan)
- Password dialog decoupling from MW #242 (tezeb)
- Refactoring of qpushbuttonwithclipboard and timers #241 (tezeb)
- added a copy button for each line to paste the content into the clipboard, "pass init -- path=" command with right path-parameter, lupdate qtpass.pro #218 (YoshiMan)

Fixed bugs:

- Do not hide passwords and no generator #267
- Weird behavior when turning on git support (auto push/pull) with non-clean git dir #128
- SingleApplication implementation buggy #26

Closed issues:

- Tab order is wrong in password dialog #331
- Missing icons since split to static lib #325
- "-session XXX" upon session restore taken as search string #320
- Instructions to install it on OSX maybe outdated #315
- QtPass hangs when trying to decrypt entry #313
- Unable to locate package (Linux Mint 17.3) #310
- Git commit signing #303
- Add to Linux brew #301
- Pass 1.7 testing #299
- Measure unit-test code coverage #298
- Config dialog: Propose "Password behaviour" label change #294
- make install currently broken. #289
- Unable to locate package (Raspbian) #287
- There is no `git cp` #272
- pass is apparently switching out pwgen #264
- Bugs since refactoring #262
- pass working fine but qtprocess failure with qtpass #260
- Feature: CTRL/CMD + Q closes the mainwindow #258
- Refactoring: removal of lastDecrypt #256
- Pass environment not set-up correctly #250
- Make fails - std c++11 not set #244
- Double-clicking might open previous entry instead of one double-clicked on #243
- Clean up ConfigDialog #235

Merged pull requests:

- Extract static library and separate main function #324 (tezeb)
- galego actualizado #323 (xmgz)
- Add sftp, ftps, webdav and webdavs as supported links #322 (cgonzalez)
- Ignore cmdline arguments if -session is used. #321 (Achimh3011)
- Finished French translation (and proof-read the already translated strings). #311 (Marcool04)
- Once again, code coverage #305 (tezeb)
- Fixed path of resources.qrc #297 (sideeffect42)
- Add pt_PT translation #295 (keitalbame)
- Update README.md #293 (joostruis)

- small band aid fix for password generation on windows #276 (treat1)
- Final step in process mgmt refactoring #275 (tezeb)
- Fix pwgen and refactor Pass::finished #271 (tezeb)
- Process specific signals for process management #270 (tezeb)
- #239 reencrypting after a drag and drop action #261 (YoshiMan)
- this if evaluetes ervery time to true #255 (YoshiMan)
- executeing pass show before editpassword dialog shows up #254 (YoshiMan)
- Minor fix for filenames and git push #251 (tezeb)
- Process management refactoring part 2 #249 (tezeb)
- refactoring - pass ifce, process mgmt #234 (tezeb)
- Solve Doubleclick issue #230 (jounathaen)
- refactoring, new QtPassSettings class, all settings should be read and written here #224 (YoshiMan)
- Moved @YoshiMan 's copy buttons inside the line Edit #222 (jounathaen)
- UI Improvements #220 (jounathaen)
- creating password store directory, if it doesnot exists, de_DE translation fixes and removed obsolete translations #216 (YoshiMan)

v1.1.6 (2016-12-02)

Full Changelog

Implemented enhancements:

- Feedback on copy button use #229
- Clickable URLs + open in default browser #226
- Deselecting password re-opens the file #221
- Copy password button should include tooltip to say why, when disabled #214
- QtPass starts by searching for -psn_0_12345 on macOS #213
- Copy after timeout #189
- Feature Request: Copy template fields with button #133
- Cannot create top level folder #127
- Feature: moving items (reordering folders) #116

Fixed bugs:

- Regression with new view mode when using templates and URLs #223
- Problems with high dpi screen #217
- Hangs forever on Generate GnuPG keypair #215

- recent change to passworddialog.cpp #188
- Re-opening entry in QtPass on Windows does not put login or URL values back in the right place #183

Closed issues:

- Click does not stick #233
- Doubleclick on Treeview does not open the edit dialouge #228
- Windows - Enable GPG SSH Authentication #225
- We need autotype . . #65

v1.1.5 (2016-10-19)

Full Changelog

Implemented enhancements:

- I translated for Simplified Chinese. #208
- Short fullname hangs QtPass keypair generation process for infinite time #202
- More options for password generation #98
- Git hangs on windows #71

Fixed bugs:

- view box is trimming whitespace #210

Closed issues:

- PREFIX is now really a prefix #185
- QtPass, git and windows #173

Merged pull requests:

- Allow ssh links #211 (cgonzalez)
- Increase maximum password length to 255 #209 (vladimiroff)
- Password templates #207 (jounathaen)
- Updated German Translation #206 (jounathaen)
- Italian translation #204 (dakk)
- keygendialog email and name validition (issue 202) #203 (dakk)
- Lookup validity field to check if keys are valid #201 (thotypous)
- Fix spelling error #200 (innir)

v1.1.4 (2016-09-26)

Full Changelog

Implemented enhancements:

- Re-assign permissions when adding users #161
- Main window immediately closes upon app launch #139

Fixed bugs:

- German umlauts fails #192
- Error after change configuration #190
- Bug: Special characters in Template #131
- Character encoding issue with GPG key #101
- saved password '§' turns to 'Â§' when copied to clipboard or shown when editing #91

Closed issues:

- Signed releases #186
- Why it's not listed in wikipedia.org/wiki/List_of_password_managers ? #164
- Bitdefender blocks installation and quarantines the .exe and .ink #138

Merged pull requests:

- issue 91 bugfix #199 (asalamon74)
- issue 101 bugfix #198 (asalamon74)
- Czech translation #195 (svetlemodry)

v1.1.3 (2016-06-10)

Full Changelog

Fixed bugs:

- edit of password broken with active "Automatically push" #177
- Clipboard not cleared when quitting or killing application #171
- Hide content doesn't work when using templates #160

Closed issues:

- Add a (small) manpage #174

v1.1.2 (2016-06-10)

Full Changelog

Implemented enhancements:

- qtpass on windows, space in front of URL and Username #182

Fixed bugs:

- Deletion of folder doesn't work on Debian/GNU Linux #181

Closed issues:

- gpg: decryption failed: No secret key #179
- "gpg-agent: command get_passphrase failed: No such file or directory" #156

Merged pull requests:

- add Appdata file and update desktop file #178 (daveol)
- HTTPS everywhere #176 (da2x)
- Fix build issues with MSVC2015 on Windows #175 (msvi)

v1.1.1 (2016-04-04)

Full Changelog

Implemented enhancements:

- Signed binaries #149
- Icon theme and Cinnamon #146
- Bind a key to the clear action #142
- Installation dependencies #140
- All text input fields need example text & edit dialogue changes #85
- OSX: Qt-window closed only reappears when 'active' and using tray incon #77

Fixed bugs:

- Spelling bug: German translation of push and pull #110
- gpg: decryption failed: No secret key #92

Closed issues:

- Remove outdated Debian packaging #165
- Same name for file and folder #159
- Icons don't work on nixos #157
- gpg: Sorry, we are in batchmode - can't get input #151

Merged pull requests:

- lupdate and Russian translation #170 (ahippo)
- Remove path to password store in commit message and a leading space. #169 (ahippo)
- Use –secure for pwgen and add more configurable options #168 (ahippo)
- Remove Debian packaging #166 (innir)
- update gl_Es #162 (xmgz)
- Two UI Tweaks #158 (lftl)
- configwindow.ui default/start tab set to "settings" #154 (jounathaen)
- FAQ update concerning button-icons on cinnamon #153 (jounathaen)

v1.1.0 (2016-01-25)

Full Changelog

Implemented enhancements:

- Clear text input: use system icon instead of x #84
- System Icons on Buttons and Doubleclick on treeView #124 (jounathaen)

Closed issues:

- [resolved] Error in compiling macOS El capitan #148

Merged pull requests:

- Pre 1.1 mixing #145 (annejan)
- RPM Spec file updates #137 (muff1nman)
- swedish translations #135 (ralphtheninja)

v1.0.6 (2016-01-03)

Full Changelog

Implemented enhancements:

- Feature: Always on top #118
- Option to show minimized instance #99

Fixed bugs:

- Bug: deleted record stays in memory #117

Closed issues:

- SIGSEGV in MainWindow::executeWrapper on clean install #122

Merged pull requests:

- improved the German translation #134 (retokromer)
- qrand always generating the same sequence of passwords #129 (treat1)
- some improvements #126 (retokromer)
- added one translation #125 (retokromer)
- initial attempt to create a RPM spec file #121 (bram-ivs)
- Cleanup and coding standards #120 (annejan)
- Modified the clipboard logic to allow for on-demand copy to clipboard. #119 (jonhanks)

v1.0.5 (2015-11-18)

Full Changelog

Fixed bugs:

- using pwgen adds carriage-return #115
- Enhancement: color code git results #111

Merged pull requests:

- Fix bug that prints "Unknown error" to the terminal #113 (dvaerum)

v1.0.4 (2015-11-03)

Full Changelog

Implemented enhancements:

- Add support for RightToLeft languages #108

v1.0.3 (2015-10-25)

Full Changelog

Implemented enhancements:

- Get PREFIX variable from environment #106
- Password file named 'git' returns error #105

Merged pull requests:

- Get PREFIX variable from environment #104 (jorti)
- spanish translations added #103 (mrpnkt)

v1.0.2 (2015-09-24)

Full Changelog

Closed issues:

- Generate password: Floating point exception (core dumped) #102
- A way to indicate the installation prefix is needed #100
- IPv4 URLs are non-clickable #97
- app crashes when "Use pwgen" is unselected, and "Generate" is clicked. #95
- Some minor improvements on the templating part #93
- app crashes with variant of "pwgen" app #90

v1.0.1 (2015-08-09)

Full Changelog

Implemented enhancements:

- Users setup - key colours could be improved #82

Closed issues:

- When QtPass starts, focus search input box #89
- Clear the password display after some time #86
- Auto push/pull #83
- qtpass doesn't commit deletes to git #81
- Always crashes while using the quick-search input #79
- Git initialisation #72
- Initialising new repo's doesn't work correctly #55
- gpg: Sorry, no terminal at all requested - can't get input #18

Merged pull requests:

- Issue 86 clear panel #87 (karlgrz)
- Update FAQ for Yubikey NEO helper in .bashrc for Ubuntu #80 (karlgrz)
- [WIP] Call 'pass git init' on creation of password-store when useGit #78 (dennisdegreef)

v1.0.0 (2015-08-01)

Full Changelog

Closed issues:

- Yubikey Neo Pin entry not working properly on Ubuntu 15.04 #73

Merged pull requests:

- Updating hungarian localisation #76 (damnlie)
- added DE translations #74 (Friedy)

v0.9.2 (2015-07-30)

Full Changelog

Closed issues:

- Show expiration date in key setup #70

v0.9.1 (2015-07-29)

Full Changelog

Closed issues:

- Minimize on startup. #69
- tray icon in xfce #58
- Git intergration #57
- Weird characters in filenames breaks loading gpg files #10

v0.9.0 (2015-07-17)

Full Changelog

Closed issues:

- Request: Integrate qtpass with pwgen for generating passwords. #68

v0.8.6 (2015-07-17)

Full Changelog

Closed issues:

- Copy password by Ctrl+C #60
- Remember window size and vertical pane width #59
- Multiline Editing #34

Merged pull requests:

- To make building successfull wi Desktop Qt 5.4.0 MSVC2012 OpenGL 32bit #67 (annejan)

v0.8.5.1 (2015-07-08)

Full Changelog

v0.8.5 (2015-07-08)

Full Changelog

Closed issues:

- Won't compile on Kubuntu 15.10 #61
- Hanging process gives weird effects #56
- Directory separator actually broken by 208171fd09c55ad765fdf4fa1de9a7f0757fa72d #53

Merged pull requests:

- Many deadlocks and other nasty bugfixes #64 (annejan)
- Mention qt5-default package in README #62 (lorrin)
- Some hacks I needed for portable gpg4win release #54 (rdoeffinger)

v0.8.4 (2015-06-11)

Full Changelog

Closed issues:

- QtPass does not detect GPG installation #50
- Cannot create new folders #48
- Better error handling when no pass or gpg found initially #13

Merged pull requests:

- Develop #52 (annejan)
- Minor thingies #51 (beefcurtains)

v0.8.3 (2015-06-09)

Full Changelog

Merged pull requests:

- Bugfixes #49 (rdoeffinger)

v0.8.2 (2015-05-27)

Full Changelog

Closed issues:

- Typo in 37f17f3808c1c97bd72c165a530c67a4bfb82edb? #45
- Signing of keys from user management #41

Merged pull requests:

- Provide more information in user list. #47 (rdoeffinger)
- Enable C++11 and use it to simplify loops. #46 (rdoeffinger)

v0.8.1 (2015-05-06)

Full Changelog

Fixed bugs:

- Some items not found on first search #8

Closed issues:

- compiling qtpass on ubuntu 15.04 - fails due to newer qmake version #43
- QProcess::start: Process is already running #40
- Extra line breaks seem to be added to the (HTML) output #39
- Missing develop branch and release testing #38
- Windows WebDAV broken by 24f8dec3c203921f765e923e6ae6a4069b8cf50a #36
- .gpg-id file not added to git #35
- Icon filenames #31
- GNUPGHOME environment variable #30
- Feature: webdav alternative to git #28
- Windows: not working due to pointless use of "sh" #16
- Windows: support static build and enable ASLR and NX #15
- Some paths to executables are printed when starting up #11

Merged pull requests:

- SingleApplication per user and leading newline removed from output #44 (annejan)

- User filtering and many fixes #42 (annejan)
- Re-enable Windows WebDAV support. #37 (rdoeffinger)
- User robustness #33 (rdoeffinger)
- Add WebDAV support. #29 (rdoeffinger)
- Add nosingleapp config. #27 (rdoeffinger)
- Add Makefile with commands to make a binary release zip file. #25 (rdoeffinger)
- Start process only after we finished disabling UI elements etc. #24 (rdoeffinger)
- Support for editing .gpg-id via GUI with public keyring list. #23 (rdoeffinger)
- More proper support for subdirectories. #22 (rdoeffinger)
- Russian translation (+typo fixed) #20 (mexus)
- Windows-related fixes. #17 (rdoeffinger)
- Deal with "special" characters #14 (JiCiT)
- galician and spanish localization files created #12 (xmgz)
- Update localization_hu_HU.ts #9 (damnlie)
- Replace which invocations with actual path resolution code #7 (shitbangs)
- Added Swedish and Polish localization to resources #6 (iamtew)
- Swedish localization #5 (iamtew)
- Update localization_hu_HU.ts #4 (reesenemesis)
- Update localization_hu_HU.ts #3 (reesenemesis)
- pass #2 (guaka)
- Beginning of German translation #1 (mwfc)

Chapter 3

Contributor Covenant Code of Conduct

Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at help@qtpass.org. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

For answers to common questions about this code of conduct, see https://www.contributor-covenant.org/faq

Chapter 4

Contributing

Make sure you have read the FAQ

Thank you for wanting to contribute to making QtPass awesome.

This document

This document is still in a very early stage and needs a lot more work.

Pull Request Process

1. Ensure install or build dependencies and artefacts are not commmitted.
2. When adding big new features or changes to the build tool, update the README.md to reflect those.
3. Make sure you update all of the CI configs if need be. These are ran on every Pull Request.

1. Increase the version numbers in relevant files when applicable. The versioning scheme we use is `SemVer`.
2. You may merge the Pull Request in once you have the sign-off of one other developer, or if you do not have permission to do that, you may request a reviewer to merge it for you.

Translations

- Add you language to the `src/src.pro` file under TRANSLATIONS.
- Next run the command `qmake` which will create and update the localization files.
- Edit your file with (let's imagine your language is sv_SE (Swedish) `linguist localization/localization_sv_SE.ts`

Qt Linguist has very nice in-context translation options `for translators`

You can do online translations via `Weblate`

IRC

For questions or brainstorming about features please join #ijhack on freenode.

Gitter

Or if you prefer to use `gitter`

License

QtPass is released under the GNU GPL v3.0 license. http://www.gnu.org/licenses/gpl-3.0.html

Chapter 5

FAQ

Issues

Can't save a password

- Is folder initialised? Easiest way is to use the [Users] button and make sure you can encrypt for someone (eg. yourself)
- Are you using git? If not, make sure it is switched off.

I have an issue with GNOME keyring

- Disable GNOME keyring
- Create a `~/.gnupg/gpg-agent.conf` containing:

```
enable-ssh-support
write-env-file
use-standard-socket
default-cache-ttl 600
max-cache-ttl 7200
```

Also, the following is useful to add to your .bashrc if you are using Yubikey NEO on Ubuntu:

```
# OpenPGP applet support for YubiKey NEO
if [ ! -f /tmp/gpg-agent.env ]; then
    killall gpg-agent;
        eval $(gpg-agent --daemon --enable-ssh-support > /tmp/gpg-agent.env);
fi
. /tmp/gpg-agent.env
```

- More info: `issue 60` and `issue 73`

I don't get a passphrase / PIN dialog

- You'll need to install pinentry-qt (or -qt4 or -qt5 or even -gtk) and possibly set the full path to that executable in your `~/.gnupg/gpg-agent.conf` for example: `pinentry-program /usr/bin/pinentry-qt4`
- On some esotheric systems it might be necessary to create a symbolic link `/usr/bin/pinentry` to your pinentry application of choice eg: `ln -s /usr/bin/pinentry-qt5 /usr/bin/pinentry`
- On macOS `pinentry-program /usr/local/bin/pinentry-mac` works after installing `pinentry-mac` from homebrew.

I have an other issue with gpg

- Possibly you have you key only in gpg and not in gpg2

```
gpg --export [ID] > public.key
gpg --export-secret-key [ID] > private.key
gpg2 --import public.key
gpg2 --import private.key
rm public.key private.key
```

Where [ID] is your gpg key-id.

- It might be the case where it is the other way around, exchange gpg and gpg2 accordingly . .

Git doesn't work on Windows

git for Windows comes with an `ssh-askpass` compatible command, git `gui--askpass` (located in `/mingw64/libexec/git-core/git-gui--askpass` on PortableGit version, presumably some place similar for the installed version).

Git has issues with GPG SSH Authentication

This tutorial might resolve your issues. https://github.com/git-for-windows/git/wiki/OpenSSH-Integration-with-Pageant

Where is the configuration stored?

QtPass tries to use the native config choice for the OS it's running.

- Linux and BSD: `$HOME/.config/IJHack/QtPass.conf`
- macOS: `$HOME/Library/Preferences/com.IJHack.QtPass.plist`
- Windows registry: `HKEY_CURRENT_USER\Software\IJhack\QtPass`

These settings can be over-ruled by a `qtpass.ini` file in the folder where the application resides. So called "portable config".

There are some things to take care of when trying to sync on some systems (especially OSX, with regards to text and binary .plist files).

More information: http://doc.qt.io/qt-5/qsettings.html#platform-specific-notes

Where can I ask for help?

- Create an `issue` issues on GitHub.
- Send an email to help@qtpass.org

Can I import from KeePass, LastPass or X?

- Yes, check `passwordstore.org/#migration` for more info.

I don't see icons on the buttons

You do not have the Qt SVG library installed. Please install using your favorite package manager.

I get icons that do not fit my (X11) default

- On some WindowManagers, Qt doesn't know what icon set to use. A trick:

```
export DESKTOP_SESSION=gnome
```

- Another possible reason is, that the currently installed Qt Version gives problems (e.g. on Linux Mint 17.3) Then you'll have to install the current version via your package manager or if this is not up-to-date, download it from `https://www.qt.io/download/` install it and run:

```
/PATHTOYOURQTINSTALLATION/5.5/gcc_64/bin/qmake
make
(sudo) make install
```

where `PATHTOYOURINSTALLATION` is the path you selected in the qt installer (default `/home/YOURUSER/Qt/`) and 5.5 has to be adapted for the Qt version you downloaded.

I don't like the design, what gives?

- It's all on GitHub, clone, change and send a pull request.
- Open an issue and point out defects or better yet propose changes.

QtPass is not in my native language

- Unfortunately, QtPass might not support your native language, or the translations might be incomplete. Check if newer versions of QtPass support it.
- If translations are available but aren't working, try to set the language manually (see below) or open an issue.

How do I set the language manually?

QtPass uses the system language. Changing it depends on your system:

- on Linux: `LANGUAGE=fr qtpass` will run QtPass in French.

How can I help improve QtPass?

I would like to donate

- Time:
  - Read contributing documentation.
  - Fork, clone hack and send a pull request.
  - Find an `issue` to work on..
  - Participate in our bug bounty, you submit an issue and help us fix it, I send you a bounty.
- Money:
  - IJhack takes donations in `Bitcoin`

Chapter 6

QtPass

QtPass is a GUI for `pass`, the standard unix password manager.

Features

- Using `pass` or `git` and `gpg2` directly
- Configurable shoulder surfing protection options
- Cross platform: Linux, BSD, macOS and Windows
- Per-folder user selection for multi recipient encryption
- Multiple profiles
- Easy onboarding

Logo based on `Heart-padlock by AnonMoos`.

Installation

From package

OpenSUSE & Fedora `yum install qtpass` `dnf install qtpass`

Debian, Ubuntu and derivates like Mint, Kali & Raspbian `apt-get install qtpass`

Arch Linux `pacman -S qtpass`

Gentoo `emerge -atv qtpass`

Sabayon `equo install qtpass`

FreeBSD `pkg install qtpass`

macOS `brew install --cask qtpass`

Windows `choco install qtpass`

From Source

Dependencies

- QtPass requires Qt 5.10 or later (Qt 6 works too)
- The Linguist package is required to compile the translations.
- For use of the fallback icons the SVG library is required.

At runtime the only real dependency is `gpg2` but to make the most of it, you'll need `git` and `pass` too.

Your GPG has to be set-up with a graphical pinentry when applicable, same goes for git authentication. On Mac macOS this currently seems to only work best with `pinentry-mac` from homebrew, although gpgtools works too.

On most unix systems all you need is:

```
qmake && make && make install
```

Using profiles

Profiles allow to group passwords. Each profile might use a different git repository and/or different gpg key. Each profile also can be associated with a pass store singing key to verify the detached .gpg-id signature. A typical use case is to separate personal and work passwords.

> **Hint**
> Instead of using different git repositories for the various profiles passwords could be synchronized with different branches from the same repository. Just clone the repository into the profile folders and checkout the related branch.

Example

The following commands set up two profile folders:

```
cd ~/.password-store/
git clone https://github.com/vendor/personal-passwords personal && echo "personal/" » .gitignore
git clone https://github.com/company/group-passwords work && echo "work/" » .gitignore
pass init -p personal [personal GnuPG-ID] && git -C personal push
pass init -p work [work GnuPG-ID] && git -C work push
```

Note:

- Replace `[personal GnuPG-ID]` and `[work GnuPG-ID]` with the ID from the related GnuPG key.
- The parts `echo ... >> .gitignore` are just needed in case there is a git repository present in the base directory.

Once the repositories and GnuPG-ID's have been defined the profiles can be set up in QtPass.

Links of interest

- `Git Tools - Credential Storage`
- `Dealing with secrets`
- `Git Credential Manager`
- `password-store`

Testing

This is done with `make check`

Codecoverage can be done with `make lcov`, `make gcov`, `make coveralls` and/or `make codecov`.

Be sure to first run: `make distclean && qmake CONFIG+=coverage qtpass.pro`

Security considerations

Using this program will not magically keep your passwords secure against compromised computers even if you use it in combination with a smartcard.

It does protect future and changed passwords though against anyone with access to your password store only but not your keys. Used with a smartcard it also protects against anyone just monitoring/copying all files/keystrokes on that machine and such an attacker would only gain access to the passwords you actually use. Once you plug in your smartcard and enter your PIN (or due to CVE-2015-3298 even without your PIN) all your passwords available to the machine can be decrypted by it, if there is malicious software targeted specifically against it installed (or at least one that knows how to use a smartcard).

To get better protection out of use with a smartcard even against a targeted attack I can think of at least two options:

- The smartcard must require explicit confirmation for each decryption operation. Or if it just provides a counter for decrypted data you could at least notice an attack afterwards, though at quite some effort on your part.
- Use a different smartcard for each (group of) key.
- If using a YubiKey or U2F module or similar that requires a "button" press for other authentication methods you can use one OTP/U2F enabled WebDAV account per password (or groups of passwords) as a quite inconvenient workaround. Unfortunately I do not know of any WebDAV service with OTP support except ownCloud (so you would have to run your own server).

Known issues

- Filtering (searching) breaks the tree/model sometimes
- Starting without a correctly set password-store folder gives weird results in the tree view

Planned features

- Plugins based on field name, plugins follow same format as password files
- Colour coding folders (possibly disabling folders you can't decrypt)
- Optional table view of decrypted folder contents
- Opening of (basic auth) URLs in default browser? Possibly with helper plugin for filling out forms?
- WebDAV (configuration) support
- Some other form of remote storage that allows for accountability / auditing (web API to retrieve the .gpg files?)

Further reading

FAQ and CONTRIBUTING documentation. CHANGELOG

Site Source code Issue queue Chat

License

GNU GPL v3.0

View official GNU site

<img src="https://opensource.org/wp-content/uploads/2022/10/osi-badge-dark.↩
svg" alt="OSI-approved license" width="127">

View the Open Source Initiative site

Chapter 7

Todo List

Member ConfigDialog::genKey (QString, QDialog ∗)

refactor the process to not be entangled so much.

Member ConfigDialog::wizard ()

make this thing more reliable.

Chapter 8

Namespace Index

8.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 9

Hierarchical Index

9.1 Class Hierarchy

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

Chapter 10

Class Index

10.1 Class List

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

Chapter 11

File Index

11.1 File List

Here is a list of all files with brief descriptions:

Chapter 12

Namespace Documentation

12.1 Enums Namespace Reference

Enumerators for configuration and runtime items.

Enumerations

- enum clipBoardType { CLIPBOARD_NEVER = 0 , CLIPBOARD_ALWAYS = 1 , CLIPBOARD_ON_DEMAND = 2 }
- enum PROCESS {
  GIT_INIT = 0 , GIT_ADD , GIT_COMMIT , GIT_RM ,
  GIT_PULL , GIT_PUSH , PASS_SHOW , PASS_INSERT ,
  PASS_REMOVE , PASS_INIT , GPG_GENKEYS , PASS_MOVE ,
  PASS_COPY , GIT_MOVE , GIT_COPY , PROCESS_COUNT ,
  INVALID , PASS_OTP_GENERATE }

12.1.1 Detailed Description

Enumerators for configuration and runtime items.

12.1.2 Enumeration Type Documentation

12.1.2.1 clipBoardType

```
enum Enums::clipBoardType
```

Enumerator

CLIPBOARD_NEVER	
CLIPBOARD_ALWAYS	
CLIPBOARD_ON_DEMAND	

Definition at line 10 of file enums.h.

12.1.2.2 PROCESS

enum Enums::PROCESS

Enumerator

GIT_INIT	
GIT_ADD	
GIT_COMMIT	
GIT_RM	
GIT_PULL	
GIT_PUSH	
PASS_SHOW	
PASS_INSERT	
PASS_REMOVE	
PASS_INIT	
GPG_GENKEYS	
PASS_MOVE	
PASS_COPY	
GIT_MOVE	
GIT_COPY	
PROCESS_COUNT	
INVALID	
PASS_OTP_GENERATE	

Definition at line 16 of file enums.h.

12.2 Ui Namespace Reference

Chapter 13

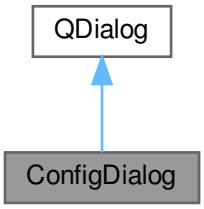
Class Documentation

13.1 ConfigDialog Class Reference

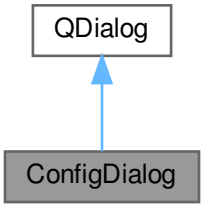
The ConfigDialog handles the configuration interface.

`#include <configdialog.h>`

Inheritance diagram for ConfigDialog:



Collaboration diagram for ConfigDialog:



Public Member Functions

- ConfigDialog (MainWindow *parent)

  ConfigDialog::ConfigDialog this sets up the configuration screen.
- ~ConfigDialog ()

  ConfigDialog::~ConfigDialog config destructor, makes sure the mainWindow knows about git, gpg and pass executables.
- void useSelection (bool useSelection)

  ConfigDialog::useSelection set the clipboard type use from MainWindow.
- void useAutoclear (bool useAutoclear)

  ConfigDialog::useAutoclear set the clipboard autoclear use from MainWindow.
- void useAutoclearPanel (bool useAutoclearPanel)

  ConfigDialog::useAutoclearPanel set the panel autoclear use from MainWindow.
- QHash< QString, QHash< QString, QString > > getProfiles ()

  ConfigDialog::getProfiles return profile list.
- void wizard ()

  ConfigDialog::wizard first-time use wizard.
- void genKey (QString, QDialog *)

  ConfigDialog::genKey tunnel function to make MainWindow generate a gpg key pair.
- void useTrayIcon (bool useSystray)

  ConfigDialog::useTrayIcon set preference for using trayicon. Enable or disable related checkboxes accordingly.
- void useGit (bool useGit)

  ConfigDialog::useGit set preference for using git.
- void useOtp (bool useOtp)

  ConfigDialog::useOtp set preference for using otp plugin.
- void useQrencode (bool useQrencode)

  ConfigDialog::useOtp set preference for using otp plugin.
- void setPwgenPath (QString)

  ConfigDialog::setPwgenPath set pwgen executable path. Enable or disable related options in the interface.
- void usePwgen (bool usePwgen)

  ConfigDialog::usePwgen set preference for using pwgen (can be overruled buy empty pwgenPath). enable or disable related options in the interface via ConfigDialog::on_checkBoxUsePwgen_clicked.
- void setPasswordConfiguration (const PasswordConfiguration &config)
- PasswordConfiguration getPasswordConfiguration ()
- void useTemplate (bool useTemplate)

  ConfigDialog::useTemplate set preference for using templates.

Protected Member Functions

- void closeEvent (QCloseEvent *event)

  ConfigDialog::closeEvent close this window.

13.1.1 Detailed Description

The ConfigDialog handles the configuration interface.

This class should also take the handling from the MainWindow class.

Definition at line 24 of file configdialog.h.

13.1.2 Constructor & Destructor Documentation

13.1.2.1 ConfigDialog()

```
ConfigDialog::ConfigDialog (
            MainWindow * parent )  [explicit]
```

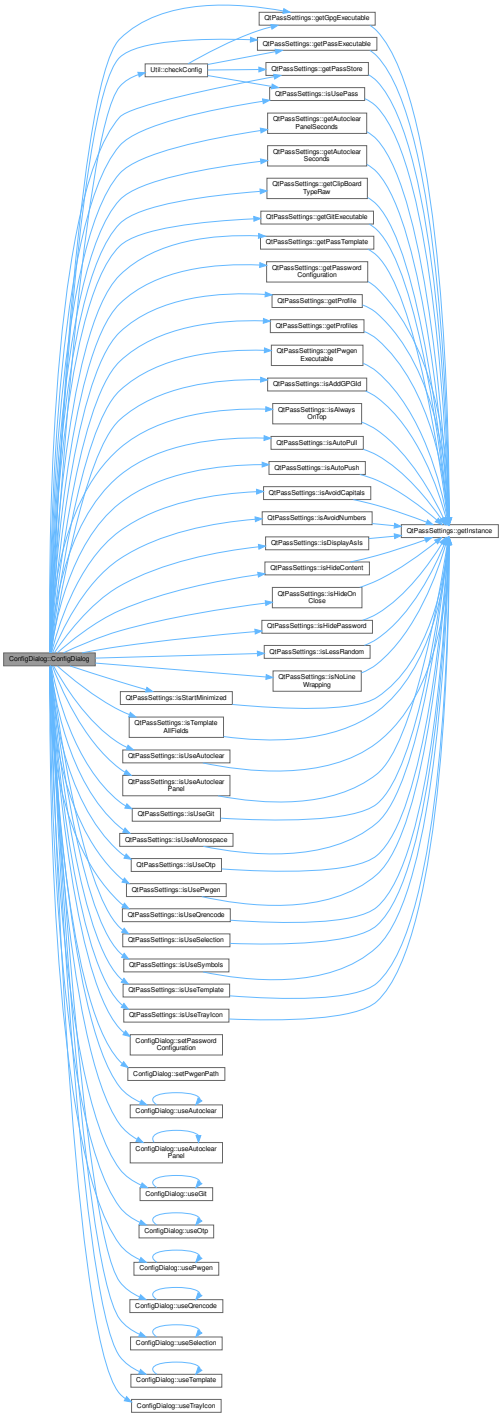
ConfigDialog::ConfigDialog this sets up the configuration screen.

Parameters

parent	

Definition at line 26 of file configdialog.cpp.

Here is the call graph for this function:



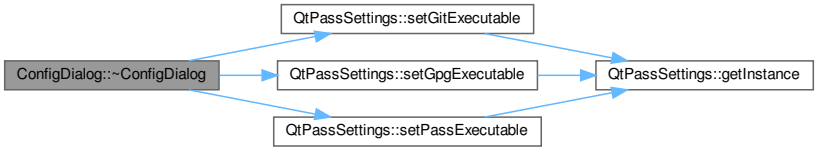
13.1.2.2 ~ConfigDialog()

```
ConfigDialog::~ConfigDialog ( )
```

ConfigDialog::~ConfigDialog config destructor, makes sure the mainWindow knows about git, gpg and pass executables.

Definition at line 137 of file configdialog.cpp.

Here is the call graph for this function:



13.1.3 Member Function Documentation

13.1.3.1 closeEvent()

```
void ConfigDialog::closeEvent (
            QCloseEvent * event )  [protected]
```

ConfigDialog::closeEvent close this window.

Parameters

event	

Definition at line 720 of file configdialog.cpp.

13.1.3.2 genKey()

```
void ConfigDialog::genKey (
            QString batch,
            QDialog * dialog )
```

ConfigDialog::genKey tunnel function to make MainWindow generate a gpg key pair.

Todo refactor the process to not be entangled so much.

Parameters

batch	
dialog	

Definition at line 455 of file configdialog.cpp.

Here is the call graph for this function:

13.1.3.3 getPasswordConfiguration()

```
PasswordConfiguration ConfigDialog::getPasswordConfiguration ( )
```

Definition at line 827 of file configdialog.cpp.

13.1.3.4 getProfiles()

```
QHash< QString, QHash< QString, QString > > ConfigDialog::getProfiles ( )
```

ConfigDialog::getProfiles return profile list.

Returns

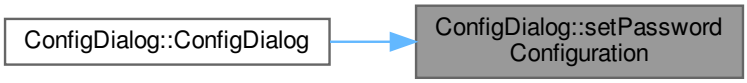
Definition at line 496 of file configdialog.cpp.

13.1.3.5 setPasswordConfiguration()

```
void ConfigDialog::setPasswordConfiguration (
            const PasswordConfiguration & config )
```

Definition at line 818 of file configdialog.cpp.

Here is the caller graph for this function:



13.1.3.6 setPwgenPath()

```
void ConfigDialog::setPwgenPath (
            QString pwgen )
```

ConfigDialog::setPwgenPath set pwgen executable path. Enable or disable related options in the interface.

Parameters

pwgen	

Definition at line 780 of file configdialog.cpp.

Here is the caller graph for this function:

13.1.3.7 useAutoclear()

```
void ConfigDialog::useAutoclear (
            bool useAutoclear )
```

ConfigDialog::useAutoclear set the clipboard autoclear use from MainWindow.

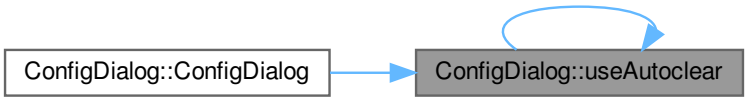
Parameters

useAutoclear	

Definition at line 417 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.1.3.8 useAutoclearPanel()

```
void ConfigDialog::useAutoclearPanel (
            bool useAutoclearPanel )
```

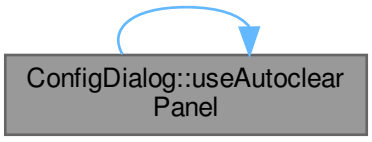
ConfigDialog::useAutoclearPanel set the panel autoclear use from MainWindow.

Parameters

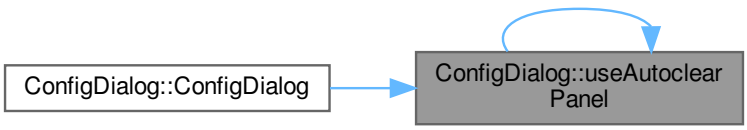
useAutoclearPanel	

Definition at line 427 of file configdialog.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.1.3.9 useGit()

```
void ConfigDialog::useGit (
            bool useGit )
```

ConfigDialog::useGit set preference for using git.

Parameters

useGit	

Definition at line 729 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

13.1.3.10 useOtp()

```
void ConfigDialog::useOtp (
            bool useOtp )
```

ConfigDialog::useOtp set preference for using otp plugin.

Parameters

useOtp	

Definition at line 738 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

13.1.3.11 usePwgen()

```
void ConfigDialog::usePwgen (
            bool usePwgen )
```

ConfigDialog::usePwgen set preference for using pwgen (can be overruled buy empty pwgenPath). enable or disable related options in the interface via ConfigDialog::on_checkBoxUsePwgen_clicked.

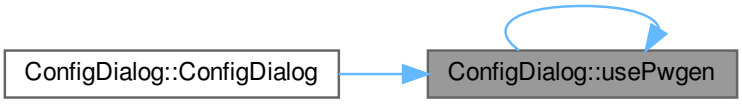
Parameters

usePwgen	

Definition at line 811 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.1.3.12 useQrencode()

```
void ConfigDialog::useQrencode (
            bool useQrencode )
```

ConfigDialog::useOtp set preference for using otp plugin.

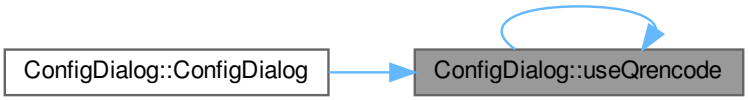
Parameters

useOtp	

Definition at line 746 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.1.3.13 useSelection()

```
void ConfigDialog::useSelection (
            bool useSelection )
```

ConfigDialog::useSelection set the clipboard type use from MainWindow.

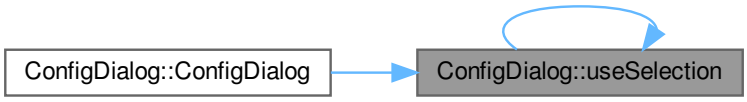
Parameters

useSelection	

Definition at line 407 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.1.3.14 useTemplate()

```
void ConfigDialog::useTemplate (
            bool useTemplate )
```

ConfigDialog::useTemplate set preference for using templates.

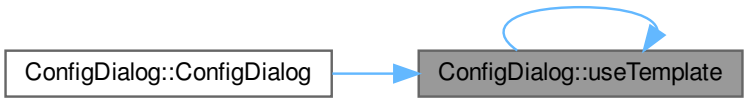
Parameters

useTemplate	

Definition at line 871 of file configdialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.1.3.15 useTrayIcon()

```
void ConfigDialog::useTrayIcon (
            bool useSystray )
```

ConfigDialog::useTrayIcon set preference for using trayicon. Enable or disable related checkboxes accordingly.

Parameters

useSystray	

Definition at line 693 of file configdialog.cpp.

Here is the caller graph for this function:

13.1.3.16 wizard()

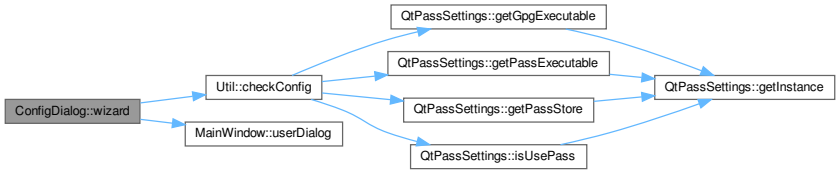
```
void ConfigDialog::wizard ( )
```

ConfigDialog::wizard first-time use wizard.

Todo make this thing more reliable.

Definition at line 594 of file configdialog.cpp.

Here is the call graph for this function:



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

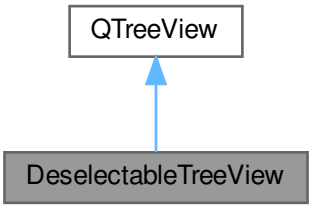
- src/configdialog.h
- src/configdialog.cpp

13.2 DeselectableTreeView Class Reference

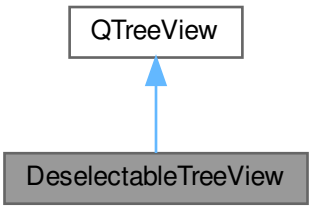
The DeselectableTreeView class loosly based on http://stackoverflow.com/questions/2761284/ thanks to Yassir Ennazk.

`#include <deselectabletreeview.h>`

Inheritance diagram for DeselectableTreeView:



Collaboration diagram for DeselectableTreeView:



Signals

- void emptyClicked ()

  emptyClicked event

Public Member Functions

- DeselectableTreeView (QWidget *parent)

  DeselectableTreeView standard constructor.
- virtual ~DeselectableTreeView ()

  ~DeselectableTreeView standard destructor

13.2.1 Detailed Description

The DeselectableTreeView class loosly based on http://stackoverflow.com/questions/2761284/ thanks to Yassir Ennazk.

Definition at line 14 of file deselectabletreeview.h.

13.2.2 Constructor & Destructor Documentation

13.2.2.1 DeselectableTreeView()

```
DeselectableTreeView::DeselectableTreeView (
            QWidget * parent )  [inline]
```

DeselectableTreeView standard constructor.

Parameters

parent	

Definition at line 22 of file deselectabletreeview.h.

13.2.2.2 ~DeselectableTreeView()

```
virtual DeselectableTreeView::~DeselectableTreeView ( )  [inline], [virtual]
```

~DeselectableTreeView standard destructor

Definition at line 26 of file deselectabletreeview.h.

13.2.3 Member Function Documentation

13.2.3.1 emptyClicked

```
void DeselectableTreeView::emptyClicked ( )  [signal]
```

emptyClicked event

Here is the caller graph for this function:

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

- src/deselectabletreeview.h

13.3 dragAndDropInfo Struct Reference

holds values to share beetween drag and drop on the passwordstorage view

`#include <storemodel.h>`

13.3.1 Detailed Description

holds values to share beetween drag and drop on the passwordstorage view

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

- src/storemodel.h

13.4 dragAndDropInfoPasswordStore Struct Reference

`#include <storemodel.h>`

Public Attributes

- bool isDir
- bool isFile
- QString path

13.4.1 Detailed Description

Definition at line 45 of file storemodel.h.

13.4.2 Member Data Documentation

13.4.2.1 isDir

```
bool dragAndDropInfoPasswordStore::isDir
```

Definition at line 46 of file storemodel.h.

13.4.2.2 isFile

```
bool dragAndDropInfoPasswordStore::isFile
```

Definition at line 47 of file storemodel.h.

13.4.2.3 path

```
QString dragAndDropInfoPasswordStore::path
```

Definition at line 48 of file storemodel.h.

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

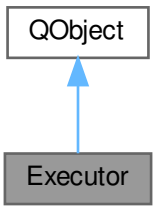
- src/storemodel.h

13.5 Executor Class Reference

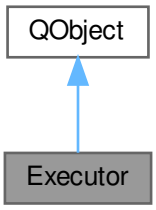
Executes external commands for handleing password, git and other data.

```
#include <executor.h>
```

Inheritance diagram for Executor:



Collaboration diagram for Executor:



Signals

- void finished (int id, int exitCode, const QString &output, const QString &errout)

  finished signal that is emited when process finishes
- void starting ()

  starting signal that is emited when process starts
- void error (int id, int exitCode, const QString &output, const QString &errout)

  error signal that is emited when process finishes with an error

Public Member Functions

- Executor (QObject ∗parent=0)

  Executor::Executor executes external applications.
- void execute (int id, const QString &app, const QStringList &args, bool readStdout, bool readStderr=true)

  Executor::execute execute an app.
- void execute (int id, const QString &workDir, const QString &app, const QStringList &args, bool readStdout, bool readStderr=true)

  Executor::execute executes an app from a workDir.
- void execute (int id, const QString &app, const QStringList &args, QString input=QString(), bool read↩ Stdout=false, bool readStderr=true)

  Executor::execute an app, takes input and presents it as stdin.
- void execute (int id, const QString &workDir, const QString &app, const QStringList &args, QString input=QString(), bool readStdout=false, bool readStderr=true)

  Executor::execute executes an app from a workDir, takes input and presents it as stdin.
- void setEnvironment (const QStringList &env)

  Executor::setEnvironment set environment variables for executor processes.
- int cancelNext ()

  Executor::cancelNext cancels execution of first process in queue if it's not already running.

Static Public Member Functions

- static int executeBlocking (QString app, const QStringList &args, QString input=QString(), QString ∗process↩ _out=Q_NULLPTR, QString ∗process_err=Q_NULLPTR)

  Executor::executeBlocking blocking version of the executor, takes input and presents it as stdin.
- static int executeBlocking (QString app, const QStringList &args, QString ∗process_out, QString ∗process↩ _err=Q_NULLPTR)

  Executor::executeBlocking blocking version of the executor.

13.5.1 Detailed Description

Executes external commands for handleing password, git and other data.

Definition at line 12 of file executor.h.

13.5.2 Constructor & Destructor Documentation

13.5.2.1 Executor()

```
Executor::Executor (
            QObject * parent = 0 )  [explicit]
```

Executor::Executor executes external applications.

Parameters

parent	

Definition at line 19 of file executor.cpp.

13.5.3 Member Function Documentation

13.5.3.1 cancelNext()

```
int Executor::cancelNext ( )
```

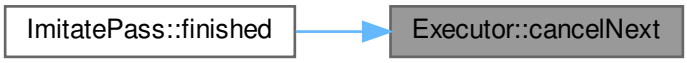
Executor::cancelNext cancels execution of first process in queue if it's not already running.

Returns

id of the cancelled process or -1 on error

Definition at line 233 of file executor.cpp.

Here is the caller graph for this function:



13.5.3.2 error

```
void Executor::error (
            int id,
            int exitCode,
            const QString & output,
            const QString & errout )  [signal]
```

error signal that is emited when process finishes with an error

Parameters

id	id of the process
exitCode	return code of the process
output	stdout produced by the process
errout	stderr produced by the process

13.5.3.3 execute() [1/4]

```
void Executor::execute (
            int id,
            const QString & app,
            const QStringList & args,
            bool readStdout,
            bool readStderr = true )
```

Executor::execute execute an app.

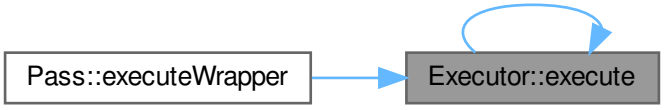
Parameters

id	
app	
args	
readStdout	
readStderr	

Definition at line 68 of file executor.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.5.3.4 execute() [2/4]

```
void Executor::execute (
            int id,
            const QString & app,
            const QStringList & args,
            QString input = QString(),
            bool readStdout = false,
            bool readStderr = true )
```

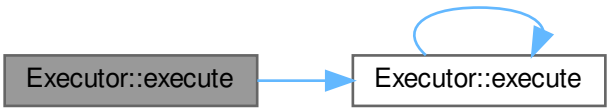
Executor::execute an app, takes input and presents it as stdin.

Parameters

id	
app	
args	
input	
readStdout	
readStderr	

Definition at line 97 of file executor.cpp.

Here is the call graph for this function:



13.5.3.5 execute() [3/4]

```
void Executor::execute (
            int id,
            const QString & workDir,
            const QString & app,
            const QStringList & args,
            bool readStdout,
            bool readStderr = true )
```

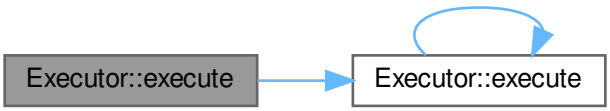
Executor::execute executes an app from a workDir.

Parameters

id	
workDir	
app	
args	
readStdout	
readStderr	

Definition at line 82 of file executor.cpp.

Here is the call graph for this function:



13.5.3.6 execute() [4/4]

```
void Executor::execute (
            int id,
            const QString & workDir,
            const QString & app,
            const QStringList & args,
            QString input = QString(),
            bool readStdout = false,
            bool readStderr = true )
```

Executor::execute executes an app from a workDir, takes input and presents it as stdin.

Parameters

id	
workDir	
app	
args	
input	
readStdout	
readStderr	

Definition at line 113 of file executor.cpp.

13.5.3.7 executeBlocking() [1/2]

```
int Executor::executeBlocking (
            QString app,
            const QStringList & args,
            QString * process_out,
            QString * process_err = Q_NULLPTR )  [static]
```

Executor::executeBlocking blocking version of the executor.

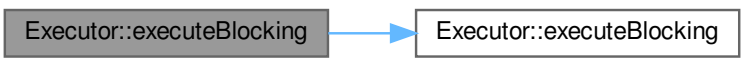
Parameters

app	
args	
process_out	
process_err	

Returns

Definition at line 213 of file executor.cpp.

Here is the call graph for this function:



13.5.3.8 executeBlocking() [2/2]

```
int Executor::executeBlocking (
            QString app,
            const QStringList & args,
            QString input = QString(),
            QString * process_out = Q_NULLPTR,
            QString * process_err = Q_NULLPTR )  [static]
```

Executor::executeBlocking blocking version of the executor, takes input and presents it as stdin.

Parameters

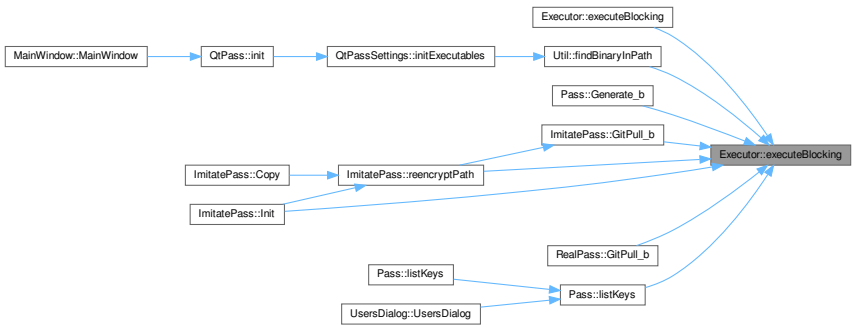
app	
args	
input	
process_out	
process_err	

Returns

TODO(bezet): it might make sense to throw here, a lot of possible errors

Definition at line 171 of file executor.cpp.

Here is the caller graph for this function:



13.5.3.9 finished

```
void Executor::finished (
            int id,
            int exitCode,
            const QString & output,
            const QString & errout )  [signal]
```

finished signal that is emited when process finishes

Parameters

id	id of the process
exitCode	return code of the process
output	stdout produced by the process
errout	stderr produced by the process

13.5.3.10 setEnvironment()

```
void Executor::setEnvironment (
            const QStringList & env )
```

Executor::setEnvironment set environment variables for executor processes.

Parameters

env	

Definition at line 223 of file executor.cpp.

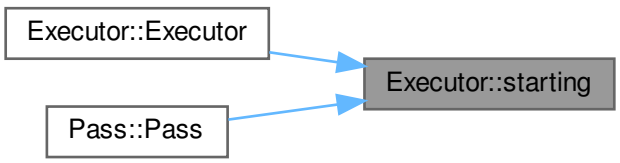
Here is the caller graph for this function:

13.5.3.11 starting

```
void Executor::starting ( )  [signal]
```

starting signal that is emited when process starts

Here is the caller graph for this function:



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

- src/executor.h
- src/executor.cpp

13.6 FileContent Class Reference

`#include <filecontent.h>`

Public Member Functions

- QString getPassword () const
- NamedValues getNamedValues () const
- QString getRemainingData () const
- QString getRemainingDataForDisplay () const

Static Public Member Functions

- static FileContent parse (const QString &fileContent, const QStringList &templateFields, bool allFields)

  parse parses the given fileContent in a FileContent object. The password is accessible through getPassword. The named value pairs (name: value) are parsed and depeding on the templateFields and allFields parameters accessible through getNamedValues, getRemainingData or getRemainingDataForDisplay.

13.6.1 Detailed Description

Definition at line 25 of file filecontent.h.

13.6.2 Member Function Documentation

13.6.2.1 getNamedValues()

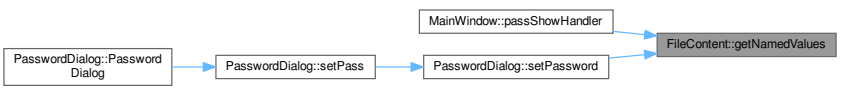
```
NamedValues FileContent::getNamedValues ( ) const
```

Returns

the named values in the file in the order of appearence, with template values first.

Definition at line 38 of file filecontent.cpp.

Here is the caller graph for this function:



13.6.2.2 getPassword()

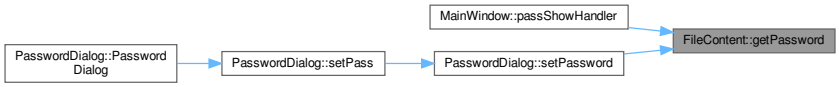
```
QString FileContent::getPassword ( ) const
```

Returns

the password from the parsed file.

Definition at line 36 of file filecontent.cpp.

Here is the caller graph for this function:



13.6.2.3 getRemainingData()

```
QString FileContent::getRemainingData ( ) const
```

Returns

the data that is not the password and not in getNamedValues.

Definition at line 40 of file filecontent.cpp.

Here is the caller graph for this function:

13.6.2.4 getRemainingDataForDisplay()

```
QString FileContent::getRemainingDataForDisplay ( ) const
```

@like getRemainingData but without data that should not be displayed (like a TOTP secret).

Definition at line 42 of file filecontent.cpp.

Here is the caller graph for this function:

13.6.2.5 parse()

```
FileContent FileContent::parse (
            const QString & fileContent,
            const QStringList & templateFields,
            bool allFields )  [static]
```

parse parses the given fileContent in a FileContent object. The password is accessible through getPassword. The named value pairs (name: value) are parsed and depeding on the templateFields and allFields parameters accessible through getNamedValues, getRemainingData or getRemainingDataForDisplay.

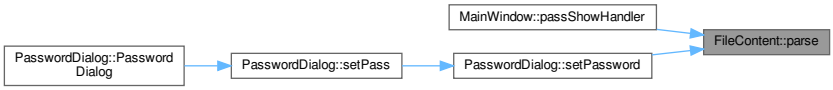
Parameters

fileContent	the file content to parse.
templateFields	the fields in the template. Fields in the template will always be in getNamedValues() at the beginning of the list in the same order.
allFields	whether all fields should be considered as named values. If set to false only templateFields are returned in getNamedValues().

Returns

Definition at line 7 of file filecontent.cpp.

Here is the caller graph for this function:



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

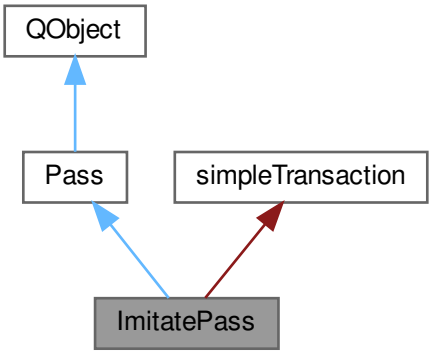
- src/filecontent.h
- src/filecontent.cpp

13.7 ImitatePass Class Reference

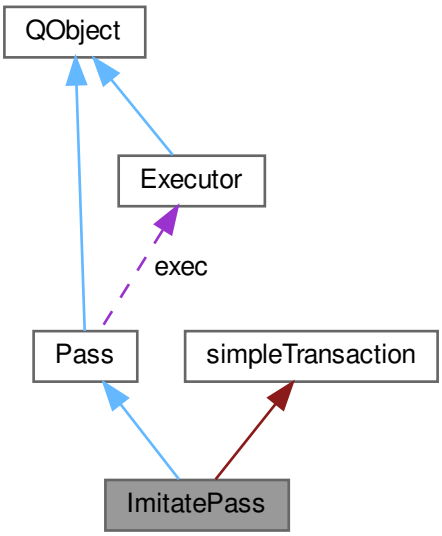
Imitates pass features when pass is not enabled or available.

```
#include <imitatepass.h>
```

Inheritance diagram for ImitatePass:



Collaboration diagram for ImitatePass:



Signals

- void startReencryptPath ()
- void endReencryptPath ()

Signals inherited from Pass

- void error (QProcess::ProcessError)
- void startingExecuteWrapper ()
- void statusMsg (QString, int)
- void critical (QString, QString)
- void processErrorExit (int exitCode, const QString &err)
- void finishedAny (const QString &, const QString &)
- void finishedGitInit (const QString &, const QString &)
- void finishedGitPull (const QString &, const QString &)
- void finishedGitPush (const QString &, const QString &)
- void finishedShow (const QString &)
- void finishedOtpGenerate (const QString &)
- void finishedInsert (const QString &, const QString &)
- void finishedRemove (const QString &, const QString &)
- void finishedInit (const QString &, const QString &)
- void finishedMove (const QString &, const QString &)
- void finishedCopy (const QString &, const QString &)
- void finishedGenerate (const QString &, const QString &)
- void finishedGenerateGPGKeys (const QString &, const QString &)

Public Member Functions

- ImitatePass ()

  ImitatePass::ImitatePass for situaions when pass is not available we imitate the behavior of pass `https://www.passwordstore.org/`.
- virtual ~ImitatePass ()
- virtual void GitInit () Q_DECL_OVERRIDE

  ImitatePass::GitInit git init wrapper.
- virtual void GitPull () Q_DECL_OVERRIDE

  ImitatePass::GitPull git init wrapper.
- virtual void GitPull_b () Q_DECL_OVERRIDE

  ImitatePass::GitPull_b git pull wrapper.
- virtual void GitPush () Q_DECL_OVERRIDE

  ImitatePass::GitPush git init wrapper.
- virtual void Show (QString file) Q_DECL_OVERRIDE

  ImitatePass::Show shows content of file.
- virtual void OtpGenerate (QString file) Q_DECL_OVERRIDE

  ImitatePass::OtpGenerate generates an otp code.
- virtual void Insert (QString file, QString newValue, bool overwrite=false) Q_DECL_OVERRIDE

  ImitatePass::Insert create new file with encrypted content.
- virtual void Remove (QString file, bool isDir=false) Q_DECL_OVERRIDE

  ImitatePass::Remove custom implementation of "pass remove".
- virtual void Init (QString path, const QList< UserInfo > &users) Q_DECL_OVERRIDE

  ImitatePass::Init initialize pass repository.
- void reencryptPath (const QString &dir)

  ImitatePass::reencryptPath reencrypt all files under the chosen directory.
- void Move (const QString src, const QString dest, const bool force=false) Q_DECL_OVERRIDE
- void Copy (const QString src, const QString dest, const bool force=false) Q_DECL_OVERRIDE

Public Member Functions inherited from Pass

- Pass ()

  Pass::Pass wrapper for using either pass or the pass imitation.
- void init ()
- virtual ~Pass ()
- virtual void GitInit ()=0
- virtual void GitPull ()=0
- virtual void GitPull_b ()=0
- virtual void GitPush ()=0
- virtual void Show (QString file)=0
- virtual void OtpGenerate (QString file)=0
- virtual void Insert (QString file, QString value, bool force)=0
- virtual void Remove (QString file, bool isDir)=0
- virtual void Move (const QString srcDir, const QString dest, const bool force=false)=0
- virtual void Copy (const QString srcDir, const QString dest, const bool force=false)=0
- virtual void Init (QString path, const QList< UserInfo > &users)=0
- virtual QString Generate_b (unsigned int length, const QString &charset)

  Pass::Generate use either pwgen or internal password generator.
- void GenerateGPGKeys (QString batch)

  Pass::GenerateGPGKeys internal gpg keypair generator . .
- QList< UserInfo > listKeys (QStringList keystrings, bool secret=false)

  Pass::listKeys list users.
- QList< UserInfo > listKeys (QString keystring="", bool secret=false)

  Pass::listKeys list users.
- void updateEnv ()

  Pass::updateEnv update the execution environment (used when switching profiles)

Protected Member Functions

- virtual void finished (int id, int exitCode, const QString &out, const QString &err) Q_DECL_OVERRIDE

  ImitatePass::finished this function is overloaded to ensure identical behaviour to RealPass ie. only PASS_∗ processes are visible inside Pass::finish, so that interface-wise it all looks the same.
- virtual void executeWrapper (PROCESS id, const QString &app, const QStringList &args, QString input, bool readStdout=true, bool readStderr=true) Q_DECL_OVERRIDE

  executeWrapper overrided so that every execution is a transaction

Protected Member Functions inherited from Pass

- void executeWrapper (PROCESS id, const QString &app, const QStringList &args, bool readStdout=true, bool readStderr=true)
- QString generateRandomPassword (const QString &charset, unsigned int length)
- quint32 boundedRandom (quint32 bound)
- virtual void executeWrapper (PROCESS id, const QString &app, const QStringList &args, QString input, bool readStdout=true, bool readStderr=true)

Additional Inherited Members

Static Public Member Functions inherited from Pass

- static QString getGpgIdPath (QString for_file)

  Pass::getGpgIdPath return gpgid file path for some file (folder).
- static QStringList getRecipientList (QString for_file)

  Pass::getRecipientList return list of gpg-id's to encrypt for.
- static QStringList getRecipientString (QString for_file, QString separator=" ", int *count=NULL)

  Pass::getRecipientString formated string for use with GPG.

Protected Types inherited from Pass

- typedef Enums::PROCESS PROCESS

Protected Slots inherited from Pass

- virtual void finished (int id, int exitCode, const QString &out, const QString &err)

  Pass::processFinished reemits specific signal based on what process has finished.

Protected Attributes inherited from Pass

- Executor exec

13.7.1 Detailed Description

Imitates pass features when pass is not enabled or available.

Definition at line 11 of file imitatepass.h.

13.7.2 Constructor & Destructor Documentation

13.7.2.1 ImitatePass()

```
ImitatePass::ImitatePass ( )  [default]
```

ImitatePass::ImitatePass for situaions when pass is not available we imitate the behavior of pass https://www.passwordstore.org/.

13.7.2.2 ~ImitatePass()

```
virtual ImitatePass::~ImitatePass ( )  [inline], [virtual]
```

Definition at line 49 of file imitatepass.h.

13.7.3 Member Function Documentation

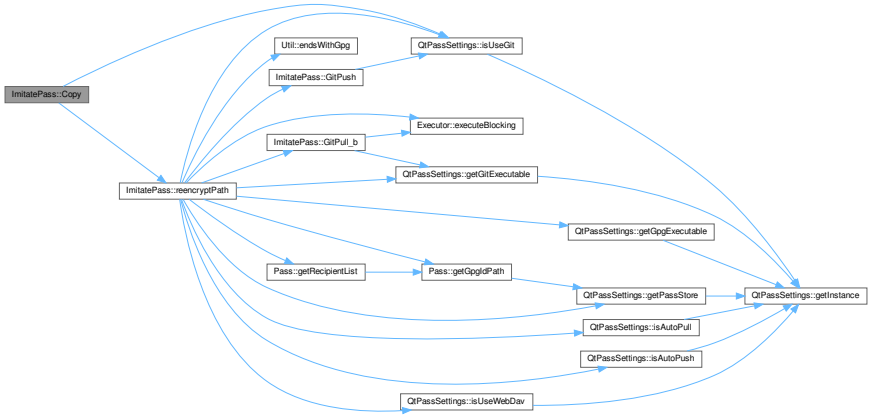
13.7.3.1 Copy()

```
void ImitatePass::Copy (
            const QString src,
            const QString dest,
            const bool force = false )  [virtual]
```

Implements Pass.

Definition at line 524 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.2 endReencryptPath

```
void ImitatePass::endReencryptPath ( )  [signal]
```

Here is the caller graph for this function:

13.7.3.3 executeWrapper()

```
void ImitatePass::executeWrapper (
            PROCESS id,
            const QString & app,
            const QStringList & args,
            QString input,
            bool readStdout = true,
            bool readStderr = true )  [protected], [virtual]
```

executeWrapper overrided so that every execution is a transaction

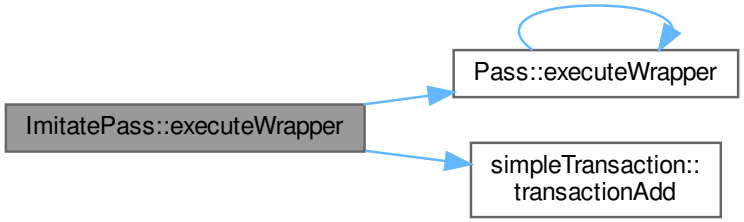
Parameters

id	
app	
args	
input	
readStdout	
readStderr	

Reimplemented from Pass.

Definition at line 623 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.4 finished()

```
void ImitatePass::finished (
            int id,
            int exitCode,
            const QString & out,
            const QString & err )  [protected], [virtual]
```

ImitatePass::finished this function is overloaded to ensure identical behaviour to RealPass ie. only PASS_∗ processes are visible inside Pass::finish, so that interface-wise it all looks the same.

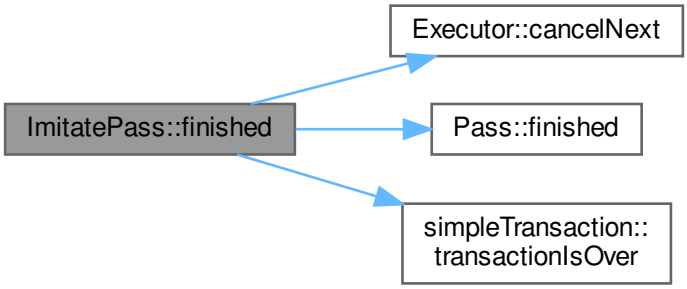
Parameters

id	
exitCode	
out	
err	

Reimplemented from Pass.

Definition at line 585 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.5 GitInit()

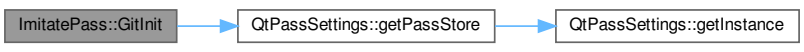
```
void ImitatePass::GitInit ( )  [virtual]
```

ImitatePass::GitInit git init wrapper.

Implements Pass.

Definition at line 37 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.6 GitPull()

```
void ImitatePass::GitPull ( )  [virtual]
```

ImitatePass::GitPull git init wrapper.

Implements Pass.

Definition at line 44 of file imitatepass.cpp.

13.7.3.7 GitPull_b()

```
void ImitatePass::GitPull_b ( )  [virtual]
```

ImitatePass::GitPull_b git pull wrapper.

Implements Pass.

Definition at line 49 of file imitatepass.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

13.7.3.8 GitPush()

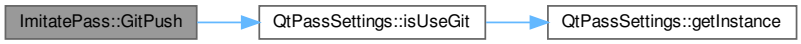
```
void ImitatePass::GitPush ( )  [virtual]
```

ImitatePass::GitPush git init wrapper.

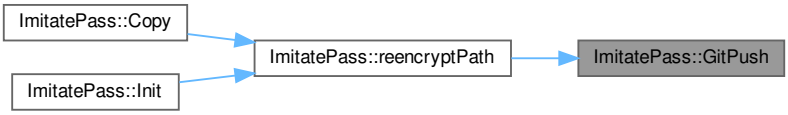
Implements Pass.

Definition at line 56 of file imitatepass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.7.3.9 Init()

```
void ImitatePass::Init (
            QString path,
            const QList< UserInfo > & users )  [virtual]
```

ImitatePass::Init initialize pass repository.

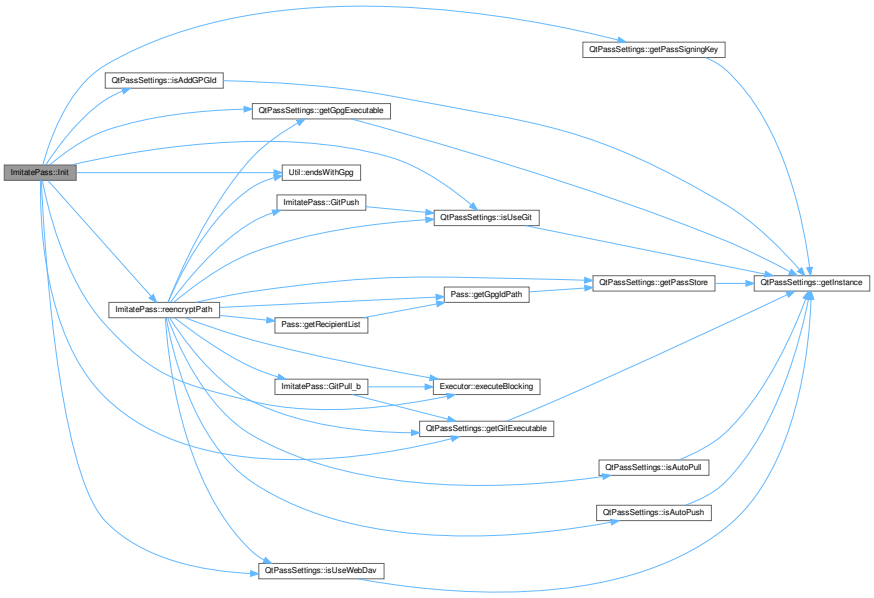
Parameters

path	path in which new password-store will be created
users	list of users who shall be able to decrypt passwords in path

Implements Pass.

Definition at line 171 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.10 Insert()

```
void ImitatePass::Insert (
            QString file,
            QString newValue,
            bool overwrite = false )  [virtual]
```

ImitatePass::Insert create new file with encrypted content.

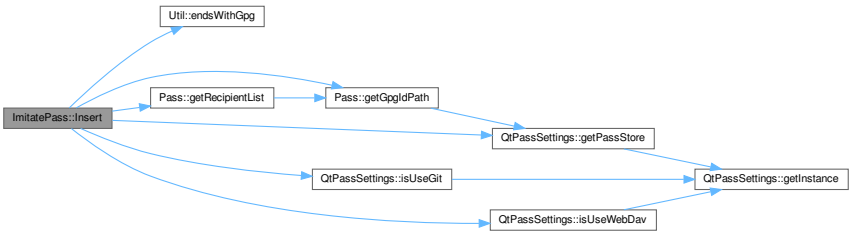
Parameters

file	file to be created
newValue	value to be stored in file
overwrite	whether to overwrite existing file

Implements Pass.

Definition at line 91 of file imitatepass.cpp.

Here is the call graph for this function:



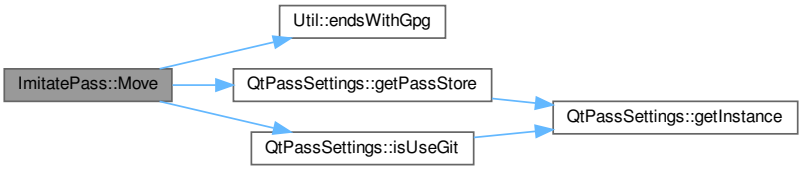
13.7.3.11 Move()

```
void ImitatePass::Move (
            const QString src,
            const QString dest,
            const bool force = false )  [virtual]
```

Implements Pass.

Definition at line 449 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.12 OtpGenerate()

```
void ImitatePass::OtpGenerate (
            QString file )  [virtual]
```

ImitatePass::OtpGenerate generates an otp code.

Implements Pass.

Definition at line 75 of file imitatepass.cpp.

13.7.3.13 reencryptPath()

```
void ImitatePass::reencryptPath (
            const QString & dir )
```

ImitatePass::reencryptPath reencrypt all files under the chosen directory.

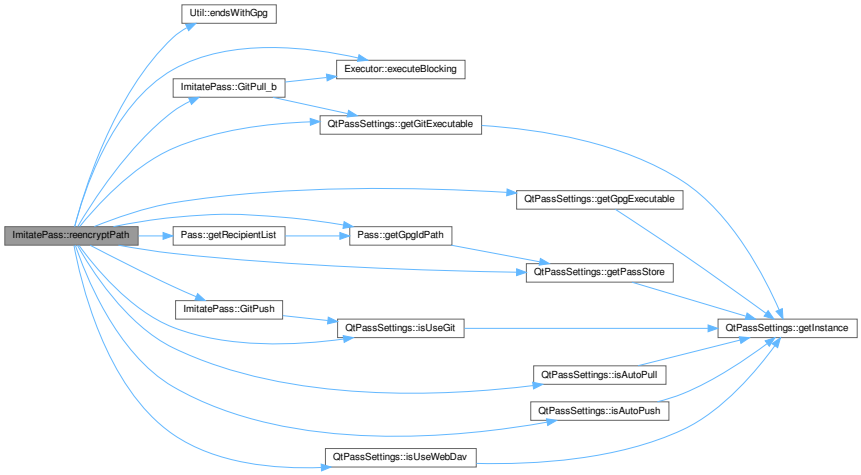
This is stil quite experimental..

Parameters

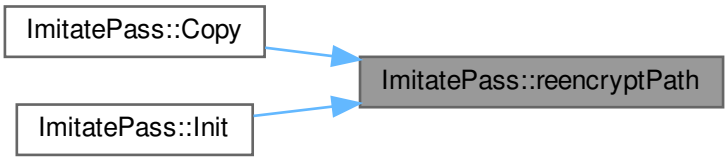
dir	

Definition at line 334 of file imitatepass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.7.3.14 Remove()

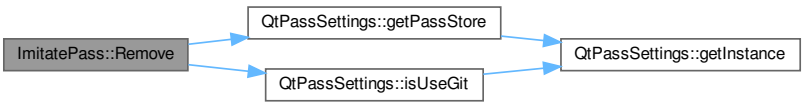
```
void ImitatePass::Remove (
            QString file,
            bool isDir = false )  [virtual]
```

ImitatePass::Remove custom implementation of "pass remove".

Implements Pass.

Definition at line 145 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.15 Show()

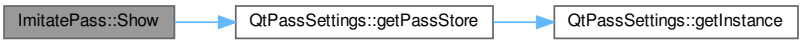
```
void ImitatePass::Show (
            QString file )  [virtual]
```

ImitatePass::Show shows content of file.

Implements Pass.

Definition at line 65 of file imitatepass.cpp.

Here is the call graph for this function:



13.7.3.16 startReencryptPath

```
void ImitatePass::startReencryptPath ( )  [signal]
```

Here is the caller graph for this function:

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

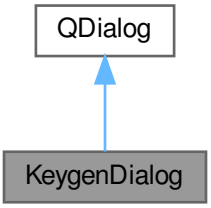
- src/imitatepass.h
- src/imitatepass.cpp

13.8 KeygenDialog Class Reference

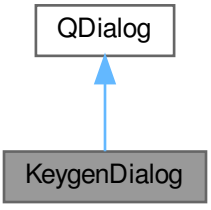
Handles GPG keypair generation.

`#include <keygendialog.h>`

Inheritance diagram for KeygenDialog:



Collaboration diagram for KeygenDialog:



Public Member Functions

- KeygenDialog (ConfigDialog *parent=0)
  KeygenDialog::KeygenDialog basic constructor.
- ~KeygenDialog ()
  KeygenDialog::~KeygenDialog even more basic destructor.

Protected Member Functions

- void closeEvent (QCloseEvent *event)
  KeygenDialog::closeEvent we are done here.

13.8.1 Detailed Description

Handles GPG keypair generation.

Definition at line 16 of file keygendialog.h.

13.8.2 Constructor & Destructor Documentation

13.8.2.1 KeygenDialog()

```
KeygenDialog::KeygenDialog (
            ConfigDialog * parent = 0 )  [explicit]
```

KeygenDialog::KeygenDialog basic constructor.

Parameters

parent	

Definition at line 16 of file keygendialog.cpp.

13.8.2.2 ~KeygenDialog()

```
KeygenDialog::~KeygenDialog ( )
```

KeygenDialog::~KeygenDialog even more basic destructor.

Definition at line 25 of file keygendialog.cpp.

13.8.3 Member Function Documentation

13.8.3.1 closeEvent()

```
void KeygenDialog::closeEvent (
            QCloseEvent * event )  [protected]
```

KeygenDialog::closeEvent we are done here.

Parameters

event	

Definition at line 188 of file keygendialog.cpp.

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

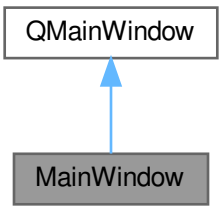
- src/keygendialog.h
- src/keygendialog.cpp

13.9 MainWindow Class Reference

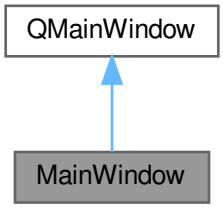
The MainWindow class does way too much, not only is it a switchboard, configuration handler and more, it's also the process-manager.

```
#include <mainwindow.h>
```

Inheritance diagram for MainWindow:



Collaboration diagram for MainWindow:



Public Slots

- void deselect ()

  MainWindow::deselect clear the selection, password and copy buffer.
- void messageAvailable (QString message)

  MainWindow::messageAvailable we have some text/message/search to do.
- void critical (QString, QString)

  MainWindow::critical critical message popup wrapper.
- void executeWrapperStarted ()
- void showStatusMessage (QString msg, int timeout=2000)

  Displays message in status bar.
- void passShowHandler (const QString &)
- void passOtpHandler (const QString &)
- void onPush ()

  MainWindow::onPush do a git push.
- void on_treeView_clicked (const QModelIndex &index)

  MainWindow::on_treeView_clicked read the selected password file.
- void startReencryptPath ()

  MainWindow::startReencryptPath disable ui elements and treeview.
- void endReencryptPath ()

  MainWindow::endReencryptPath re-enable ui elements.

Signals

- void passShowHandlerFinished (QString output)
- void passGitInitNeeded ()
- void generateGPGKeyPair (QString batch)

Public Member Functions

- MainWindow (const QString &searchText=QString(), QWidget ∗parent=nullptr)

  MainWindow::MainWindow handles all of the main functionality and also the main window.
- ∼MainWindow ()
- void restoreWindow ()
- void generateKeyPair (QString, QDialog ∗)

  MainWindow::generateKeyPair internal gpg keypair generator . .
- void userDialog (QString="")

  MainWindow::userDialog see MainWindow::onUsers()
- void config ()

  MainWindow::config pops up the configuration screen and handles all inter-window communication.
- void setUiElementsEnabled (bool state)

  MainWindow::setUiElementsEnabled enable or disable the relevant UI elements.
- void flashText (const QString &text, const bool isError, const bool isHtml=false)
- const QModelIndex getCurrentTreeViewIndex ()
- QDialog ∗ getKeygenDialog ()
- void cleanKeygenDialog ()

Protected Member Functions

- void closeEvent (QCloseEvent ∗event)

  MainWindow::closeEvent hide or quit.
- void keyPressEvent (QKeyEvent ∗event)

  MainWindow::keyPressEvent did anyone press return, enter or escape?
- void changeEvent (QEvent ∗event)

  MainWindow::changeEvent sets focus to the search box.
- bool eventFilter (QObject ∗obj, QEvent ∗event)

  MainWindow::eventFilter filter out some events and focus the treeview.

13.9.1 Detailed Description

The MainWindow class does way too much, not only is it a switchboard, configuration handler and more, it's also the process-manager.

This class could really do with an overhaul.

Definition at line 37 of file mainwindow.h.

13.9.2 Constructor & Destructor Documentation

13.9.2.1 MainWindow()

```
MainWindow::MainWindow (
            const QString & searchText = QString(),
            QWidget * parent = nullptr )  [explicit]
```

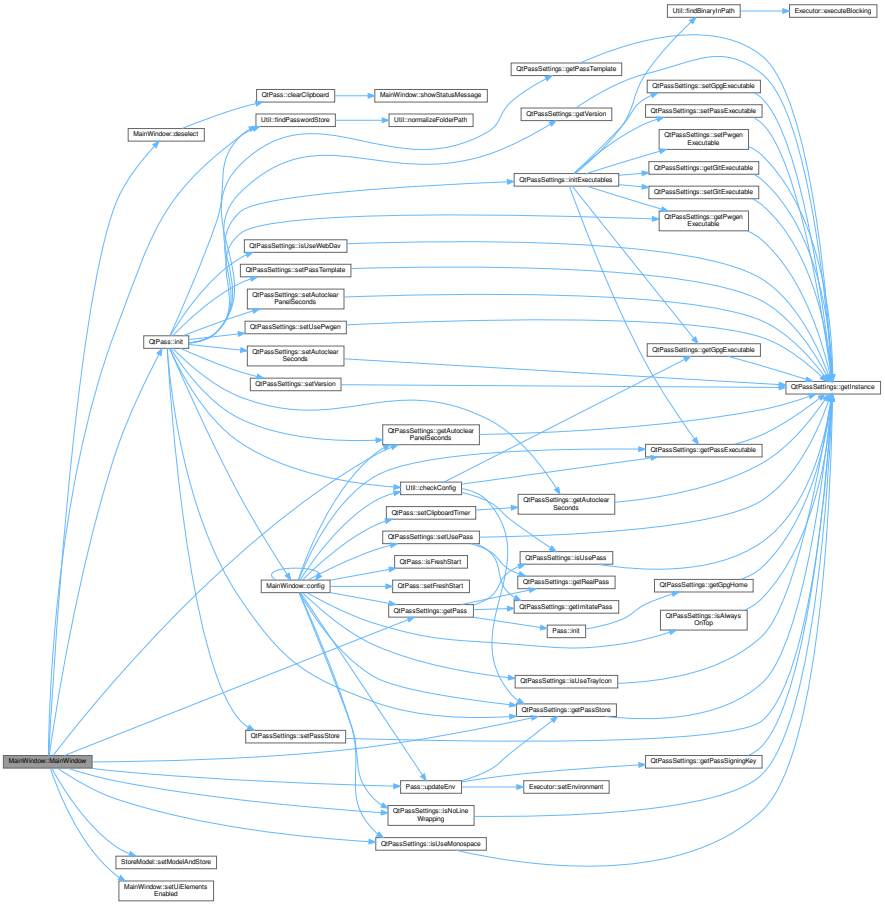
MainWindow::MainWindow handles all of the main functionality and also the main window.

Parameters

searchText	for searching from cli
parent	pointer

Definition at line 36 of file mainwindow.cpp.

Here is the call graph for this function:



13.9.2.2 ~MainWindow()

```
MainWindow::~MainWindow ( )
```

Definition at line 128 of file mainwindow.cpp.

13.9.3 Member Function Documentation

13.9.3.1 changeEvent()

```
void MainWindow::changeEvent (
            QEvent * event )  [protected]
```

MainWindow::changeEvent sets focus to the search box.

Parameters

event	

Definition at line 145 of file mainwindow.cpp.

13.9.3.2 cleanKeygenDialog()

```
void MainWindow::cleanKeygenDialog ( )
```

Definition at line 205 of file mainwindow.cpp.

13.9.3.3 closeEvent()

```
void MainWindow::closeEvent (
            QCloseEvent * event )  [protected]
```

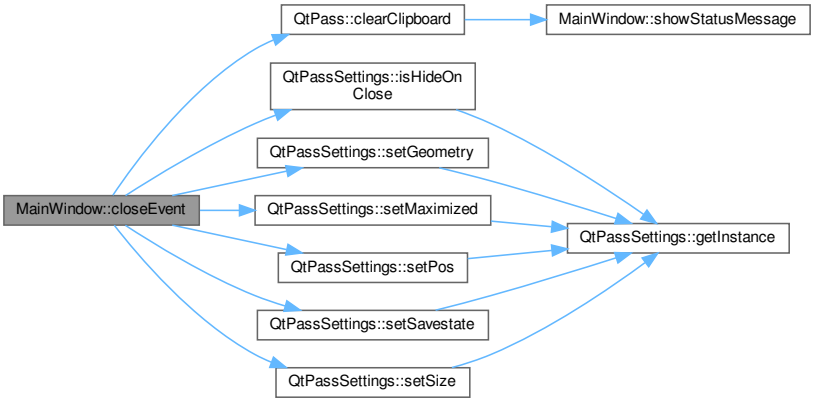
MainWindow::closeEvent hide or quit.

Parameters

event	

Definition at line 846 of file mainwindow.cpp.

Here is the call graph for this function:



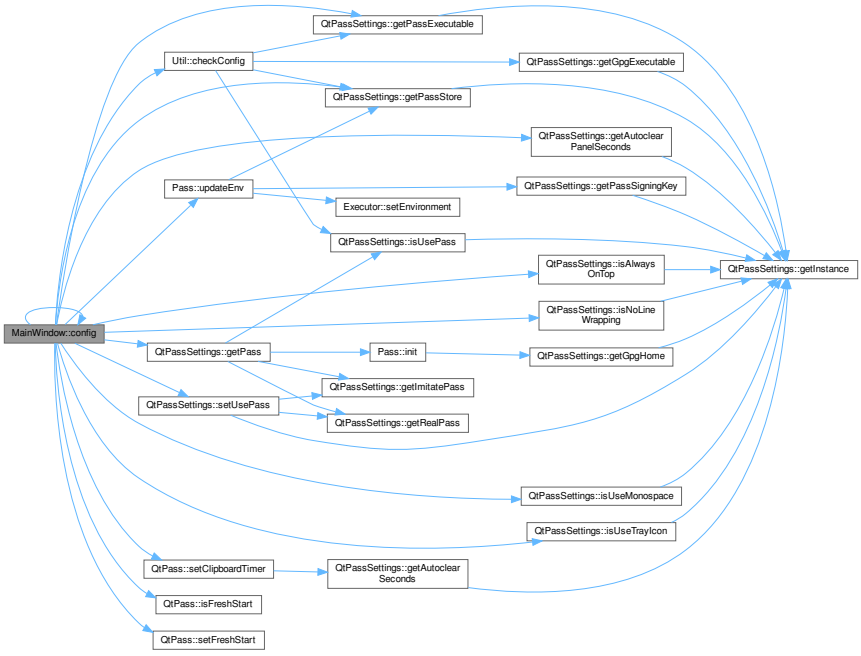
13.9.3.4 config()

```
void MainWindow::config ( )
```

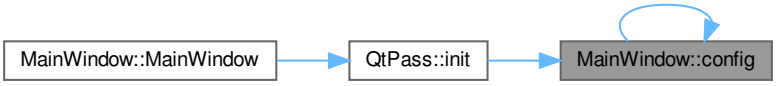
MainWindow::config pops up the configuration screen and handles all inter-window communication.

Definition at line 230 of file mainwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.9.3.5 critical

```
void MainWindow::critical (
            QString title,
            QString msg )  [slot]
```

MainWindow::critical critical message popup wrapper.

Parameters

title	
msg	

Definition at line 1228 of file mainwindow.cpp.

13.9.3.6 deselect

```
void MainWindow::deselect ( )  [slot]
```

MainWindow::deselect clear the selection, password and copy buffer.

Definition at line 366 of file mainwindow.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

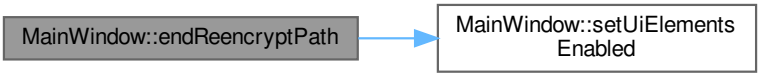
13.9.3.7 endReencryptPath

```
void MainWindow::endReencryptPath ( )  [slot]
```

MainWindow::endReencryptPath re-enable ui elements.

Definition at line 1195 of file mainwindow.cpp.

Here is the call graph for this function:



13.9.3.8 eventFilter()

```
bool MainWindow::eventFilter (
            QObject * obj,
            QEvent * event )  [protected]
```

MainWindow::eventFilter filter out some events and focus the treeview.

Parameters

obj	
event	

Returns

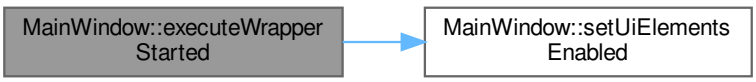
Definition at line 871 of file mainwindow.cpp.

13.9.3.9 executeWrapperStarted

```
void MainWindow::executeWrapperStarted ( )  [slot]
```

Definition at line 376 of file mainwindow.cpp.

Here is the call graph for this function:



13.9.3.10 flashText()

```
void MainWindow::flashText (
            const QString & text,
            const bool isError,
            const bool isHtml = false )
```

Definition at line 210 of file mainwindow.cpp.

13.9.3.11 generateGPGKeyPair

```
void MainWindow::generateGPGKeyPair (
            QString batch )  [signal]
```

Here is the caller graph for this function:

13.9.3.12 generateKeyPair()

```
void MainWindow::generateKeyPair (
            QString batch,
            QDialog * keygenWindow )
```

MainWindow::generateKeyPair internal gpg keypair generator . .

Parameters

batch	
keygenWindow	

Definition at line 753 of file mainwindow.cpp.

Here is the caller graph for this function:

13.9.3.13 getCurrentTreeViewIndex()

```
const QModelIndex MainWindow::getCurrentTreeViewIndex ( )
```

Definition at line 201 of file mainwindow.cpp.

13.9.3.14 getKeygenDialog()

```
QDialog * MainWindow::getKeygenDialog ( )  [inline]
```

Definition at line 56 of file mainwindow.h.

13.9.3.15 keyPressEvent()

```
void MainWindow::keyPressEvent (
            QKeyEvent * event )  [protected]
```

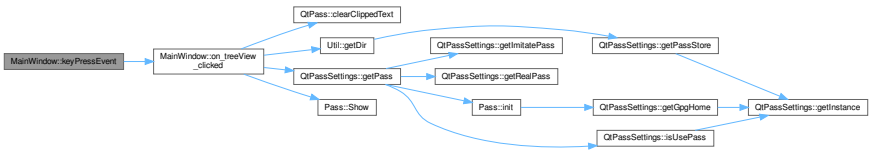
MainWindow::keyPressEvent did anyone press return, enter or escape?

Parameters

event	

Definition at line 885 of file mainwindow.cpp.

Here is the call graph for this function:



13.9.3.16 messageAvailable

```
void MainWindow::messageAvailable (
            QString message )  [slot]
```

MainWindow::messageAvailable we have some text/message/search to do.

Parameters

message	

Definition at line 736 of file mainwindow.cpp.

Here is the caller graph for this function:

13.9.3.17 on_treeView_clicked

```
void MainWindow::on_treeView_clicked (
            const QModelIndex & index )  [slot]
```

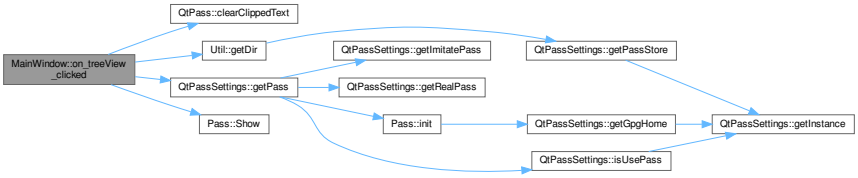
MainWindow::on_treeView_clicked read the selected password file.

Parameters

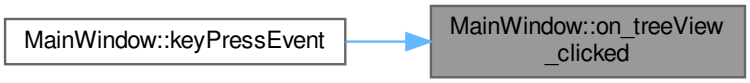
index	

Definition at line 332 of file mainwindow.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



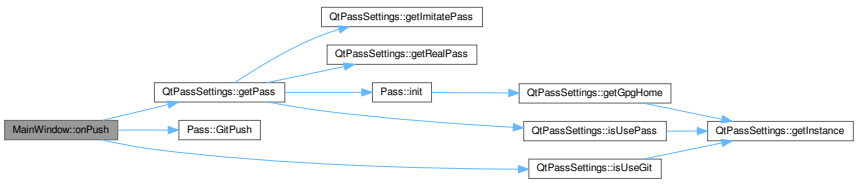
13.9.3.18 onPush

```
void MainWindow::onPush ( )  [slot]
```

MainWindow::onPush do a git push.

Definition at line 302 of file mainwindow.cpp.

Here is the call graph for this function:



13.9.3.19 passGitInitNeeded

```
void MainWindow::passGitInitNeeded ( )  [signal]
```

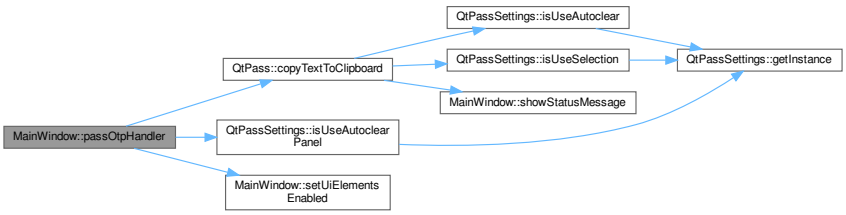
Here is the caller graph for this function:

13.9.3.20 passOtpHandler

```
void MainWindow::passOtpHandler (
            const QString & p_output )  [slot]
```

Definition at line 429 of file mainwindow.cpp.

Here is the call graph for this function:

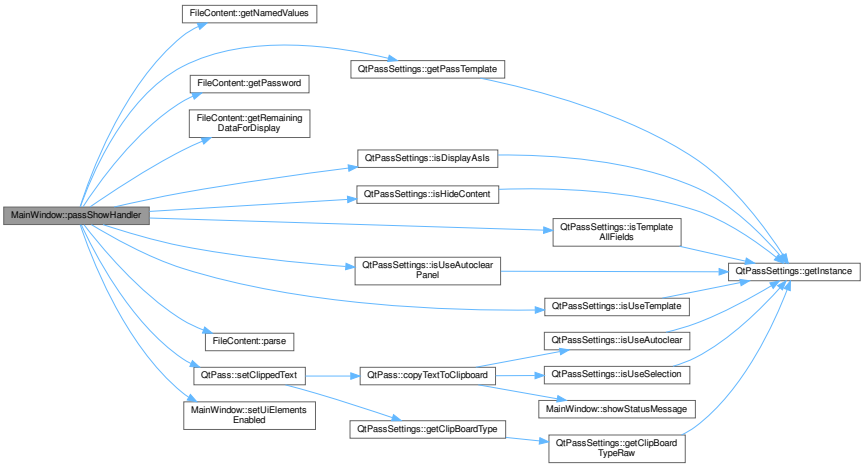


13.9.3.21 passShowHandler

```
void MainWindow::passShowHandler (
            const QString & p_output )  [slot]
```

Definition at line 383 of file mainwindow.cpp.

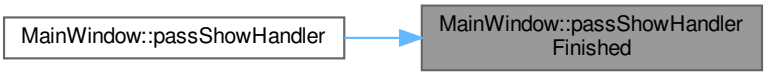
Here is the call graph for this function:



13.9.3.22 passShowHandlerFinished

```
void MainWindow::passShowHandlerFinished (
            QString output )  [signal]
```

Here is the caller graph for this function:

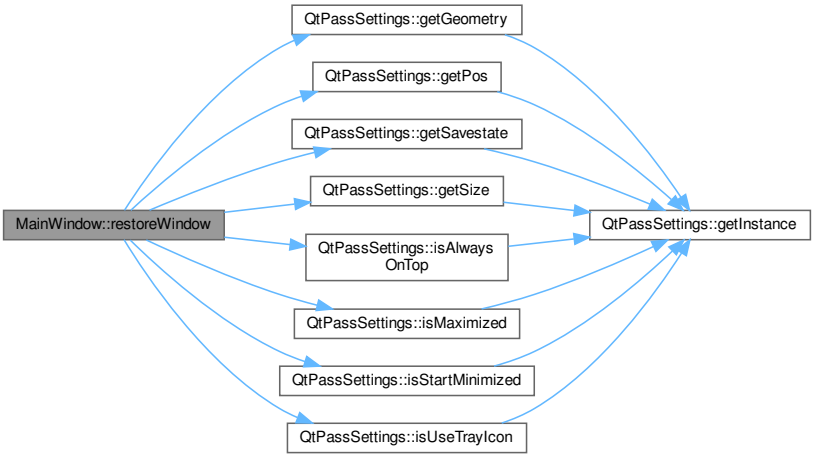


13.9.3.23 restoreWindow()

```
void MainWindow::restoreWindow ( )
```

Definition at line 478 of file mainwindow.cpp.

Here is the call graph for this function:



13.9.3.24 setUiElementsEnabled()

```
void MainWindow::setUiElementsEnabled (
            bool state )
```

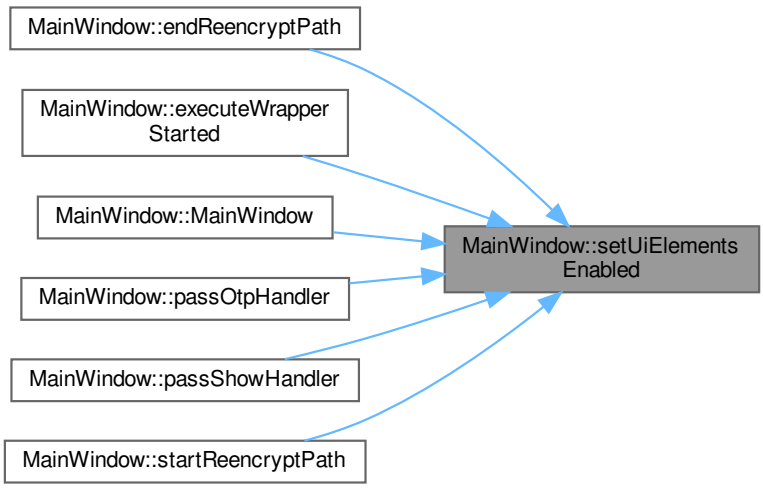
MainWindow::setUiElementsEnabled enable or disable the relevant UI elements.

Parameters

state	

Definition at line 462 of file mainwindow.cpp.

Here is the caller graph for this function:



13.9.3.25 showStatusMessage

```
void MainWindow::showStatusMessage (
            QString msg,
            int timeout = 2000 )  [slot]
```

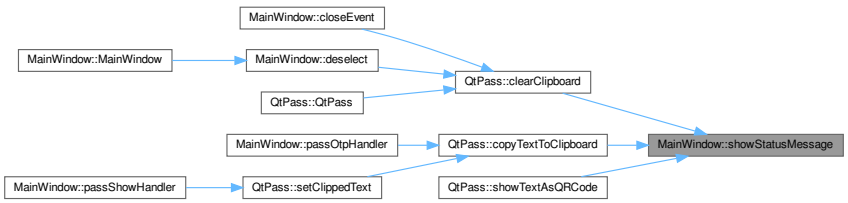
Displays message in status bar.

Parameters

msg	text to be displayed
timeout	time for which msg shall be visible

Definition at line 1180 of file mainwindow.cpp.

Here is the caller graph for this function:



13.9.3.26 startReencryptPath

```
void MainWindow::startReencryptPath ( )  [slot]
```

MainWindow::startReencryptPath disable ui elements and treeview.

Definition at line 1187 of file mainwindow.cpp.

Here is the call graph for this function:

13.9.3.27 userDialog()

```
void MainWindow::userDialog (
            QString dir = "" )
```

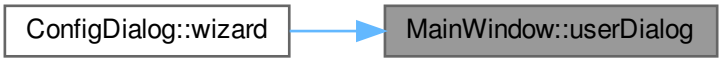
MainWindow::userDialog see MainWindow::onUsers()

Parameters

dir	folder to edit users for.

Definition at line 709 of file mainwindow.cpp.

Here is the caller graph for this function:



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

- src/mainwindow.h
- src/mainwindow.cpp

13.10 NamedValue Struct Reference

`#include <filecontent.h>`

Public Attributes

- QString name
- QString value

13.10.1 Detailed Description

Definition at line 8 of file filecontent.h.

13.10.2 Member Data Documentation

13.10.2.1 name

`QString NamedValue::name`

Definition at line 9 of file filecontent.h.

13.10.2.2 value

`QString NamedValue::value`

Definition at line 10 of file filecontent.h.

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

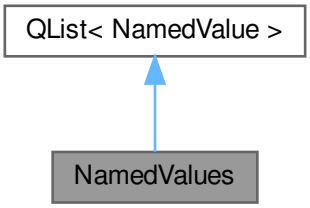
- src/filecontent.h

13.11 NamedValues Class Reference

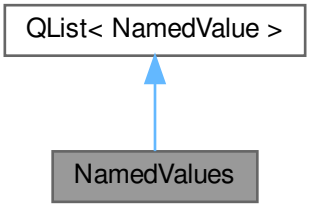
The NamedValues class is mostly a list of NamedValue but also has a method to take a specific NamedValue pair out of the list.

`#include <filecontent.h>`

Inheritance diagram for NamedValues:



Collaboration diagram for NamedValues:



Public Member Functions

- NamedValues ()
- NamedValues (std::initializer_list< NamedValue > values)
- QString takeValue (const QString &name)

13.11.1 Detailed Description

The NamedValues class is mostly a list of NamedValue but also has a method to take a specific NamedValue pair out of the list.

Definition at line 17 of file filecontent.h.

13.11.2 Constructor & Destructor Documentation

13.11.2.1 NamedValues() [1/2]

```
NamedValues::NamedValues ( )
```

Definition at line 54 of file filecontent.cpp.

13.11.2.2 NamedValues() [2/2]

```
NamedValues::NamedValues (
            std::initializer_list< NamedValue > values )
```

Definition at line 56 of file filecontent.cpp.

13.11.3 Member Function Documentation

13.11.3.1 takeValue()

```
QString NamedValues::takeValue (
            const QString & name )
```

Definition at line 59 of file filecontent.cpp.

Here is the caller graph for this function:

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

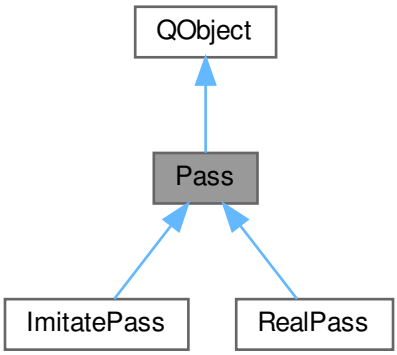
- src/filecontent.h
- src/filecontent.cpp

13.12 Pass Class Reference

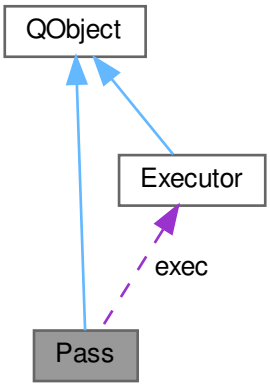
Acts as an abstraction for pass or pass imitation.

`#include <pass.h>`

Inheritance diagram for Pass:



Collaboration diagram for Pass:



Signals

- void error (QProcess::ProcessError)
- void startingExecuteWrapper ()
- void statusMsg (QString, int)

- void critical (QString, QString)
- void processErrorExit (int exitCode, const QString &err)
- void finishedAny (const QString &, const QString &)
- void finishedGitInit (const QString &, const QString &)
- void finishedGitPull (const QString &, const QString &)
- void finishedGitPush (const QString &, const QString &)
- void finishedShow (const QString &)
- void finishedOtpGenerate (const QString &)
- void finishedInsert (const QString &, const QString &)
- void finishedRemove (const QString &, const QString &)
- void finishedInit (const QString &, const QString &)
- void finishedMove (const QString &, const QString &)
- void finishedCopy (const QString &, const QString &)
- void finishedGenerate (const QString &, const QString &)
- void finishedGenerateGPGKeys (const QString &, const QString &)

Public Member Functions

- Pass ()

  Pass::Pass wrapper for using either pass or the pass imitation.
- void init ()
- virtual ~Pass ()
- virtual void GitInit ()=0
- virtual void GitPull ()=0
- virtual void GitPull_b ()=0
- virtual void GitPush ()=0
- virtual void Show (QString file)=0
- virtual void OtpGenerate (QString file)=0
- virtual void Insert (QString file, QString value, bool force)=0
- virtual void Remove (QString file, bool isDir)=0
- virtual void Move (const QString srcDir, const QString dest, const bool force=false)=0
- virtual void Copy (const QString srcDir, const QString dest, const bool force=false)=0
- virtual void Init (QString path, const QList< UserInfo > &users)=0
- virtual QString Generate_b (unsigned int length, const QString &charset)

  Pass::Generate use either pwgen or internal password generator.
- void GenerateGPGKeys (QString batch)

  Pass::GenerateGPGKeys internal gpg keypair generator . .
- QList< UserInfo > listKeys (QStringList keystrings, bool secret=false)

  Pass::listKeys list users.
- QList< UserInfo > listKeys (QString keystring="", bool secret=false)

  Pass::listKeys list users.
- void updateEnv ()

  Pass::updateEnv update the execution environment (used when switching profiles)

Static Public Member Functions

- static QString getGpgIdPath (QString for_file)

  Pass::getGpgIdPath return gpgid file path for some file (folder).
- static QStringList getRecipientList (QString for_file)

  Pass::getRecipientList return list of gpg-id's to encrypt for.
- static QStringList getRecipientString (QString for_file, QString separator=" ", int *count=NULL)

  Pass::getRecipientString formated string for use with GPG.

Protected Types

- typedef Enums::PROCESS PROCESS

Protected Slots

- virtual void finished (int id, int exitCode, const QString &out, const QString &err)

  Pass::processFinished reemits specific signal based on what process has finished.

Protected Member Functions

- void executeWrapper (PROCESS id, const QString &app, const QStringList &args, bool readStdout=true, bool readStderr=true)
- QString generateRandomPassword (const QString &charset, unsigned int length)
- quint32 boundedRandom (quint32 bound)
- virtual void executeWrapper (PROCESS id, const QString &app, const QStringList &args, QString input, bool readStdout=true, bool readStderr=true)

Protected Attributes

- Executor exec

13.12.1 Detailed Description

Acts as an abstraction for pass or pass imitation.

Definition at line 18 of file pass.h.

13.12.2 Member Typedef Documentation

13.12.2.1 PROCESS

```
typedef Enums::PROCESS Pass::PROCESS  [protected]
```

Definition at line 27 of file pass.h.

13.12.3 Constructor & Destructor Documentation

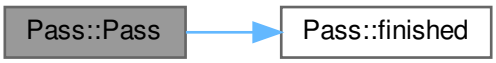
13.12.3.1 Pass()

```
Pass::Pass ( )
```

Pass::Pass wrapper for using either pass or the pass imitation.

Definition at line 17 of file pass.cpp.

Here is the call graph for this function:



13.12.3.2 ~Pass()

```
virtual Pass::~Pass ( )  [inline], [virtual]
```

Definition at line 33 of file pass.h.

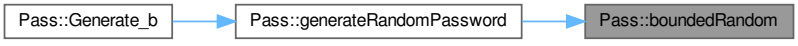
13.12.4 Member Function Documentation

13.12.4.1 boundedRandom()

```
quint32 Pass::boundedRandom (
            quint32 bound )  [protected]
```

Definition at line 349 of file pass.cpp.

Here is the caller graph for this function:



13.12.4.2 Copy()

```
virtual void Pass::Copy (
            const QString srcDir,
            const QString dest,
            const bool force = false )  [pure virtual]
```

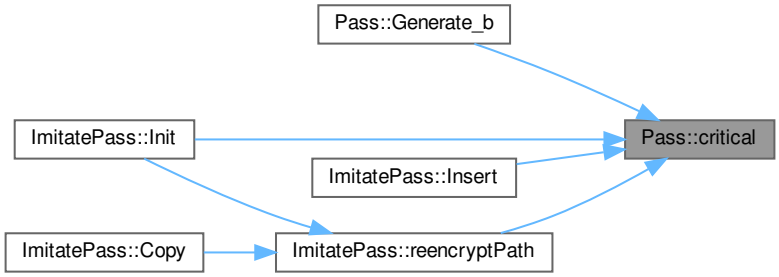
Implemented in ImitatePass, and RealPass.

Here is the caller graph for this function:

13.12.4.3 critical

```
void Pass::critical (
            QString ,
            QString  )  [signal]
```

Here is the caller graph for this function:



13.12.4.4 error

```
void Pass::error (
            QProcess::ProcessError  )  [signal]
```

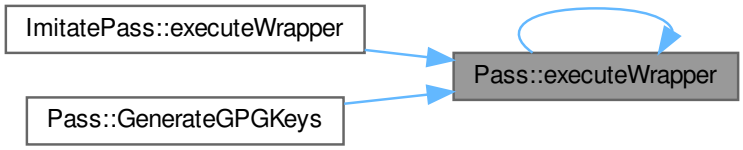
13.12.4.5 executeWrapper() [1/2]

```
void Pass::executeWrapper (
            PROCESS id,
            const QString & app,
            const QStringList & args,
            bool readStdout = true,
            bool readStderr = true )  [protected]
```

Definition at line 31 of file pass.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



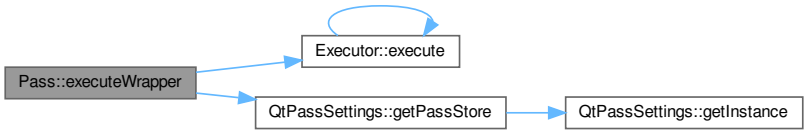
13.12.4.6 executeWrapper() [2/2]

```
void Pass::executeWrapper (
            PROCESS id,
            const QString & app,
            const QStringList & args,
            QString input,
            bool readStdout = true,
            bool readStderr = true )  [protected], [virtual]
```

Reimplemented in ImitatePass.

Definition at line 37 of file pass.cpp.

Here is the call graph for this function:



13.12.4.7 finished

```
void Pass::finished (
            int id,
            int exitCode,
            const QString & out,
            const QString & err )  [protected], [virtual], [slot]
```

Pass::processFinished reemits specific signal based on what process has finished.

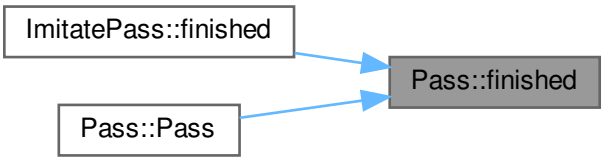
Parameters

id	id of Pass process that was scheduled and finished
exitCode	return code of a process
out	output generated by process(if capturing was requested, empty otherwise)
err	error output generated by process(if capturing was requested, or error occurred)

Reimplemented in ImitatePass.

Definition at line 195 of file pass.cpp.

Here is the caller graph for this function:



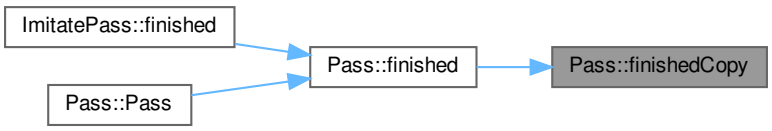
13.12.4.8 finishedAny

```
void Pass::finishedAny (
            const QString & ,
            const QString &  )  [signal]
```

13.12.4.9 finishedCopy

```
void Pass::finishedCopy (
            const QString & ,
            const QString &  )  [signal]
```

Here is the caller graph for this function:



13.12.4.10 finishedGenerate

```
void Pass::finishedGenerate (
            const QString & ,
            const QString &  )  [signal]
```

13.12.4.11 finishedGenerateGPGKeys

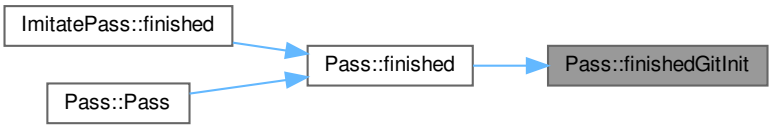
```
void Pass::finishedGenerateGPGKeys (
            const QString & ,
            const QString &  )  [signal]
```

Here is the caller graph for this function:

13.12.4.12 finishedGitInit

```
void Pass::finishedGitInit (
            const QString & ,
            const QString &  )  [signal]
```

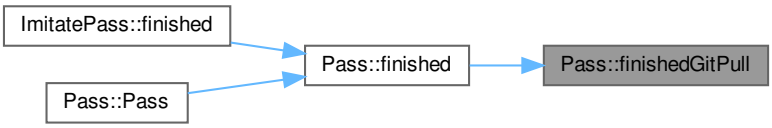
Here is the caller graph for this function:



13.12.4.13 finishedGitPull

```
void Pass::finishedGitPull (
            const QString & ,
            const QString &  )  [signal]
```

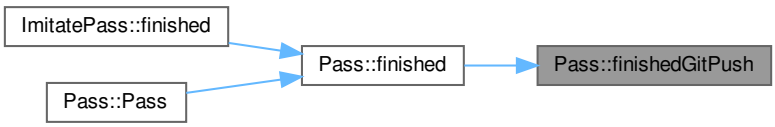
Here is the caller graph for this function:



13.12.4.14 finishedGitPush

```
void Pass::finishedGitPush (
            const QString & ,
            const QString &  )  [signal]
```

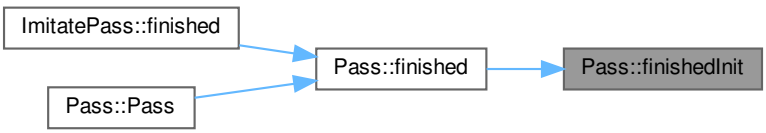
Here is the caller graph for this function:



13.12.4.15 finishedInit

```
void Pass::finishedInit (
            const QString & ,
            const QString &  )  [signal]
```

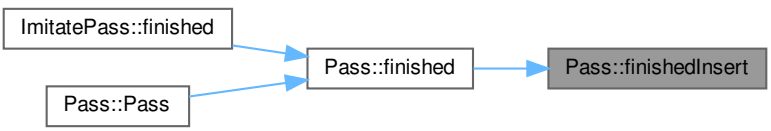
Here is the caller graph for this function:



13.12.4.16 finishedInsert

```
void Pass::finishedInsert (
            const QString & ,
            const QString &  )  [signal]
```

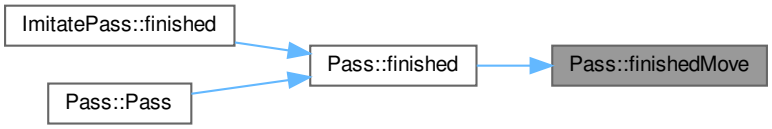
Here is the caller graph for this function:



13.12.4.17 finishedMove

```
void Pass::finishedMove (
            const QString & ,
            const QString &  )  [signal]
```

Here is the caller graph for this function:



13.12.4.18 finishedOtpGenerate

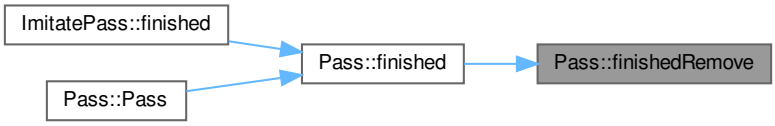
```
void Pass::finishedOtpGenerate (
            const QString &  )  [signal]
```

Here is the caller graph for this function:

13.12.4.19 finishedRemove

```
void Pass::finishedRemove (
            const QString & ,
            const QString &  )  [signal]
```

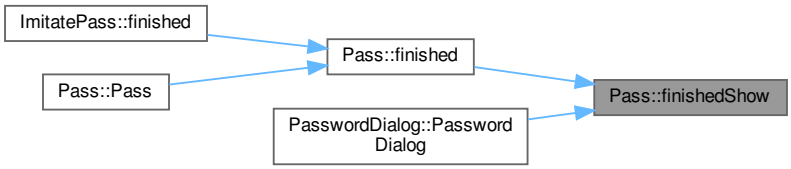
Here is the caller graph for this function:



13.12.4.20 finishedShow

```
void Pass::finishedShow (
            const QString &  )  [signal]
```

Here is the caller graph for this function:



13.12.4.21 Generate_b()

```
QString Pass::Generate_b (
            unsigned int length,
            const QString & charset )  [virtual]
```

Pass::Generate use either pwgen or internal password generator.

Parameters

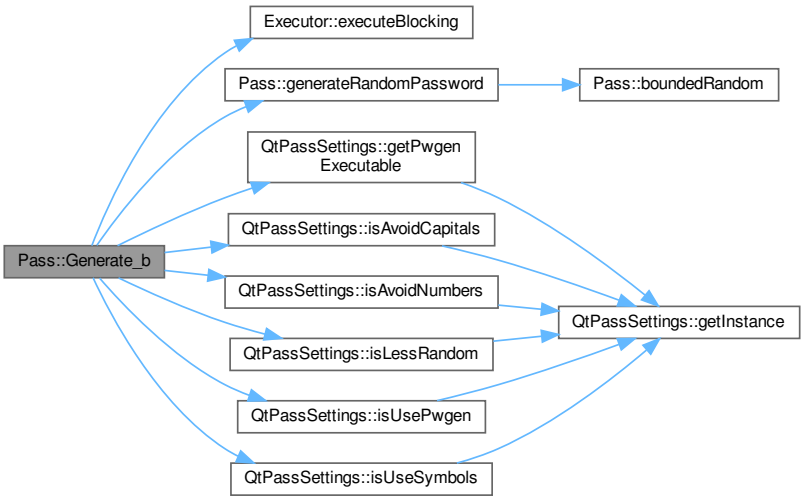
length	of the desired password
charset	to use for generation

Returns

the password

Definition at line 71 of file pass.cpp.

Here is the call graph for this function:



13.12.4.22 GenerateGPGKeys()

```
void Pass::GenerateGPGKeys (
            QString batch )
```

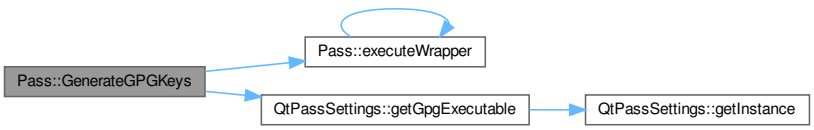
Pass::GenerateGPGKeys internal gpg keypair generator . .

Parameters

batch	GnuPG style configuration string

Definition at line 116 of file pass.cpp.

Here is the call graph for this function:



13.12.4.23 generateRandomPassword()

```
QString Pass::generateRandomPassword (
            const QString & charset,
            unsigned int length )  [protected]
```

Definition at line 364 of file pass.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

13.12.4.24 getGpgIdPath()

```
QString Pass::getGpgIdPath (
            QString for_file )  [static]
```

Pass::getGpgIdPath return gpgid file path for some file (folder).

Parameters

for_file	which file (folder) would you like the gpgid file path for.

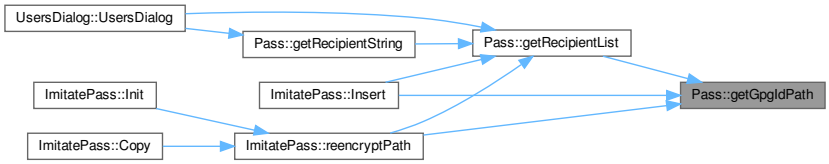
Returns

path to the gpgid file.

Definition at line 290 of file pass.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.12.4.25 getRecipientList()

```
QStringList Pass::getRecipientList (
            QString for_file )  [static]
```

Pass::getRecipientList return list of gpg-id's to encrypt for.

Parameters

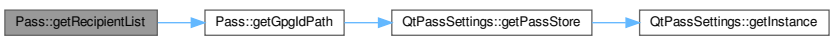
for_file	which file (folder) would you like recepients for

Returns

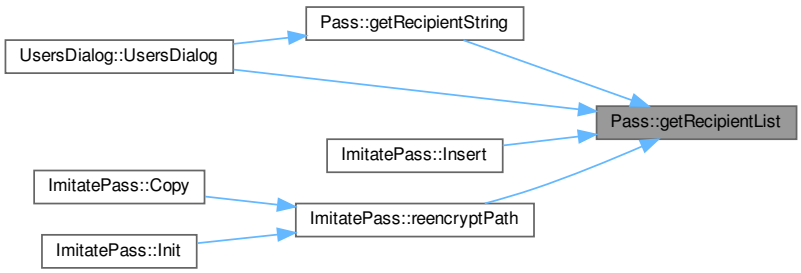
recepients gpg-id contents

Definition at line 318 of file pass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.12.4.26 getRecipientString()

```
QStringList Pass::getRecipientString (
            QString for_file,
            QString separator = " ",
            int * count = NULL )  [static]
```

Pass::getRecipientString formated string for use with GPG.

Parameters

for_file	which file (folder) would you like recepients for
separator	formating separator eg: " -r "
count	

Returns

recepient string

Definition at line 339 of file pass.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

13.12.4.27 GitInit()

```
virtual void Pass::GitInit ( )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

13.12.4.28 GitPull()

```
virtual void Pass::GitPull ( )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

13.12.4.29 GitPull_b()

```
virtual void Pass::GitPull_b ( )  [pure virtual]
```

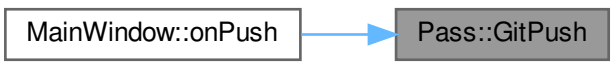
Implemented in ImitatePass, and RealPass.

13.12.4.30 GitPush()

```
virtual void Pass::GitPush ( )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

Here is the caller graph for this function:



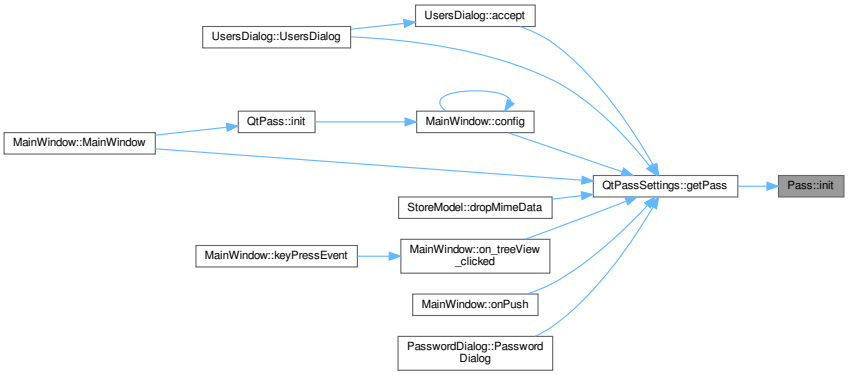
13.12.4.31 init()

```
void Pass::init ( )
```

Definition at line 47 of file pass.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

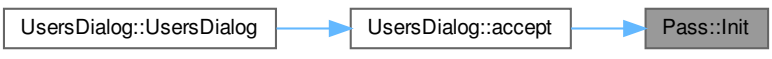


13.12.4.32 Init()

```
virtual void Pass::Init (
            QString path,
            const QList< UserInfo > & users )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

Here is the caller graph for this function:



13.12.4.33 Insert()

```
virtual void Pass::Insert (
            QString file,
            QString value,
            bool force )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

13.12.4.34 listKeys() [1/2]

```
QList< UserInfo > Pass::listKeys (
            QString keystring = "",
            bool secret = false )
```

Pass::listKeys list users.

Parameters

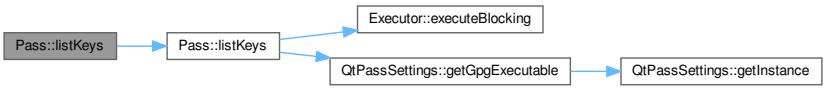
keystring	
secret	list private keys

Returns

QList<UserInfo> users

Definition at line 181 of file pass.cpp.

Here is the call graph for this function:



13.12.4.35 listKeys() [2/2]

```
QList< UserInfo > Pass::listKeys (
            QStringList keystrings,
            bool secret = false )
```

Pass::listKeys list users.

Parameters

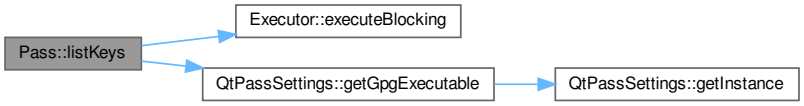
keystrings	
secret	list private keys

Returns

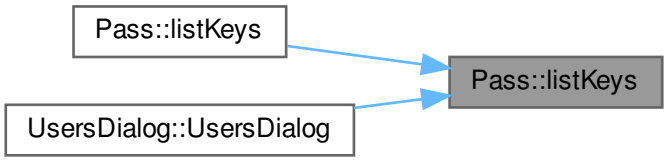
QList<UserInfo> users

Definition at line 130 of file pass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.12.4.36 Move()

```
virtual void Pass::Move (
            const QString srcDir,
            const QString dest,
            const bool force = false )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

Here is the caller graph for this function:

13.12.4.37 OtpGenerate()

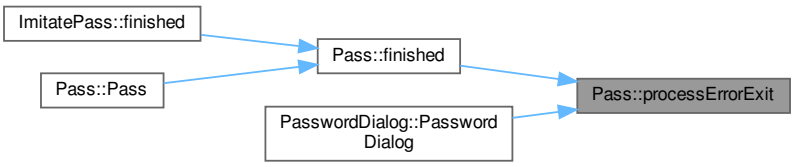
```
virtual void Pass::OtpGenerate (
            QString file )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

13.12.4.38 processErrorExit

```
void Pass::processErrorExit (
            int exitCode,
            const QString & err )  [signal]
```

Here is the caller graph for this function:



13.12.4.39 Remove()

```
virtual void Pass::Remove (
            QString file,
            bool isDir )  [pure virtual]
```

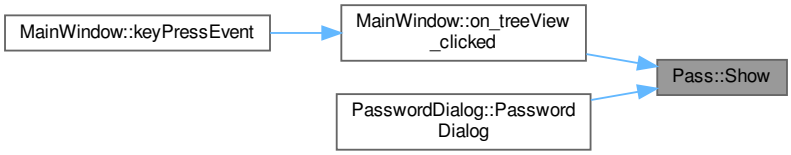
Implemented in ImitatePass, and RealPass.

13.12.4.40 Show()

```
virtual void Pass::Show (
            QString file )  [pure virtual]
```

Implemented in ImitatePass, and RealPass.

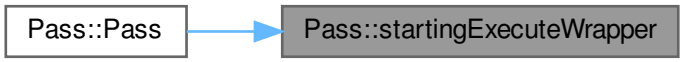
Here is the caller graph for this function:



13.12.4.41 startingExecuteWrapper

```
void Pass::startingExecuteWrapper ( )  [signal]
```

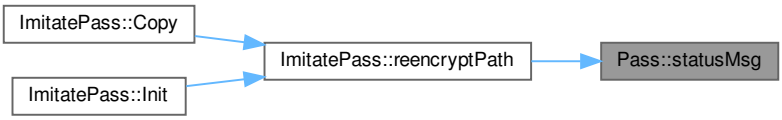
Here is the caller graph for this function:



13.12.4.42 statusMsg

```
void Pass::statusMsg (
            QString ,
            int  )  [signal]
```

Here is the caller graph for this function:



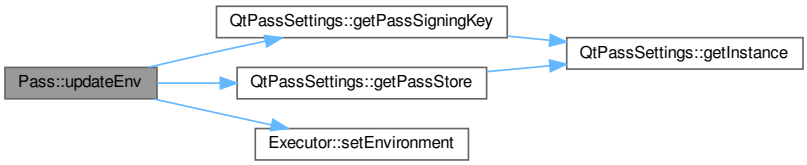
13.12.4.43 updateEnv()

```
void Pass::updateEnv ( )
```

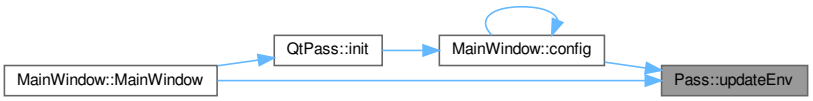
Pass::updateEnv update the execution environment (used when switching profiles)

Definition at line 248 of file pass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.12.5 Member Data Documentation

13.12.5.1 exec

```
Executor Pass::exec  [protected]
```

Definition at line 25 of file pass.h.

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

- src/pass.h
- src/pass.cpp

13.13 PasswordConfiguration Struct Reference

Holds the Password configuration settings.

```
#include <passwordconfiguration.h>
```

Public Types

- enum characterSet {
  ALLCHARS = 0 , ALPHABETICAL , ALPHANUMERIC , CUSTOM ,
  CHARSETS_COUNT }

  The selected character set.

Public Member Functions

- PasswordConfiguration ()

Public Attributes

- enum PasswordConfiguration::characterSet selected
- int length

  Length of the password.
- QString Characters [CHARSETS_COUNT]

  The different character sets.

13.13.1 Detailed Description

Holds the Password configuration settings.

Definition at line 10 of file passwordconfiguration.h.

13.13.2 Member Enumeration Documentation

13.13.2.1 characterSet

`enum PasswordConfiguration::characterSet`

The selected character set.

Enumerator

ALLCHARS	
ALPHABETICAL	
ALPHANUMERIC	
CUSTOM	
CHARSETS_COUNT	

Definition at line 14 of file passwordconfiguration.h.

13.13.3 Constructor & Destructor Documentation

13.13.3.1 PasswordConfiguration()

```
PasswordConfiguration::PasswordConfiguration ( )  [inline]
```

Definition at line 29 of file passwordconfiguration.h.

13.13.4 Member Data Documentation

13.13.4.1 Characters

```
QString PasswordConfiguration::Characters[CHARSETS_COUNT]
```

The different character sets.

Definition at line 28 of file passwordconfiguration.h.

13.13.4.2 length

```
int PasswordConfiguration::length
```

Length of the password.

Definition at line 24 of file passwordconfiguration.h.

13.13.4.3 selected

```
enum PasswordConfiguration::characterSet PasswordConfiguration::selected
```

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

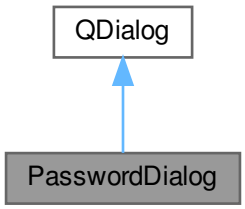
- src/passwordconfiguration.h

13.14 PasswordDialog Class Reference

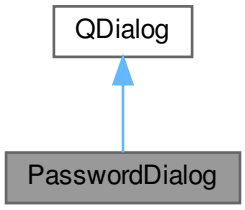
PasswordDialog Handles the inserting and editing of passwords.

`#include <passworddialog.h>`

Inheritance diagram for PasswordDialog:



Collaboration diagram for PasswordDialog:



Public Slots

- void setPass (const QString &output)

Public Member Functions

- PasswordDialog (const PasswordConfiguration &passConfig, QWidget *parent=nullptr)

  PasswordDialog::PasswordDialog basic constructor.
- PasswordDialog (const QString &file, const bool &isNew, QWidget *parent=nullptr)

  PasswordDialog::PasswordDialog complete constructor.
- ~PasswordDialog ()

  Pass{}{}wordDialog::~PasswordDialog basic destructor.
- void setPassword (QString password)

Sets content in the password field in the interface.

- QString getPassword ()

  Returns the password as set in the password field in the interface.

- void setTemplate (QString rawFields, bool useTemplate)

  Sets content in the template for the interface.

- void templateAll (bool templateAll)

  PasswordDialog::templateAll basic setter for use in PasswordDialog::setPassword templating all tokenisable lines.

- void setLength (int l)

  PasswordDialog::setLength PasswordDialog::setLength password length.

- void setPasswordCharTemplate (int t)

  PasswordDialog::setPasswordCharTemplate PasswordDialog::setPasswordCharTemplate chose the template style.

- void usePwgen (bool usePwgen)

  PasswordDialog::usePwgen PasswordDialog::usePwgen don't use own password generator.

13.14.1 Detailed Description

PasswordDialog Handles the inserting and editing of passwords.

Includes templated views.

Definition at line 20 of file passworddialog.h.

13.14.2 Constructor & Destructor Documentation

13.14.2.1 PasswordDialog() [1/2]

```
PasswordDialog::PasswordDialog (
            const PasswordConfiguration & passConfig,
            QWidget * parent = nullptr )  [explicit]
```

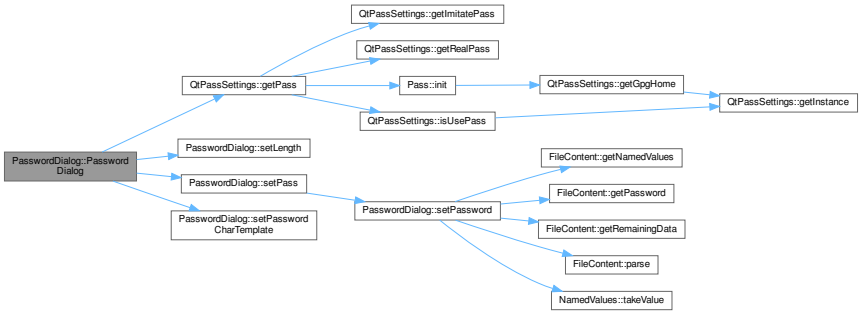
PasswordDialog::PasswordDialog basic constructor.

Parameters

passConfig	configuration constant
parent	

Definition at line 20 of file passworddialog.cpp.

Here is the call graph for this function:



13.14.2.2 PasswordDialog() [2/2]

```
PasswordDialog::PasswordDialog (
            const QString & file,
            const bool & isNew,
            QWidget * parent = nullptr )
```

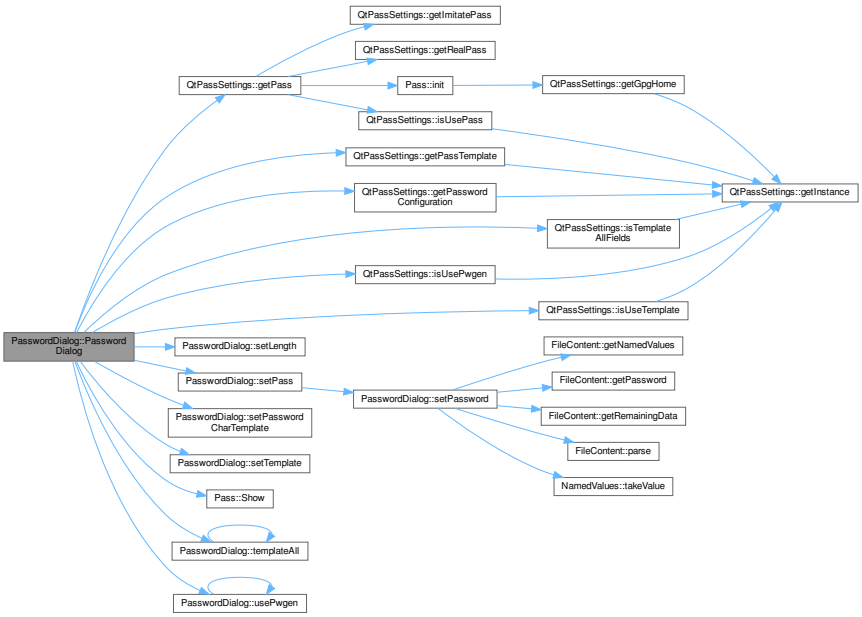
PasswordDialog::PasswordDialog complete constructor.

Parameters

file	
isNew	
parent	pointer

Definition at line 41 of file passworddialog.cpp.

Here is the call graph for this function:



13.14.2.3 ~PasswordDialog()

`PasswordDialog::~PasswordDialog ( )`

Pass{}{}wordDialog::~PasswordDialog basic destructor.

Definition at line 72 of file passworddialog.cpp.

13.14.3 Member Function Documentation

13.14.3.1 getPassword()

`QString PasswordDialog::getPassword ( )`

Returns the password as set in the password field in the interface.

PasswordDialog::getPassword join the (templated) fields to a QString for writing back.

Returns

password as a QString

See also

setPassword

Returns

collappsed password.

Definition at line 156 of file passworddialog.cpp.

13.14.3.2 setLength()

```
void PasswordDialog::setLength (
            int l )
```

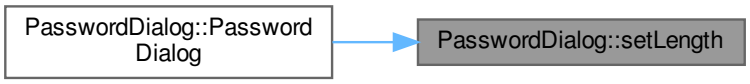
PasswordDialog::setLength PasswordDialog::setLength password length.

Parameters

l	

Definition at line 208 of file passworddialog.cpp.

Here is the caller graph for this function:

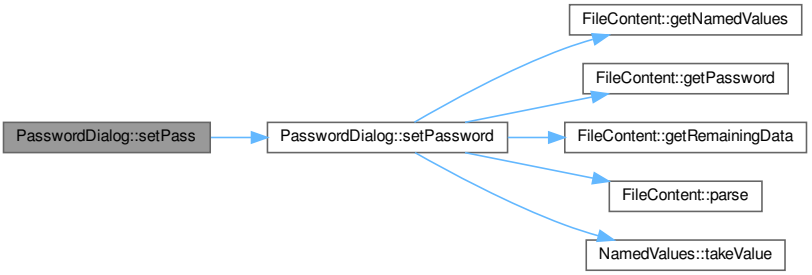


13.14.3.3 setPass

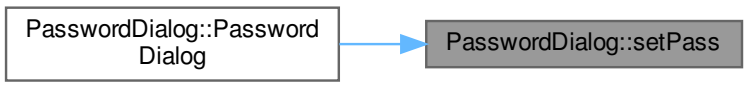
```
void PasswordDialog::setPass (
            const QString & output )  [slot]
```

Definition at line 229 of file passworddialog.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.14.3.4 setPassword()

```
void PasswordDialog::setPassword (
            QString password )
```

Sets content in the password field in the interface.

PasswordDialog::setPassword populate the (templated) fields.

Parameters

password	the password as a QString

See also

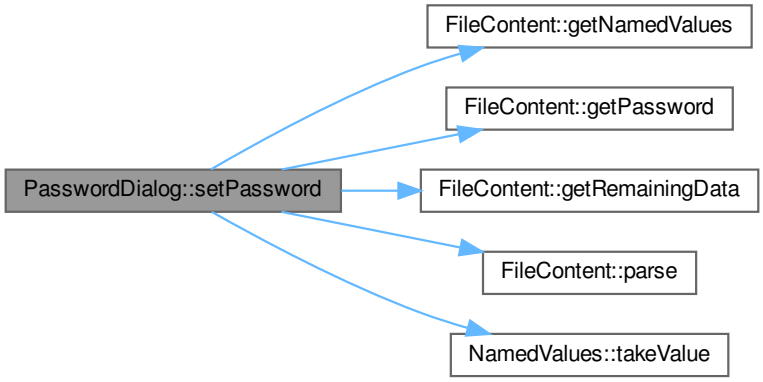
getPassword

Parameters

password	

Definition at line 124 of file passworddialog.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

13.14.3.5 setPasswordCharTemplate()

```
void PasswordDialog::setPasswordCharTemplate (
            int t )
```

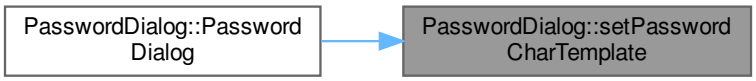
PasswordDialog::setPasswordCharTemplate PasswordDialog::setPasswordCharTemplate chose the template style.

Parameters

t	

Definition at line 215 of file passworddialog.cpp.

Here is the caller graph for this function:



13.14.3.6 setTemplate()

```
void PasswordDialog::setTemplate (
            QString rawFields,
            bool useTemplate )
```

Sets content in the template for the interface.

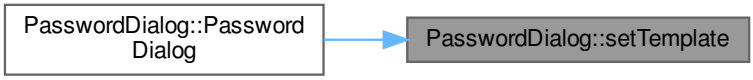
PasswordDialog::setTemplate set the template and create the fields.

Parameters

rawFields	is the template as a QString
useTemplate	whether the template is used
rawFields	

Definition at line 174 of file passworddialog.cpp.

Here is the caller graph for this function:



13.14.3.7 templateAll()

```
void PasswordDialog::templateAll (
            bool templateAll )
```

PasswordDialog::templateAll basic setter for use in PasswordDialog::setPassword templating all tokenisable lines.

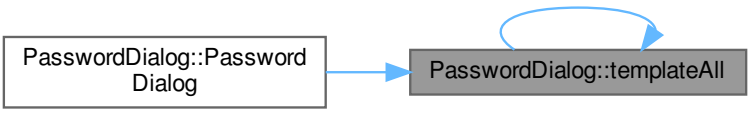
Parameters

templateAll	

Definition at line 199 of file passworddialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.14.3.8 usePwgen()

```
void PasswordDialog::usePwgen (
            bool usePwgen )
```

PasswordDialog::usePwgen PasswordDialog::usePwgen don't use own password generator.

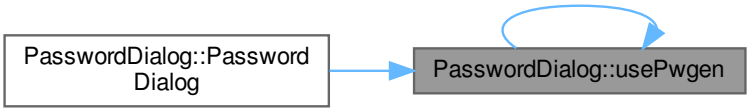
Parameters

usePwgen	

Definition at line 224 of file passworddialog.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



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

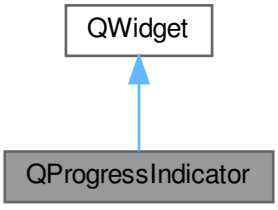
- src/passworddialog.h
- src/passworddialog.cpp

13.15 QProgressIndicator Class Reference

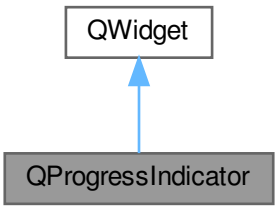
The QProgressIndicator class lets an application display a progress indicator to show that a lengthy task is under way.

`#include <qprogressindicator.h>`

Inheritance diagram for QProgressIndicator:



Collaboration diagram for QProgressIndicator:



Public Slots

- void startAnimation ()

  Starts the spin animation.
- void stopAnimation ()

  Stops the spin animation.
- void setAnimationDelay (int delay)

  Sets the delay between animation steps.
- void setDisplayedWhenStopped (bool state)

  Sets whether the component hides itself when it is not animating.
- void setColor (const QColor &color)

  Sets the color of the components to the given color.

Public Member Functions

- QProgressIndicator (QWidget ∗parent=0)

  QProgressIndicator::QProgressIndicator constructor.
- int animationDelay () const

  Returns the delay between animation steps.
- bool isAnimated () const

  Returns a Boolean value indicating whether the component is currently animated.
- bool isDisplayedWhenStopped () const

  Returns a Boolean value indicating whether the receiver shows itself even when it is not animating.
- const QColor & color () const

  Returns the color of the component.
- virtual QSize sizeHint () const

  QProgressIndicator::sizeHint default minimum size.
- int heightForWidth (int w) const

  QProgressIndicator::heightForWidth square ratio.

Protected Member Functions

- virtual void timerEvent (QTimerEvent ∗event)

  QProgressIndicator::timerEvent do the actual animation.
- virtual void paintEvent (QPaintEvent ∗event)

  QProgressIndicator::paintEvent draw the spinner.

13.15.1 Detailed Description

The QProgressIndicator class lets an application display a progress indicator to show that a lengthy task is under way.

Progress indicators are indeterminate and do nothing more than spin to show that the application is busy.

See also

QProgressBar

Definition at line 43 of file qprogressindicator.h.

13.15.2 Constructor & Destructor Documentation

13.15.2.1 QProgressIndicator()

```
QProgressIndicator::QProgressIndicator (
            QWidget * parent = 0 )  [explicit]
```

QProgressIndicator::QProgressIndicator constructor.

Parameters

parent	widget the indicator is placed in.

Definition at line 34 of file qprogressindicator.cpp.

13.15.3 Member Function Documentation

13.15.3.1 animationDelay()

```
int QProgressIndicator::animationDelay ( ) const  [inline]
```

Returns the delay between animation steps.

Returns

The number of milliseconds between animation steps. By default, the animation delay is set to 40 milliseconds.

See also

setAnimationDelay

Definition at line 53 of file qprogressindicator.h.

13.15.3.2 color()

```
const QColor & QProgressIndicator::color ( ) const  [inline]
```

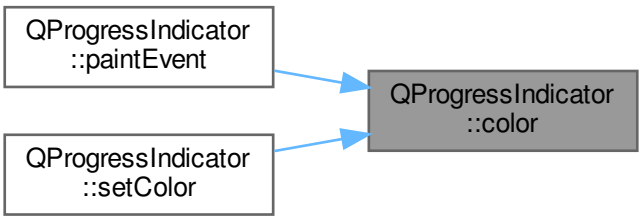
Returns the color of the component.

See also

setColor

Definition at line 73 of file qprogressindicator.h.

Here is the caller graph for this function:



13.15.3.3 heightForWidth()

```
int QProgressIndicator::heightForWidth (
            int w ) const
```

QProgressIndicator::heightForWidth square ratio.

Parameters

w	requested width

Returns

w returned height

Definition at line 96 of file qprogressindicator.cpp.

13.15.3.4 isAnimated()

```
bool QProgressIndicator::isAnimated ( ) const
```

Returns a Boolean value indicating whether the component is currently animated.

Returns

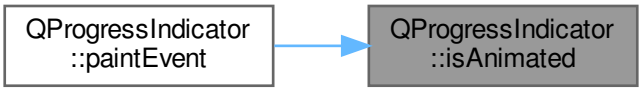
Animation state.

See also

startAnimation stopAnimation

Definition at line 41 of file qprogressindicator.cpp.

Here is the caller graph for this function:



13.15.3.5 isDisplayedWhenStopped()

```
bool QProgressIndicator::isDisplayedWhenStopped ( ) const
```

Returns a Boolean value indicating whether the receiver shows itself even when it is not animating.

Returns

Return true if the progress indicator shows itself even when it is not animating. By default, it returns false.

See also

setDisplayedWhenStopped

Definition at line 49 of file qprogressindicator.cpp.

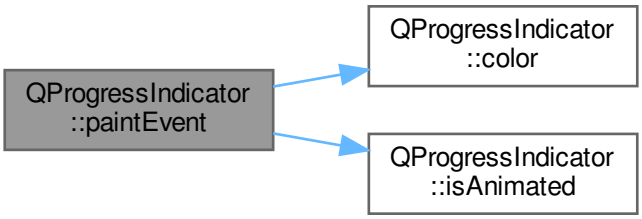
13.15.3.6 paintEvent()

```
void QProgressIndicator::paintEvent (
            QPaintEvent * event )  [protected], [virtual]
```

QProgressIndicator::paintEvent draw the spinner.

Definition at line 110 of file qprogressindicator.cpp.

Here is the call graph for this function:



13.15.3.7 setAnimationDelay

```
void QProgressIndicator::setAnimationDelay (
            int delay )  [slot]
```

Sets the delay between animation steps.

Setting the *delay* to a value larger than 40 slows the animation, while setting the *delay* to a smaller value speeds it up.

Parameters

delay	The delay, in milliseconds.

See also

animationDelay

Definition at line 69 of file qprogressindicator.cpp.

13.15.3.8 setColor

```
void QProgressIndicator::setColor (
            const QColor & color )  [slot]
```

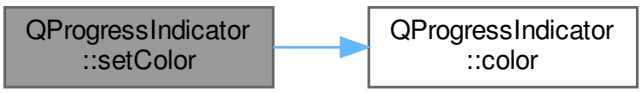
Sets the color of the components to the given color.

See also

color

Definition at line 79 of file qprogressindicator.cpp.

Here is the call graph for this function:



13.15.3.9 setDisplayedWhenStopped

```
void QProgressIndicator::setDisplayedWhenStopped (
            bool state )  [slot]
```

Sets whether the component hides itself when it is not animating.

Parameters

state	The animation state. Set false to hide the progress indicator when it is not animating; otherwise true.

See also

isDisplayedWhenStopped

Definition at line 43 of file qprogressindicator.cpp.

13.15.3.10 sizeHint()

```
QSize QProgressIndicator::sizeHint ( ) const  [virtual]
```

QProgressIndicator::sizeHint default minimum size.

Returns

QSize(20, 20)

Definition at line 89 of file qprogressindicator.cpp.

13.15.3.11 startAnimation

```
void QProgressIndicator::startAnimation ( )  [slot]
```

Starts the spin animation.

See also

stopAnimation isAnimated

Definition at line 53 of file qprogressindicator.cpp.

13.15.3.12 stopAnimation

```
void QProgressIndicator::stopAnimation ( )  [slot]
```

Stops the spin animation.

See also

startAnimation isAnimated

Definition at line 60 of file qprogressindicator.cpp.

13.15.3.13 timerEvent()

```
void QProgressIndicator::timerEvent (
            QTimerEvent * event )  [protected], [virtual]
```

QProgressIndicator::timerEvent do the actual animation.

Definition at line 101 of file qprogressindicator.cpp.

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

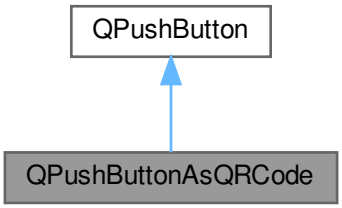
- src/qprogressindicator.h
- src/qprogressindicator.cpp

13.16 QPushButtonAsQRCode Class Reference

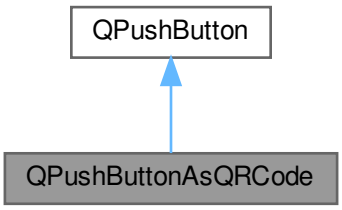
Stylish widget to display the field as QR Code.

`#include <qpushbuttonasqrcode.h>`

Inheritance diagram for QPushButtonAsQRCode:



Collaboration diagram for QPushButtonAsQRCode:



Signals

- void clicked (QString)

Public Member Functions

- QPushButtonAsQRCode (const QString &textToCopy="", QWidget *parent=nullptr)
  QPushButtonAsQRCode::QPushButtonAsQRCode basic constructor.
- QString getTextToCopy () const
  QPushButtonAsQRCode::getTextToCopy returns the text of associated text field.
- void setTextToCopy (const QString &value)
  QPushButtonAsQRCode::setTextToCopy sets text from associated text field.

13.16.1 Detailed Description

Stylish widget to display the field as QR Code.

Definition at line 11 of file qpushbuttonasqrcode.h.

13.16.2 Constructor & Destructor Documentation

13.16.2.1 QPushButtonAsQRCode()

```
QPushButtonAsQRCode::QPushButtonAsQRCode (
            const QString & textToCopy = "",
            QWidget * parent = nullptr )  [explicit]
```

QPushButtonAsQRCode::QPushButtonAsQRCode basic constructor.

Parameters

textToCopy	the text to display as qrcode
parent	the parent window

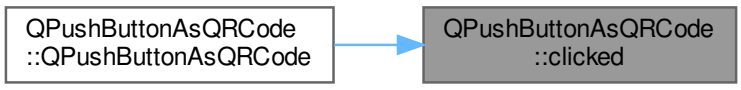
Definition at line 12 of file qpushbuttonasqrcode.cpp.

13.16.3 Member Function Documentation

13.16.3.1 clicked

```
void QPushButtonAsQRCode::clicked (
            QString  )  [signal]
```

Here is the caller graph for this function:



13.16.3.2 getTextToCopy()

```
QString QPushButtonAsQRCode::getTextToCopy ( ) const
```

QPushButtonAsQRCode::getTextToCopy returns the text of associated text field.

Returns

QString textToCopy

Definition at line 25 of file qpushbuttonasqrcode.cpp.

13.16.3.3 setTextToCopy()

```
void QPushButtonAsQRCode::setTextToCopy (
            const QString & value )
```

QPushButtonAsQRCode::setTextToCopy sets text from associated text field.

Parameters

value	QString text to be copied

Definition at line 32 of file qpushbuttonasqrcode.cpp.

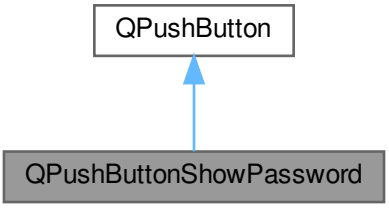
The documentation for this class was generated from the following files:

- src/qpushbuttonasqrcode.h
- src/qpushbuttonasqrcode.cpp

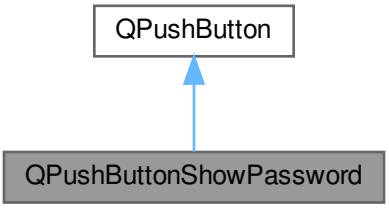
13.17 QPushButtonShowPassword Class Reference

```
#include <qpushbuttonshowpassword.h>
```

Inheritance diagram for QPushButtonShowPassword:



Collaboration diagram for QPushButtonShowPassword:



Signals

- void clicked (QString)

Public Member Functions

- QPushButtonShowPassword (QLineEdit ∗line, QWidget ∗parent=nullptr)
  QPushButtonAsQRCode::QPushButtonAsQRCode basic constructor.

13.17.1 Detailed Description

Definition at line 12 of file qpushbuttonshowpassword.h.

13.17.2 Constructor & Destructor Documentation

13.17.2.1 QPushButtonShowPassword()

```
QPushButtonShowPassword::QPushButtonShowPassword (
            QLineEdit * line,
            QWidget * parent = nullptr )  [explicit]
```

QPushButtonAsQRCode::QPushButtonAsQRCode basic constructor.

Parameters

textToCopy	the text to display as qrcode
parent	the parent window

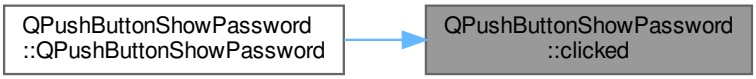
Definition at line 12 of file qpushbuttonshowpassword.cpp.

13.17.3 Member Function Documentation

13.17.3.1 clicked

```
void QPushButtonShowPassword::clicked (
            QString  )  [signal]
```

Here is the caller graph for this function:



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

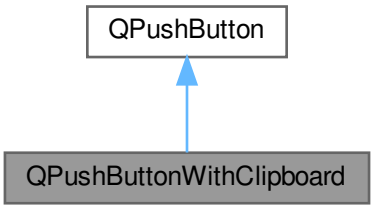
- src/qpushbuttonshowpassword.h
- src/qpushbuttonshowpassword.cpp

13.18 QPushButtonWithClipboard Class Reference

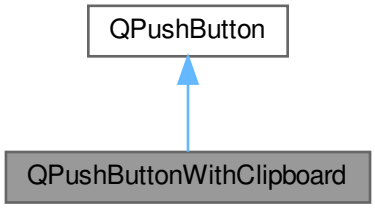
Stylish widget to allow copying of password and account details.

```
#include <qpushbuttonwithclipboard.h>
```

Inheritance diagram for QPushButtonWithClipboard:



Collaboration diagram for QPushButtonWithClipboard:



Signals

- void clicked (QString)

Public Member Functions

- QPushButtonWithClipboard (const QString &textToCopy="", QWidget *parent=nullptr)
  QPushButtonWithClipboard::QPushButtonWithClipboard basic constructor.
- QString getTextToCopy () const
  QPushButtonWithClipboard::getTextToCopy returns the text of associated text field.
- void setTextToCopy (const QString &value)
  QPushButtonWithClipboard::setTextToCopy sets text from associated text field.

13.18.1 Detailed Description

Stylish widget to allow copying of password and account details.

Definition at line 11 of file qpushbuttonwithclipboard.h.

13.18.2 Constructor & Destructor Documentation

13.18.2.1 QPushButtonWithClipboard()

```
QPushButtonWithClipboard::QPushButtonWithClipboard (
            const QString & textToCopy = "",
            QWidget * parent = nullptr )  [explicit]
```

QPushButtonWithClipboard::QPushButtonWithClipboard basic constructor.

Parameters

textToCopy	the text to paste into the clipboard
parent	the parent window

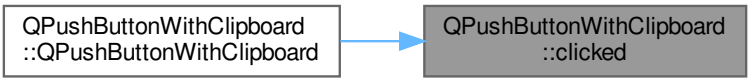
Definition at line 12 of file qpushbuttonwithclipboard.cpp.

13.18.3 Member Function Documentation

13.18.3.1 clicked

```
void QPushButtonWithClipboard::clicked (
            QString  )  [signal]
```

Here is the caller graph for this function:



13.18.3.2 getTextToCopy()

```
QString QPushButtonWithClipboard::getTextToCopy ( ) const
```

QPushButtonWithClipboard::getTextToCopy returns the text of associated text field.

Returns

QString textToCopy

Definition at line 27 of file qpushbuttonwithclipboard.cpp.

13.18.3.3 setTextToCopy()

```
void QPushButtonWithClipboard::setTextToCopy (
            const QString & value )
```

QPushButtonWithClipboard::setTextToCopy sets text from associated text field.

Parameters

value	QString text to be copied

Definition at line 34 of file qpushbuttonwithclipboard.cpp.

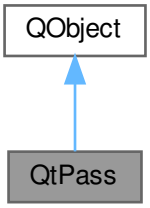
The documentation for this class was generated from the following files:

- src/qpushbuttonwithclipboard.h
- src/qpushbuttonwithclipboard.cpp

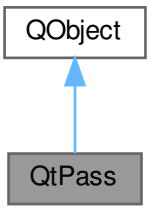
13.19 QtPass Class Reference

```
#include <qtpass.h>
```

Inheritance diagram for QtPass:



Collaboration diagram for QtPass:



Public Slots

- void clearClipboard ()

  MainWindow::clearClipboard remove clipboard contents.
- void copyTextToClipboard (const QString &text)

  MainWindow::copyTextToClipboard copies text to your clipboard.
- void showTextAsQRCode (const QString &text)

  displays the text as qrcode

Public Member Functions

- QtPass (MainWindow *mainWindow)
- ~QtPass ()

  QtPass::~QtPass destroy!
- bool init ()

  QtPass::init make sure we are ready to go as soon as possible.
- void setClippedText (const QString &, const QString &p_output=QString())
- void clearClippedText ()
- void setClipboardTimer ()
- bool isFreshStart ()
- void setFreshStart (const bool &fs)

13.19.1 Detailed Description

Definition at line 10 of file qtpass.h.

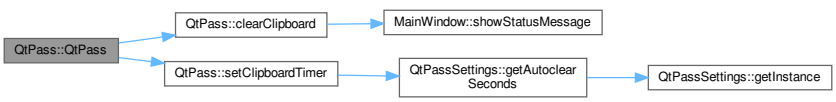
13.19.2 Constructor & Destructor Documentation

13.19.2.1 QtPass()

```
QtPass::QtPass (
            MainWindow * mainWindow )
```

Definition at line 28 of file qtpass.cpp.

Here is the call graph for this function:



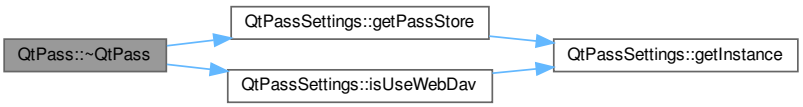
13.19.2.2 ~QtPass()

```
QtPass::~QtPass ( )
```

QtPass::~QtPass destroy!

Definition at line 44 of file qtpass.cpp.

Here is the call graph for this function:



13.19.3 Member Function Documentation

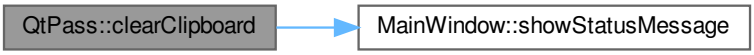
13.19.3.1 clearClipboard

```
void QtPass::clearClipboard ( )  [slot]
```

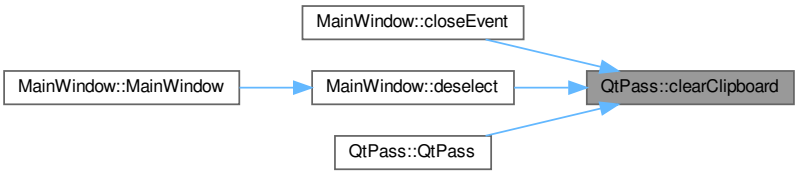
MainWindow::clearClipboard remove clipboard contents.

Definition at line 366 of file qtpass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.19.3.2 clearClippedText()

```
void QtPass::clearClippedText ( )
```

Definition at line 357 of file qtpass.cpp.

Here is the caller graph for this function:

13.19.3.3 copyTextToClipboard

```
void QtPass::copyTextToClipboard (
            const QString & text )  [slot]
```

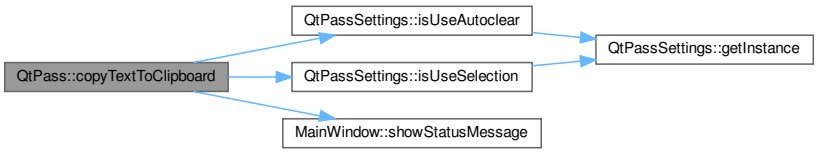
MainWindow::copyTextToClipboard copies text to your clipboard.

Parameters

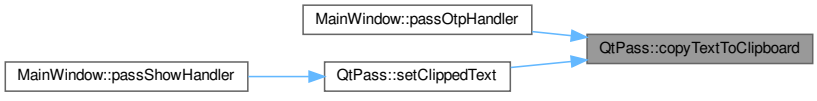
text	

Definition at line 391 of file qtpass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



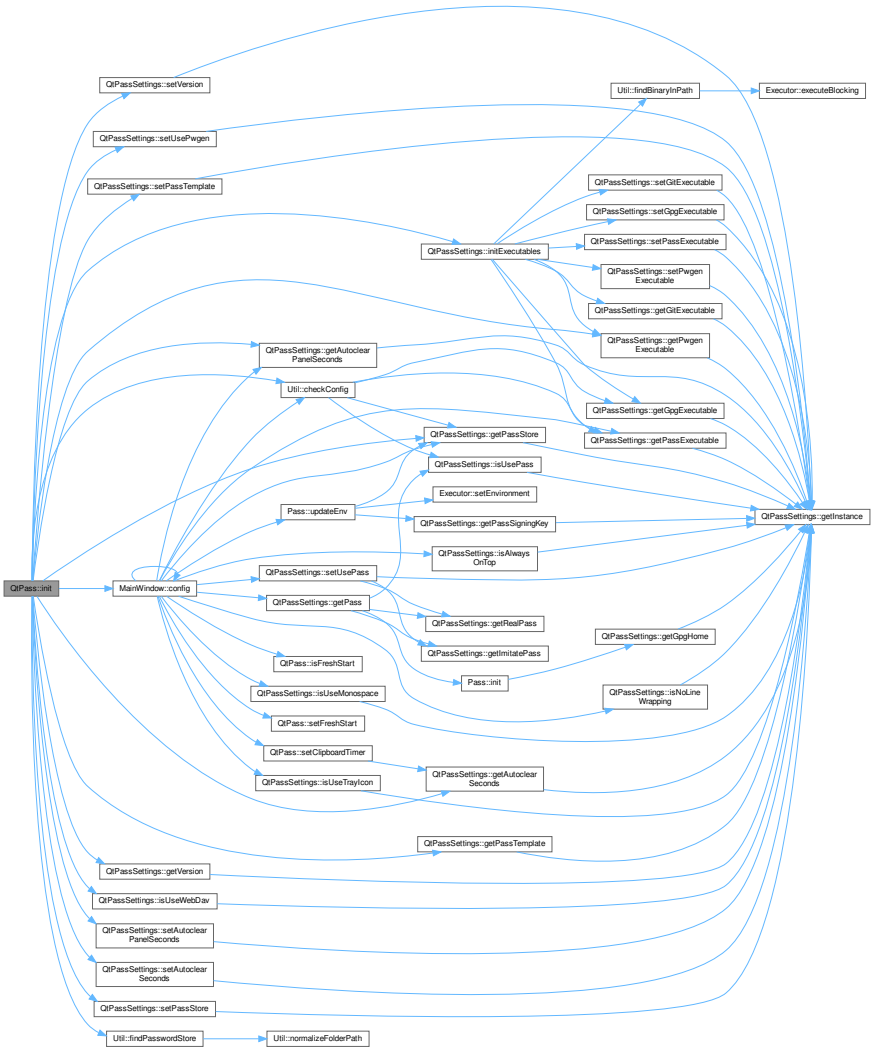
13.19.3.4 init()

```
bool QtPass::init ( )
```

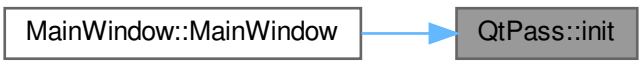
QtPass::init make sure we are ready to go as soon as possible.

Definition at line 61 of file qtpass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.19.3.5 isFreshStart()

```
bool QtPass::isFreshStart ( )  [inline]
```

Definition at line 21 of file qtpass.h.

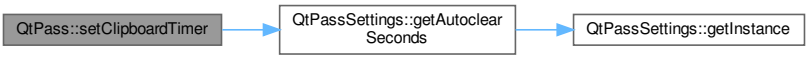
Here is the caller graph for this function:

13.19.3.6 setClipboardTimer()

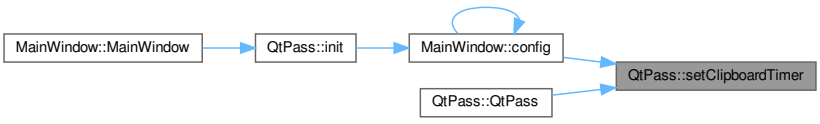
```
void QtPass::setClipboardTimer ( )
```

Definition at line 359 of file qtpass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

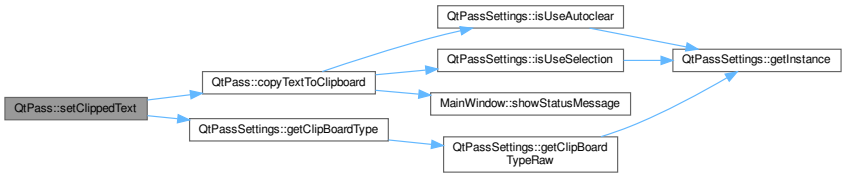


13.19.3.7 setClippedText()

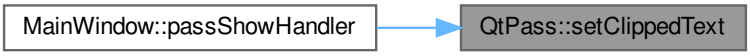
```
void QtPass::setClippedText (
            const QString & password,
            const QString & p_output = QString() )
```

Definition at line 349 of file qtpass.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

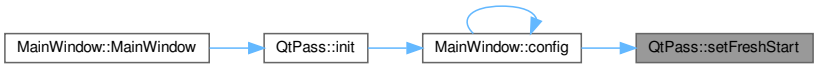


13.19.3.8 setFreshStart()

```
void QtPass::setFreshStart (
            const bool & fs )  [inline]
```

Definition at line 22 of file qtpass.h.

Here is the caller graph for this function:



13.19.3.9 showTextAsQRCode

```
void QtPass::showTextAsQRCode (
            const QString & text )  [slot]
```

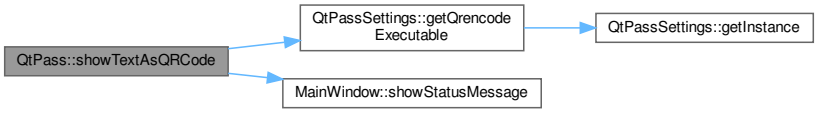
displays the text as qrcode

Parameters

text	

Definition at line 410 of file qtpass.cpp.

Here is the call graph for this function:



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

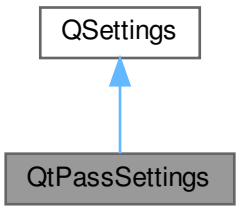
- src/qtpass.h
- src/qtpass.cpp

13.20 QtPassSettings Class Reference

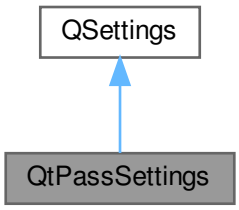
Singleton that stores qtpass' settings, saves and loads config.

`#include <qtpasssettings.h>`

Inheritance diagram for QtPassSettings:



Collaboration diagram for QtPassSettings:



Static Public Member Functions

- static QtPassSettings * getInstance ()
- static QString getVersion (const QString &defaultValue=QVariant().toString())
- static void setVersion (const QString &version)
- static QByteArray getGeometry (const QByteArray &defaultValue=QVariant().toByteArray())
- static void setGeometry (const QByteArray &geometry)
- static QByteArray getSavestate (const QByteArray &defaultValue=QVariant().toByteArray())
- static void setSavestate (const QByteArray &saveState)
- static QPoint getPos (const QPoint &defaultValue=QVariant().toPoint())
- static void setPos (const QPoint &pos)
- static QSize getSize (const QSize &defaultValue=QVariant().toSize())
- static void setSize (const QSize &size)
- static bool isMaximized (const bool &defaultValue=QVariant().toBool())
- static void setMaximized (const bool &maximized)
- static bool isUsePass (const bool &defaultValue=QVariant().toBool())
- static void setUsePass (const bool &usePass)
- static int getClipBoardTypeRaw (const Enums::clipBoardType &defaultvalue=Enums::CLIPBOARD_NEVER)
- static Enums::clipBoardType getClipBoardType (const Enums::clipBoardType &defaultvalue=Enums::CLIPBOARD_NEVER)
- static void setClipBoardType (const int &clipBoardType)
- static bool isUseSelection (const bool &defaultValue=QVariant().toBool())
- static void setUseSelection (const bool &useSelection)
- static bool isUseAutoclear (const bool &defaultValue=QVariant().toBool())
- static void setUseAutoclear (const bool &useAutoclear)
- static int getAutoclearSeconds (const int &defaultValue=QVariant().toInt())
- static void setAutoclearSeconds (const int &autoClearSeconds)
- static bool isUseAutoclearPanel (const bool &defaultValue=QVariant().toBool())
- static void setUseAutoclearPanel (const bool &useAutoclearPanel)
- static int getAutoclearPanelSeconds (const int &defaultValue=QVariant().toInt())
- static void setAutoclearPanelSeconds (const int &autoClearPanelSeconds)
- static bool isHidePassword (const bool &defaultValue=QVariant().toBool())
- static void setHidePassword (const bool &hidePassword)
- static bool isHideContent (const bool &defaultValue=QVariant().toBool())
- static void setHideContent (const bool &hideContent)
- static bool isUseMonospace (const bool &defaultValue=QVariant().toBool())
- static void setUseMonospace (const bool &useMonospace)
- static bool isDisplayAsIs (const bool &defaultValue=QVariant().toBool())
- static void setDisplayAsIs (const bool &displayAsIs)

- static bool isNoLineWrapping (const bool &defaultValue=QVariant().toBool())
- static void setNoLineWrapping (const bool &noLineWrapping)
- static bool isAddGPGId (const bool &defaultValue=QVariant().toBool())
- static void setAddGPGId (const bool &addGPGId)
- static QString getPassStore (const QString &defaultValue=QVariant().toString())
- static void setPassStore (const QString &passStore)
- static QString getPassSigningKey (const QString &defaultValue=QVariant().toString())
- static void setPassSigningKey (const QString &passSigningKey)
- static void initExecutables ()
- static QString getPassExecutable (const QString &defaultValue=QVariant().toString())
- static void setPassExecutable (const QString &passExecutable)
- static QString getGitExecutable (const QString &defaultValue=QVariant().toString())
- static void setGitExecutable (const QString &gitExecutable)
- static QString getGpgExecutable (const QString &defaultValue=QVariant().toString())
- static void setGpgExecutable (const QString &gpgExecutable)
- static QString getPwgenExecutable (const QString &defaultValue=QVariant().toString())
- static void setPwgenExecutable (const QString &pwgenExecutable)
- static QString getGpgHome (const QString &defaultValue=QVariant().toString())
- static bool isUseWebDav (const bool &defaultValue=QVariant().toBool())
- static void setUseWebDav (const bool &useWebDav)
- static QString getWebDavUrl (const QString &defaultValue=QVariant().toString())
- static void setWebDavUrl (const QString &webDavUrl)
- static QString getWebDavUser (const QString &defaultValue=QVariant().toString())
- static void setWebDavUser (const QString &webDavUser)
- static QString getWebDavPassword (const QString &defaultValue=QVariant().toString())
- static void setWebDavPassword (const QString &webDavPassword)
- static QString getProfile (const QString &defaultValue=QVariant().toString())
- static void setProfile (const QString &profile)
- static bool isUseGit (const bool &defaultValue=QVariant().toBool())
- static void setUseGit (const bool &useGit)
- static bool isUseOtp (const bool &defaultValue=QVariant().toBool())
- static void setUseOtp (const bool &useOtp)
- static bool isUseQrencode (const bool &defaultValue=QVariant().toBool())
- static void setUseQrencode (const bool &useQrencode)
- static QString getQrencodeExecutable (const QString &defaultValue=QVariant().toString())
- static void setQrencodeExecutable (const QString &qrencodeExecutable)
- static bool isUsePwgen (const bool &defaultValue=QVariant().toBool())
- static void setUsePwgen (const bool &usePwgen)
- static bool isAvoidCapitals (const bool &defaultValue=QVariant().toBool())
- static void setAvoidCapitals (const bool &avoidCapitals)
- static bool isAvoidNumbers (const bool &defaultValue=QVariant().toBool())
- static void setAvoidNumbers (const bool &avoidNumbers)
- static bool isLessRandom (const bool &defaultValue=QVariant().toBool())
- static void setLessRandom (const bool &lessRandom)
- static bool isUseSymbols (const bool &defaultValue=QVariant().toBool())
- static void setUseSymbols (const bool &useSymbols)
- static PasswordConfiguration getPasswordConfiguration ()
- static void setPasswordConfiguration (const PasswordConfiguration &config)
- static void setPasswordLength (const int &passwordLength)
- static void setPasswordCharsselection (const int &passwordCharsselection)
- static void setPasswordChars (const QString &passwordChars)
- static bool isUseTrayIcon (const bool &defaultValue=QVariant().toBool())
- static void setUseTrayIcon (const bool &useTrayIcon)
- static bool isHideOnClose (const bool &defaultValue=QVariant().toBool())
- static void setHideOnClose (const bool &hideOnClose)

- static bool isStartMinimized (const bool &defaultValue=QVariant().toBool())
- static void setStartMinimized (const bool &startMinimized)
- static bool isAlwaysOnTop (const bool &defaultValue=QVariant().toBool())
- static void setAlwaysOnTop (const bool &alwaysOnTop)
- static bool isAutoPull (const bool &defaultValue=QVariant().toBool())
- static void setAutoPull (const bool &autoPull)
- static bool isAutoPush (const bool &defaultValue=QVariant().toBool())
- static void setAutoPush (const bool &autoPush)
- static QString getPassTemplate (const QString &defaultValue=QVariant().toString())
- static void setPassTemplate (const QString &passTemplate)
- static bool isUseTemplate (const bool &defaultValue=QVariant().toBool())
- static void setUseTemplate (const bool &useTemplate)
- static bool isTemplateAllFields (const bool &defaultValue=QVariant().toBool())
- static void setTemplateAllFields (const bool &templateAllFields)
- static QHash< QString, QHash< QString, QString > > getProfiles ()
- static void setProfiles (const QHash< QString, QHash< QString, QString > > &profiles)
- static Pass * getPass ()
- static RealPass * getRealPass ()
- static ImitatePass * getImitatePass ()

13.20.1 Detailed Description

Singleton that stores qtpass' settings, saves and loads config.

Definition at line 21 of file qtpasssettings.h.

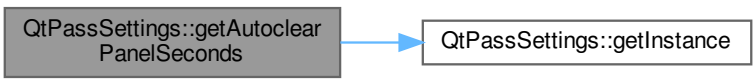
13.20.2 Member Function Documentation

13.20.2.1 getAutoclearPanelSeconds()

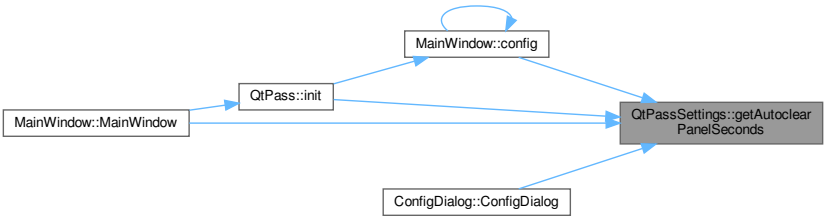
```
int QtPassSettings::getAutoclearPanelSeconds (
            const int & defaultValue = QVariant().toInt() )  [static]
```

Definition at line 236 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

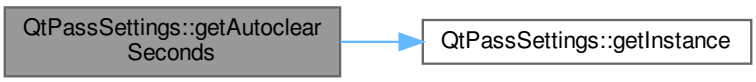


13.20.2.2 getAutoclearSeconds()

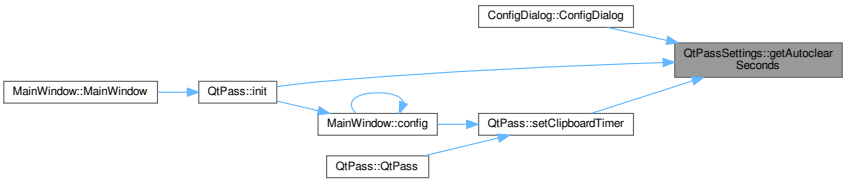
```
int QtPassSettings::getAutoclearSeconds (
            const int & defaultValue = QVariant().toInt() )  [static]
```

Definition at line 216 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.20.2.3 getClipBoardType()

```
Enums::clipBoardType QtPassSettings::getClipBoardType (
            const Enums::clipBoardType & defaultvalue = Enums::CLIPBOARD_NEVER )  [static]
```

Definition at line 191 of file qtpasssettings.cpp.

Here is the call graph for this function:

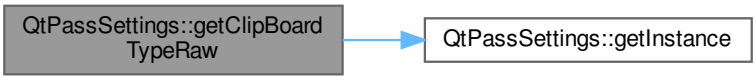
Here is the caller graph for this function:

13.20.2.4 getClipBoardTypeRaw()

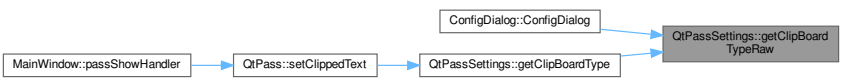
```
int QtPassSettings::getClipBoardTypeRaw (
            const Enums::clipBoardType & defaultvalue = Enums::CLIPBOARD_NEVER )  [static]
```

Definition at line 183 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



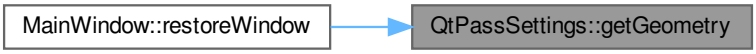
13.20.2.5 getGeometry()

```
QByteArray QtPassSettings::getGeometry (
            const QByteArray & defaultValue = QVariant().toByteArray() )  [static]
```

Definition at line 128 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



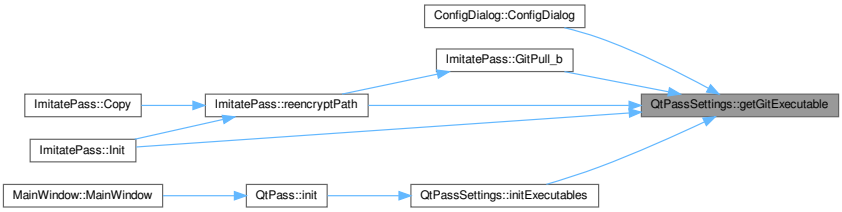
13.20.2.6 getGitExecutable()

```
QString QtPassSettings::getGitExecutable (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 361 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



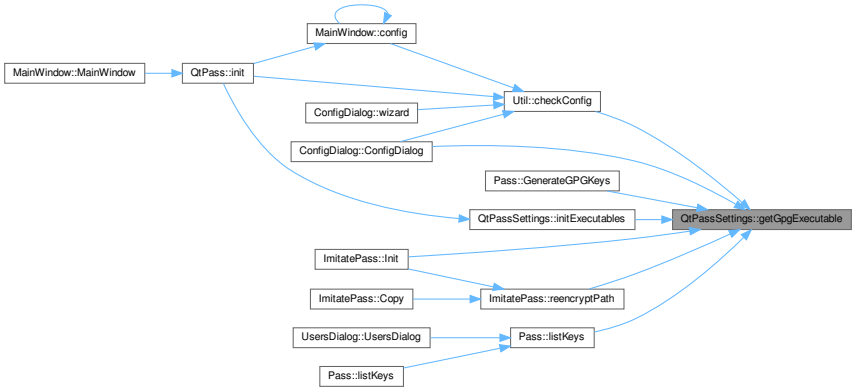
13.20.2.7 getGpgExecutable()

```
QString QtPassSettings::getGpgExecutable (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 370 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



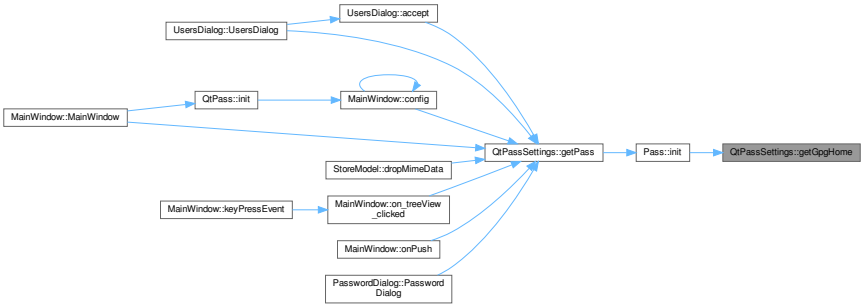
13.20.2.8 getGpgHome()

```
QString QtPassSettings::getGpgHome (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 388 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

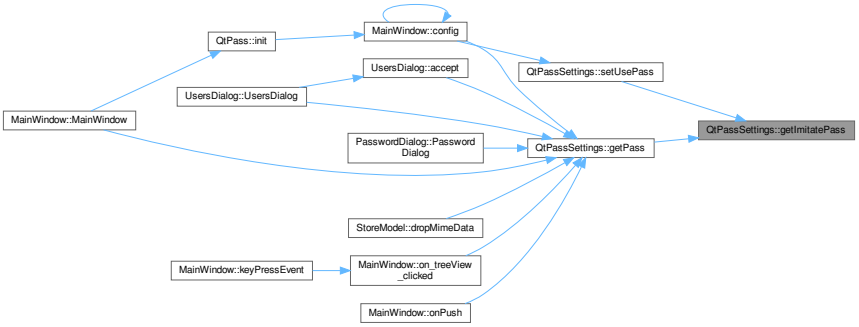


13.20.2.9 getImitatePass()

```
ImitatePass * QtPassSettings::getImitatePass ( )  [static]
```

Definition at line 618 of file qtpasssettings.cpp.

Here is the caller graph for this function:

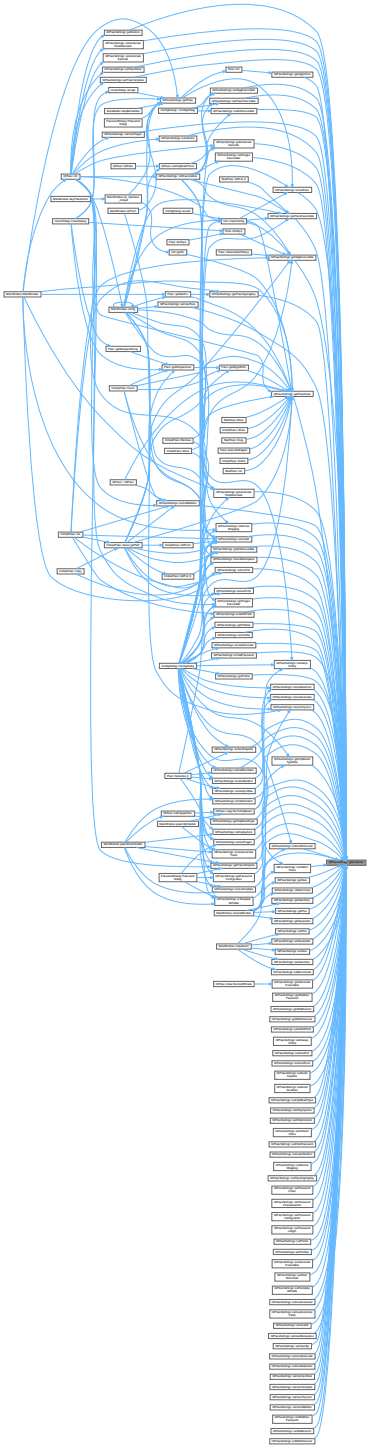


13.20.2.10 getInstance()

```
QtPassSettings * QtPassSettings::getInstance ( )  [static]
```

Definition at line 19 of file qtpasssettings.cpp.

Here is the caller graph for this function:

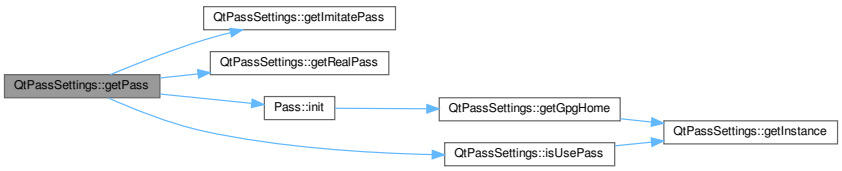


13.20.2.11 getPass()

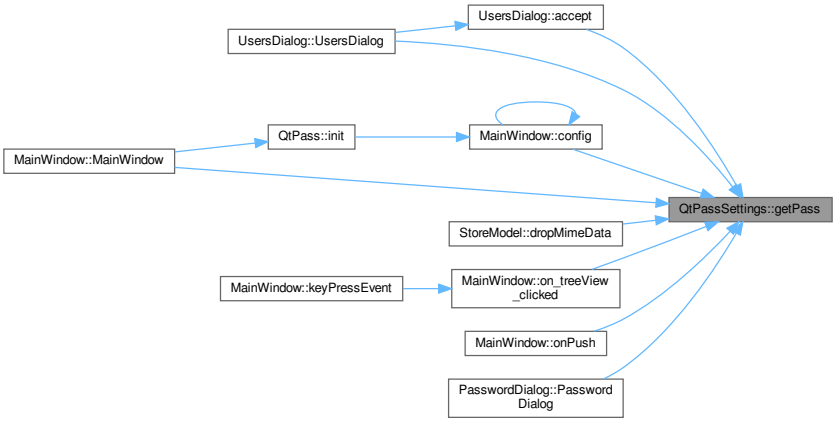
```
Pass * QtPassSettings::getPass ( )  [static]
```

Definition at line 107 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

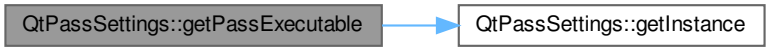


13.20.2.12 getPassExecutable()

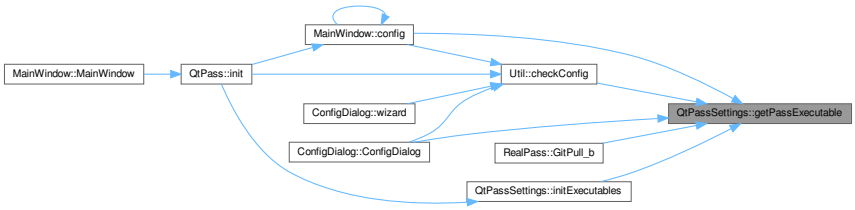
```
QString QtPassSettings::getPassExecutable (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 352 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

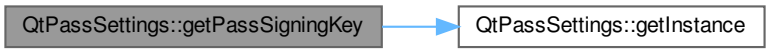


13.20.2.13 getPassSigningKey()

```
QString QtPassSettings::getPassSigningKey (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 326 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

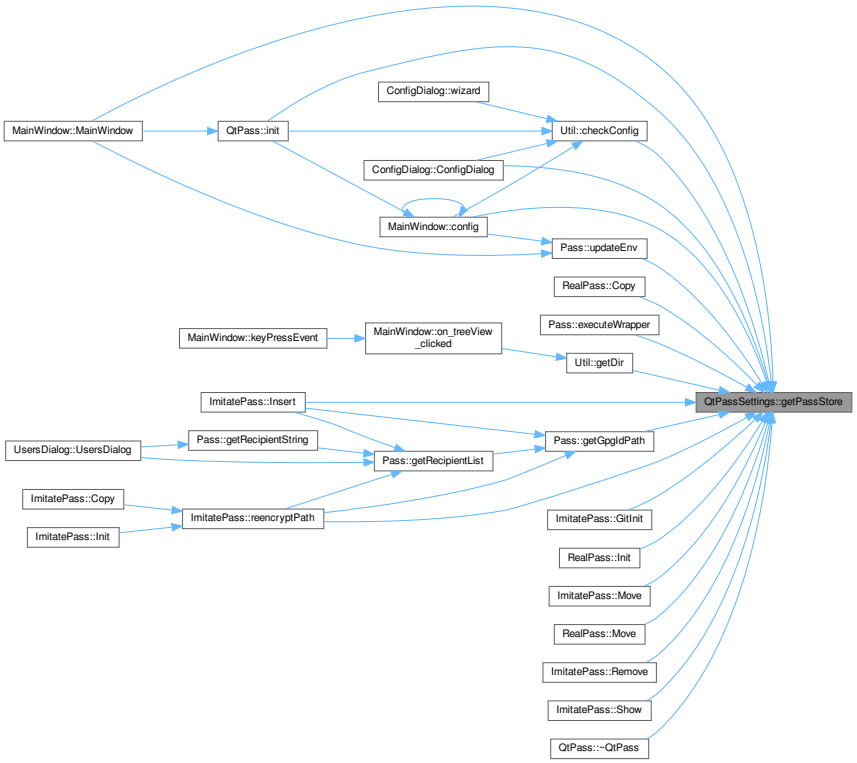
13.20.2.14 getPassStore()

```
QString QtPassSettings::getPassStore (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 301 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



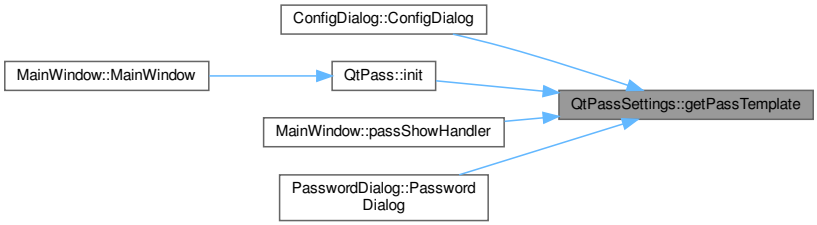
13.20.2.15 getPassTemplate()

```
QString QtPassSettings::getPassTemplate (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 585 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

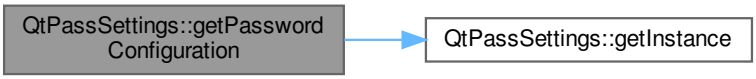


13.20.2.16 getPasswordConfiguration()

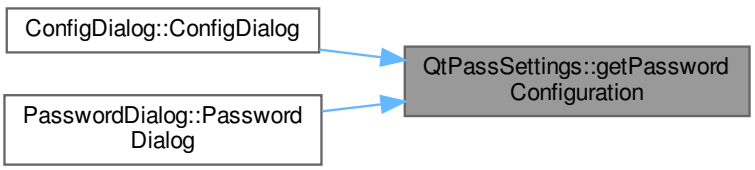
```
PasswordConfiguration QtPassSettings::getPasswordConfiguration ( )  [static]
```

Definition at line 35 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

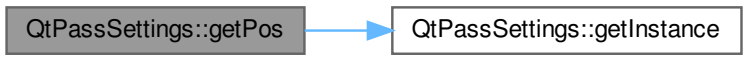


13.20.2.17 getPos()

```
QPoint QtPassSettings::getPos (
            const QPoint & defaultValue = QVariant().toPoint() )  [static]
```

Definition at line 146 of file qtpasssettings.cpp.

Here is the call graph for this function:



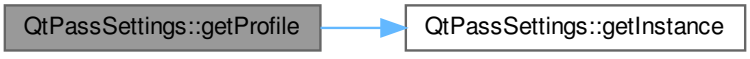
Here is the caller graph for this function:

13.20.2.18 getProfile()

```
QString QtPassSettings::getProfile (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 430 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

13.20.2.19 getProfiles()

```
QHash< QString, QHash< QString, QString > > QtPassSettings::getProfiles ( )  [static]
```

Definition at line 61 of file qtpasssettings.cpp.

Here is the call graph for this function:

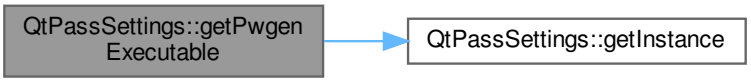
Here is the caller graph for this function:

13.20.2.20 getPwgenExecutable()

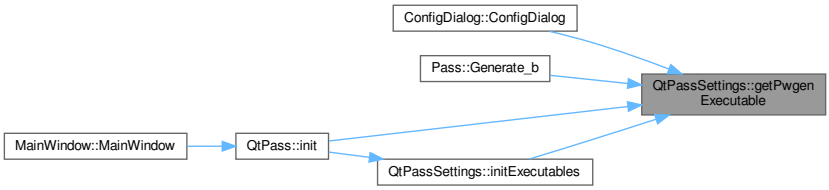
```
QString QtPassSettings::getPwgenExecutable (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 379 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

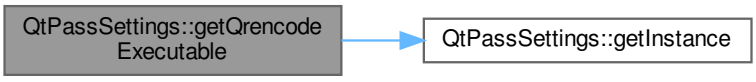


13.20.2.21 getQrencodeExecutable()

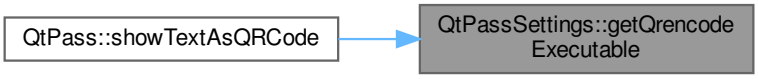
```
QString QtPassSettings::getQrencodeExecutable (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 464 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

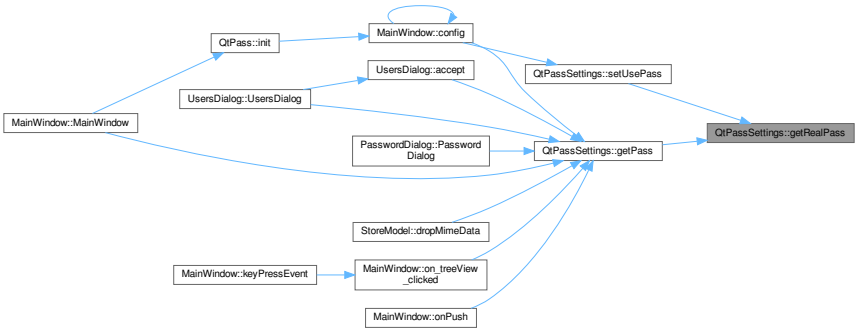


13.20.2.22 getRealPass()

```
RealPass * QtPassSettings::getRealPass ( )  [static]
```

Definition at line 613 of file qtpasssettings.cpp.

Here is the caller graph for this function:



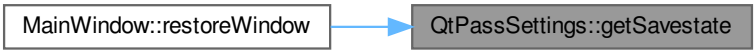
13.20.2.23 getSavestate()

```
QByteArray QtPassSettings::getSavestate (
            const QByteArray & defaultValue = QVariant().toByteArray() )  [static]
```

Definition at line 137 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

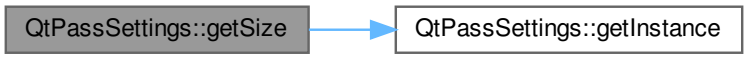


13.20.2.24 getSize()

```
QSize QtPassSettings::getSize (
            const QSize & defaultValue = QVariant().toSize() )  [static]
```

Definition at line 153 of file qtpasssettings.cpp.

Here is the call graph for this function:



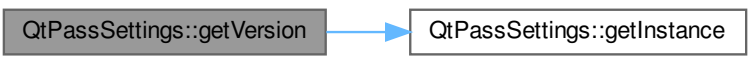
Here is the caller graph for this function:

13.20.2.25 getVersion()

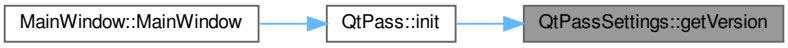
```
QString QtPassSettings::getVersion (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 119 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

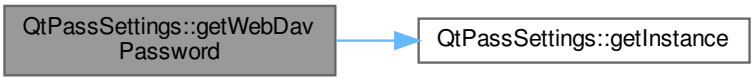


13.20.2.26 getWebDavPassword()

```
QString QtPassSettings::getWebDavPassword (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 421 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.27 getWebDavUrl()

```
QString QtPassSettings::getWebDavUrl (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 403 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.28 getWebDavUser()

```
QString QtPassSettings::getWebDavUser (
            const QString & defaultValue = QVariant().toString() )  [static]
```

Definition at line 412 of file qtpasssettings.cpp.

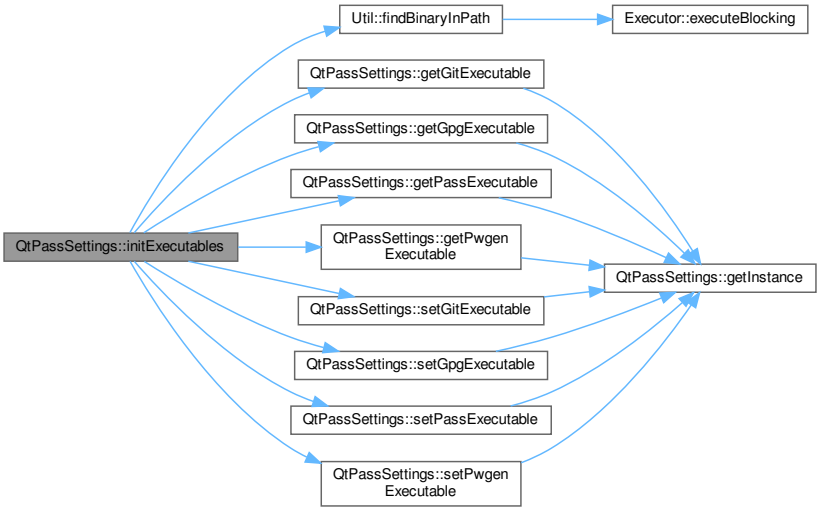
Here is the call graph for this function:

13.20.2.29 initExecutables()

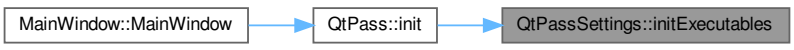
```
void QtPassSettings::initExecutables ( )  [static]
```

Definition at line 335 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



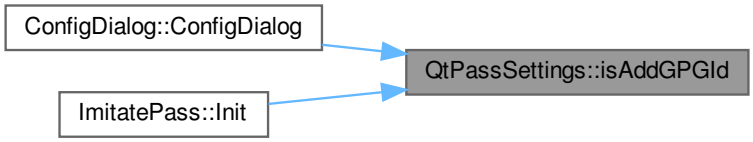
13.20.2.30 isAddGPGId()

```
bool QtPassSettings::isAddGPGId (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 292 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

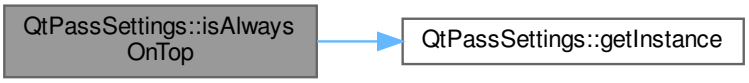


13.20.2.31 isAlwaysOnTop()

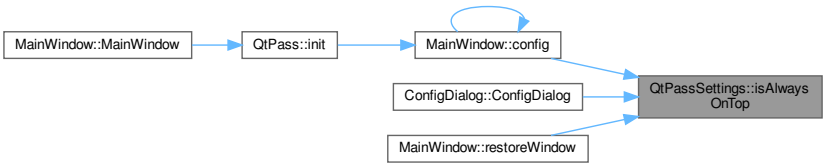
```
bool QtPassSettings::isAlwaysOnTop (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 558 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

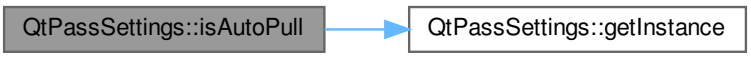


13.20.2.32 isAutoPull()

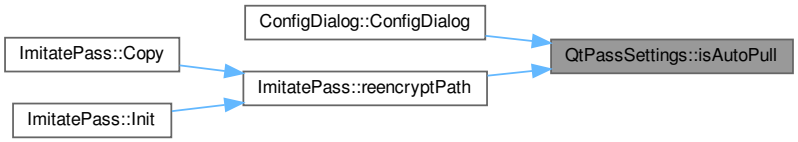
```
bool QtPassSettings::isAutoPull (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 567 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



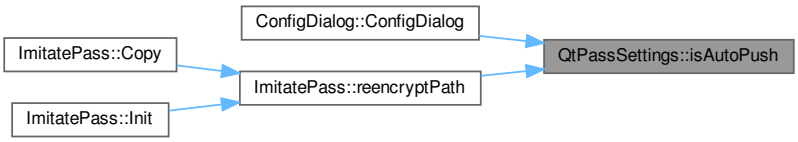
13.20.2.33 isAutoPush()

```
bool QtPassSettings::isAutoPush (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 576 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



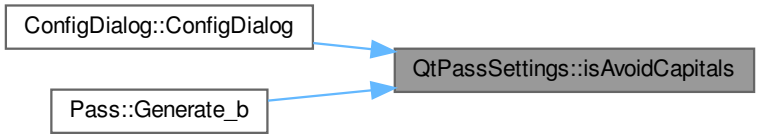
13.20.2.34 isAvoidCapitals()

```
bool QtPassSettings::isAvoidCapitals (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 483 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



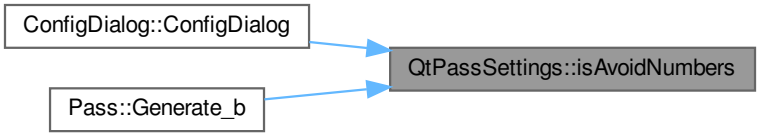
13.20.2.35 isAvoidNumbers()

```
bool QtPassSettings::isAvoidNumbers (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 492 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



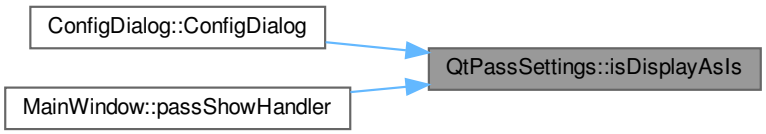
13.20.2.36 isDisplayAsIs()

```
bool QtPassSettings::isDisplayAsIs (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 274 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



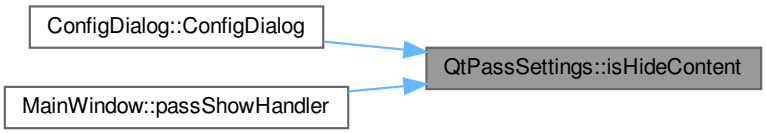
13.20.2.37 isHideContent()

```
bool QtPassSettings::isHideContent (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 256 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

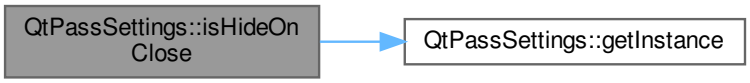


13.20.2.38 isHideOnClose()

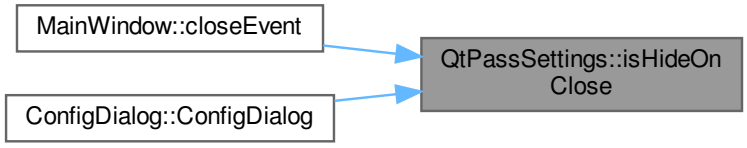
```
bool QtPassSettings::isHideOnClose (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 540 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.20.2.39 isHidePassword()

```
bool QtPassSettings::isHidePassword (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 247 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

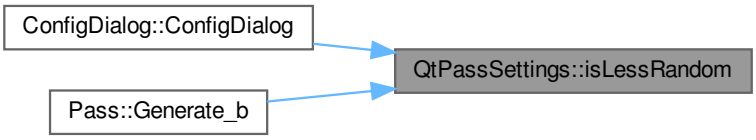
13.20.2.40 isLessRandom()

```
bool QtPassSettings::isLessRandom (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 501 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.20.2.41 isMaximized()

```
bool QtPassSettings::isMaximized (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 160 of file qtpasssettings.cpp.

Here is the call graph for this function:

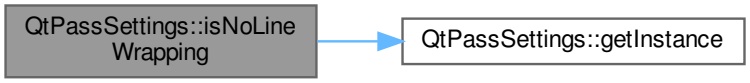
Here is the caller graph for this function:

13.20.2.42 isNoLineWrapping()

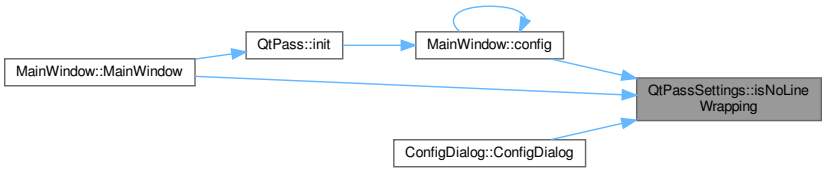
```
bool QtPassSettings::isNoLineWrapping (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 283 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



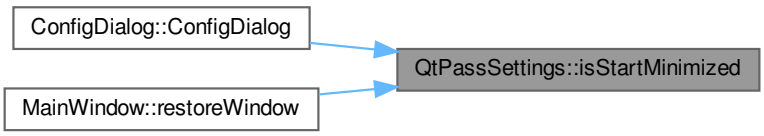
13.20.2.43 isStartMinimized()

```
bool QtPassSettings::isStartMinimized (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 549 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

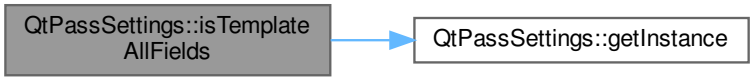


13.20.2.44 isTemplateAllFields()

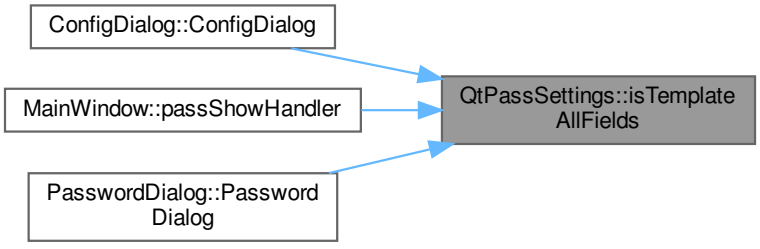
```
bool QtPassSettings::isTemplateAllFields (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 603 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



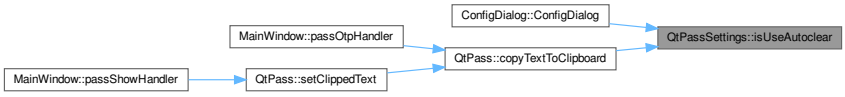
13.20.2.45 isUseAutoclear()

```
bool QtPassSettings::isUseAutoclear (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 207 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

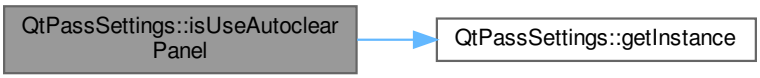


13.20.2.46 isUseAutoclearPanel()

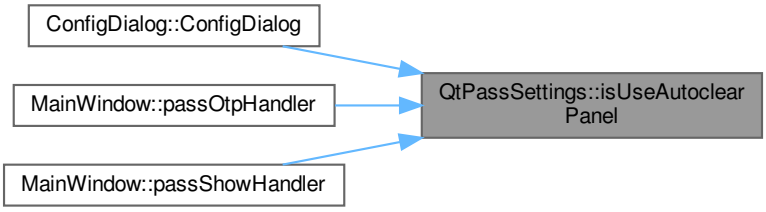
```
bool QtPassSettings::isUseAutoclearPanel (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 226 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

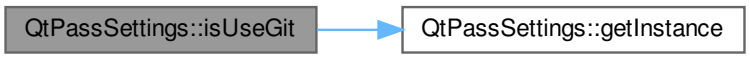


13.20.2.47 isUseGit()

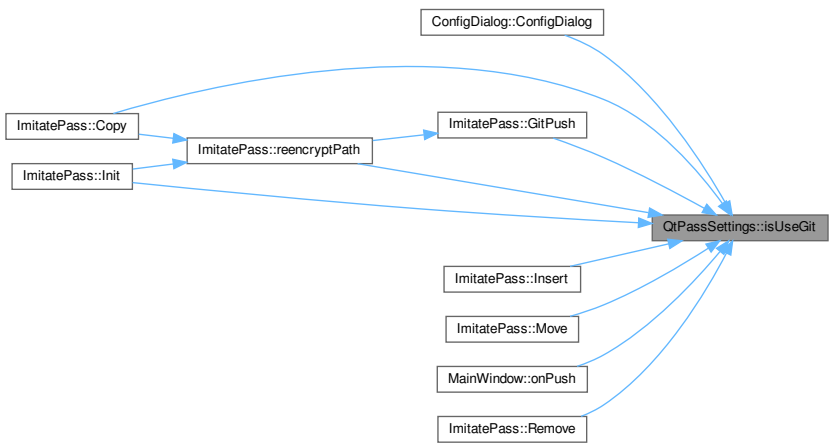
```
bool QtPassSettings::isUseGit (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 439 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



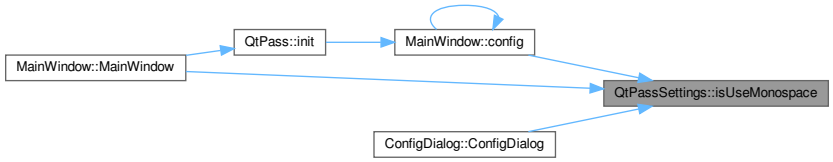
13.20.2.48 isUseMonospace()

```
bool QtPassSettings::isUseMonospace (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 265 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

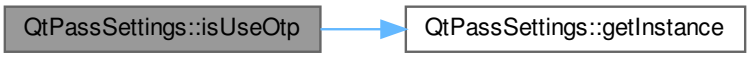


13.20.2.49 isUseOtp()

```
bool QtPassSettings::isUseOtp (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 446 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

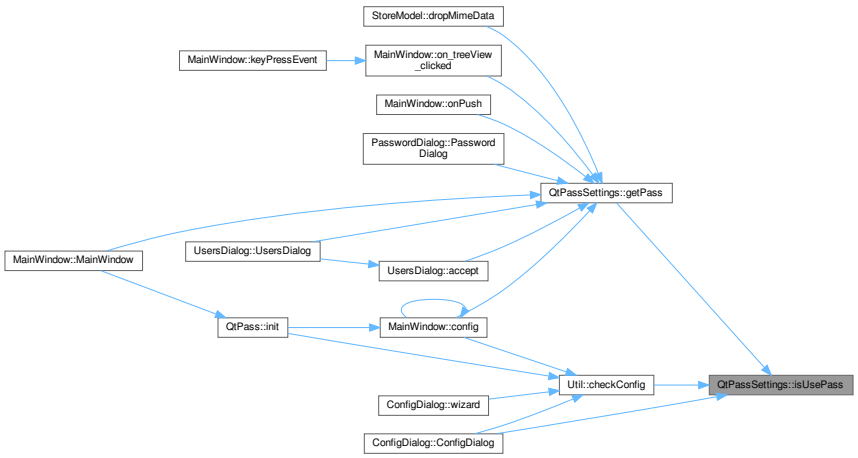
13.20.2.50 isUsePass()

```
bool QtPassSettings::isUsePass (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 169 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



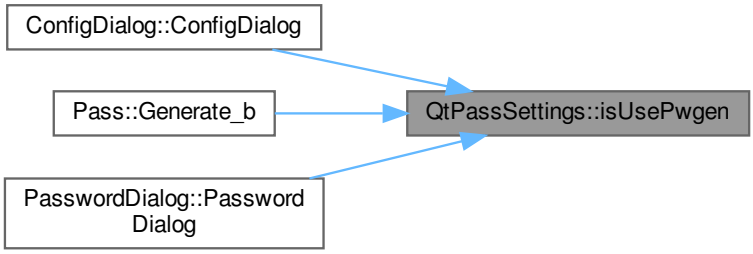
13.20.2.51 isUsePwgen()

```
bool QtPassSettings::isUsePwgen (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 474 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



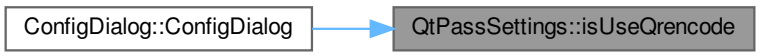
13.20.2.52 isUseQrencode()

```
bool QtPassSettings::isUseQrencode (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 454 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



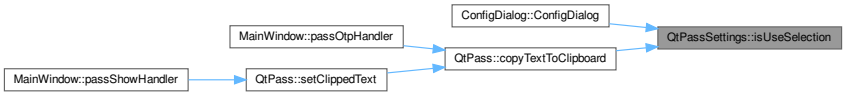
13.20.2.53 isUseSelection()

```
bool QtPassSettings::isUseSelection (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 198 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



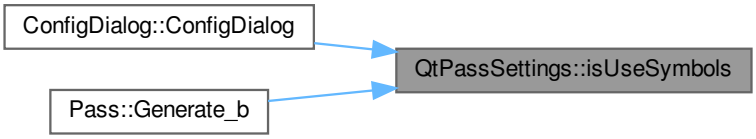
13.20.2.54 isUseSymbols()

```
bool QtPassSettings::isUseSymbols (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 510 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



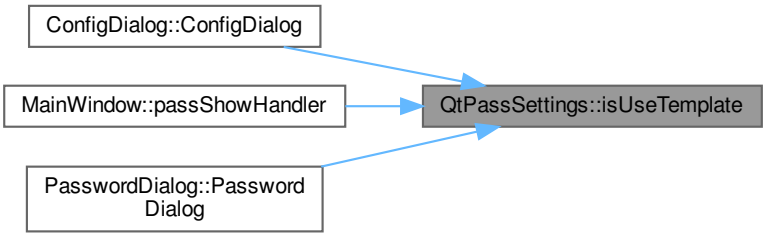
13.20.2.55 isUseTemplate()

```
bool QtPassSettings::isUseTemplate (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 594 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



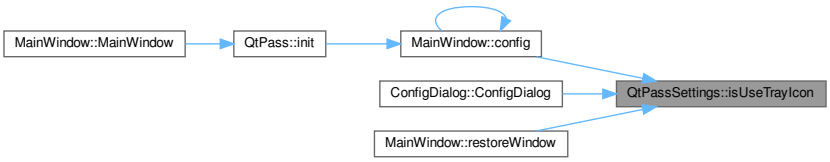
13.20.2.56 isUseTrayIcon()

```
bool QtPassSettings::isUseTrayIcon (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 531 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



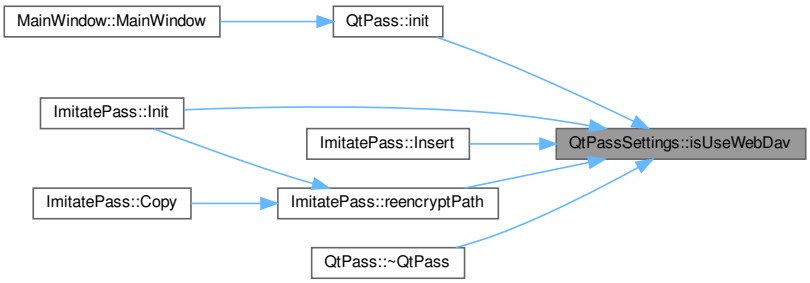
13.20.2.57 isUseWebDav()

```
bool QtPassSettings::isUseWebDav (
            const bool & defaultValue = QVariant().toBool() )  [static]
```

Definition at line 394 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.20.2.58 setAddGPGId()

```
void QtPassSettings::setAddGPGId (
            const bool & addGPGId )  [static]
```

Definition at line 297 of file qtpasssettings.cpp.

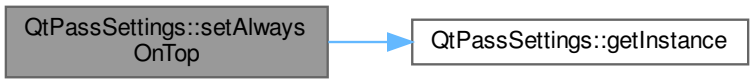
Here is the call graph for this function:

13.20.2.59 setAlwaysOnTop()

```
void QtPassSettings::setAlwaysOnTop (
            const bool & alwaysOnTop )  [static]
```

Definition at line 563 of file qtpasssettings.cpp.

Here is the call graph for this function:

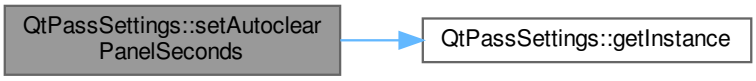


13.20.2.60 setAutoclearPanelSeconds()

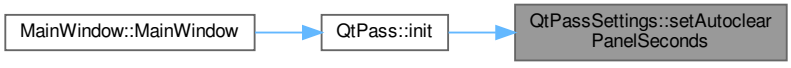
```
void QtPassSettings::setAutoclearPanelSeconds (
            const int & autoClearPanelSeconds )  [static]
```

Definition at line 241 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

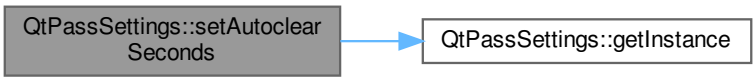


13.20.2.61 setAutoclearSeconds()

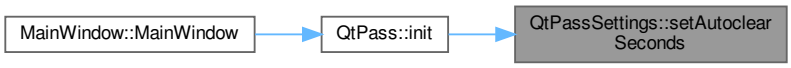
```
void QtPassSettings::setAutoclearSeconds (
            const int & autoClearSeconds )  [static]
```

Definition at line 221 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

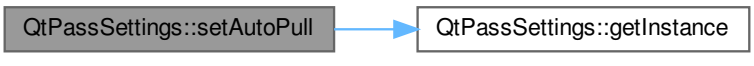


13.20.2.62 setAutoPull()

```
void QtPassSettings::setAutoPull (
            const bool & autoPull )  [static]
```

Definition at line 572 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.63 setAutoPush()

```
void QtPassSettings::setAutoPush (
            const bool & autoPush )  [static]
```

Definition at line 581 of file qtpasssettings.cpp.

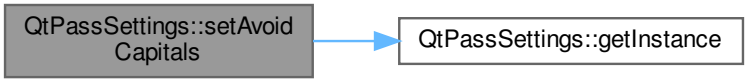
Here is the call graph for this function:

13.20.2.64 setAvoidCapitals()

```
void QtPassSettings::setAvoidCapitals (
            const bool & avoidCapitals )  [static]
```

Definition at line 488 of file qtpasssettings.cpp.

Here is the call graph for this function:

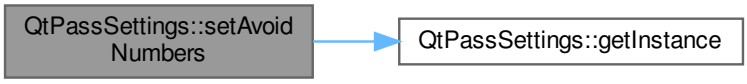


13.20.2.65 setAvoidNumbers()

```
void QtPassSettings::setAvoidNumbers (
            const bool & avoidNumbers )  [static]
```

Definition at line 497 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.66 setClipBoardType()

```
void QtPassSettings::setClipBoardType (
            const int & clipBoardType )  [static]
```

Definition at line 194 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.67 setDisplayAsIs()

```
void QtPassSettings::setDisplayAsIs (
            const bool & displayAsIs )  [static]
```

Definition at line 279 of file qtpasssettings.cpp.

Here is the call graph for this function:

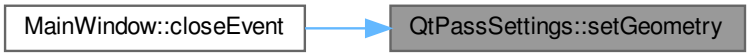
13.20.2.68 setGeometry()

```
void QtPassSettings::setGeometry (
            const QByteArray & geometry )  [static]
```

Definition at line 133 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



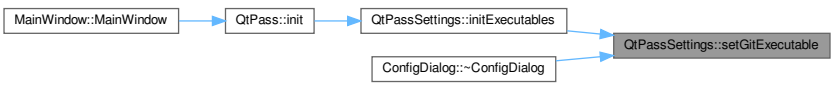
13.20.2.69 setGitExecutable()

```
void QtPassSettings::setGitExecutable (
            const QString & gitExecutable )  [static]
```

Definition at line 366 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



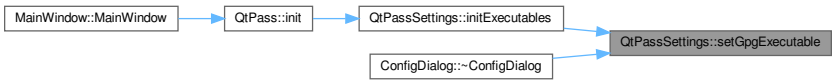
13.20.2.70 setGpgExecutable()

```
void QtPassSettings::setGpgExecutable (
            const QString & gpgExecutable )  [static]
```

Definition at line 375 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.20.2.71 setHideContent()

```
void QtPassSettings::setHideContent (
            const bool & hideContent )  [static]
```

Definition at line 261 of file qtpasssettings.cpp.

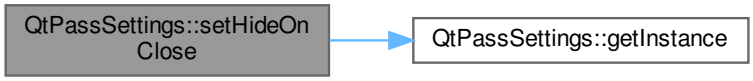
Here is the call graph for this function:

13.20.2.72 setHideOnClose()

```
void QtPassSettings::setHideOnClose (
            const bool & hideOnClose )  [static]
```

Definition at line 545 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.73 setHidePassword()

```
void QtPassSettings::setHidePassword (
            const bool & hidePassword )  [static]
```

Definition at line 252 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.74 setLessRandom()

```
void QtPassSettings::setLessRandom (
            const bool & lessRandom )  [static]
```

Definition at line 506 of file qtpasssettings.cpp.

Here is the call graph for this function:

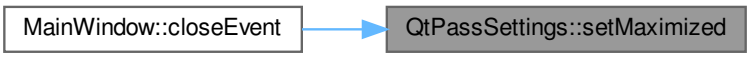
13.20.2.75 setMaximized()

```
void QtPassSettings::setMaximized (
            const bool & maximized )  [static]
```

Definition at line 165 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

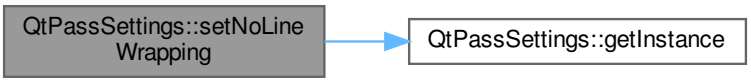


13.20.2.76 setNoLineWrapping()

```
void QtPassSettings::setNoLineWrapping (
            const bool & noLineWrapping )  [static]
```

Definition at line 288 of file qtpasssettings.cpp.

Here is the call graph for this function:

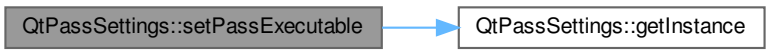


13.20.2.77 setPassExecutable()

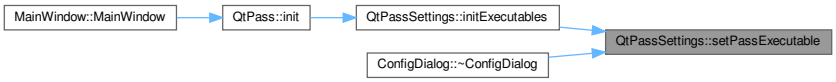
```
void QtPassSettings::setPassExecutable (
            const QString & passExecutable )  [static]
```

Definition at line 357 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

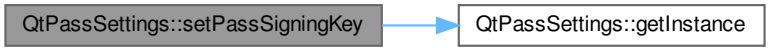


13.20.2.78 setPassSigningKey()

```
void QtPassSettings::setPassSigningKey (
            const QString & passSigningKey )  [static]
```

Definition at line 331 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.79 setPassStore()

```
void QtPassSettings::setPassStore (
            const QString & passStore )  [static]
```

Definition at line 322 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

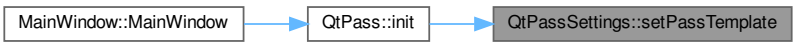
13.20.2.80 setPassTemplate()

```
void QtPassSettings::setPassTemplate (
            const QString & passTemplate )  [static]
```

Definition at line 590 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

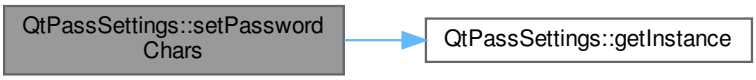


13.20.2.81 setPasswordChars()

```
void QtPassSettings::setPasswordChars (
            const QString & passwordChars )  [static]
```

Definition at line 527 of file qtpasssettings.cpp.

Here is the call graph for this function:

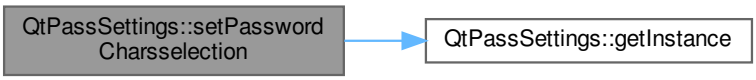


13.20.2.82 setPasswordCharsselection()

```
void QtPassSettings::setPasswordCharsselection (
            const int & passwordCharsselection )  [static]
```

Definition at line 522 of file qtpasssettings.cpp.

Here is the call graph for this function:

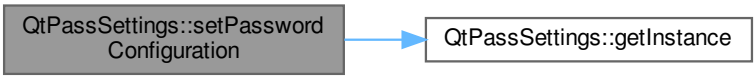


13.20.2.83 setPasswordConfiguration()

```
void QtPassSettings::setPasswordConfiguration (
            const PasswordConfiguration & config )  [static]
```

Definition at line 52 of file qtpasssettings.cpp.

Here is the call graph for this function:

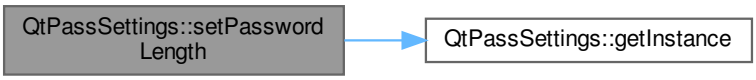


13.20.2.84 setPasswordLength()

```
void QtPassSettings::setPasswordLength (
            const int & passwordLength )  [static]
```

Definition at line 519 of file qtpasssettings.cpp.

Here is the call graph for this function:

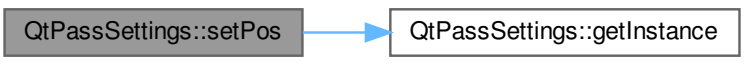


13.20.2.85 setPos()

```
void QtPassSettings::setPos (
            const QPoint & pos )  [static]
```

Definition at line 149 of file qtpasssettings.cpp.

Here is the call graph for this function:



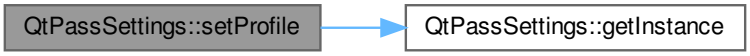
Here is the caller graph for this function:

13.20.2.86 setProfile()

```
void QtPassSettings::setProfile (
            const QString & profile )  [static]
```

Definition at line 435 of file qtpasssettings.cpp.

Here is the call graph for this function:

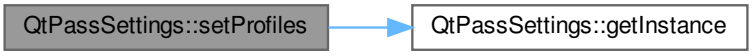


13.20.2.87 setProfiles()

```
void QtPassSettings::setProfiles (
            const QHash< QString, QHash< QString, QString > > & profiles )  [static]
```

Definition at line 92 of file qtpasssettings.cpp.

Here is the call graph for this function:

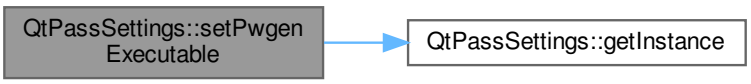


13.20.2.88 setPwgenExecutable()

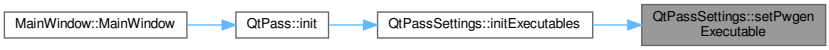
```
void QtPassSettings::setPwgenExecutable (
            const QString & pwgenExecutable )  [static]
```

Definition at line 384 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

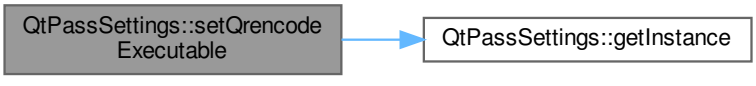


13.20.2.89 setQrencodeExecutable()

```
void QtPassSettings::setQrencodeExecutable (
            const QString & qrencodeExecutable )  [static]
```

Definition at line 469 of file qtpasssettings.cpp.

Here is the call graph for this function:



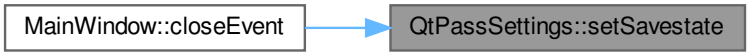
13.20.2.90 setSavestate()

```
void QtPassSettings::setSavestate (
            const QByteArray & saveState )  [static]
```

Definition at line 142 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

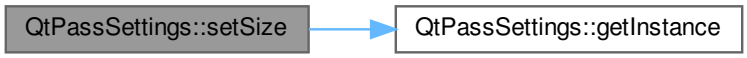


13.20.2.91 setSize()

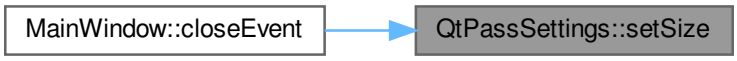
```
void QtPassSettings::setSize (
            const QSize & size )  [static]
```

Definition at line 156 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

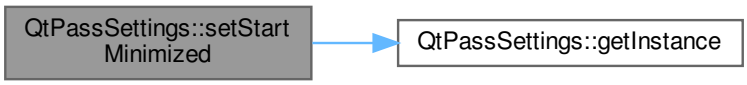


13.20.2.92 setStartMinimized()

```
void QtPassSettings::setStartMinimized (
            const bool & startMinimized )  [static]
```

Definition at line 554 of file qtpasssettings.cpp.

Here is the call graph for this function:

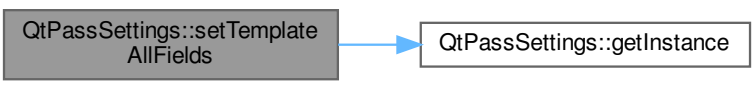


13.20.2.93 setTemplateAllFields()

```
void QtPassSettings::setTemplateAllFields (
            const bool & templateAllFields )  [static]
```

Definition at line 608 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.94 setUseAutoclear()

```
void QtPassSettings::setUseAutoclear (
            const bool & useAutoclear )  [static]
```

Definition at line 212 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.95 setUseAutoclearPanel()

```
void QtPassSettings::setUseAutoclearPanel (
            const bool & useAutoclearPanel )  [static]
```

Definition at line 231 of file qtpasssettings.cpp.

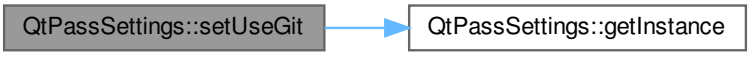
Here is the call graph for this function:

13.20.2.96 setUseGit()

```
void QtPassSettings::setUseGit (
            const bool & useGit )  [static]
```

Definition at line 442 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.97 setUseMonospace()

```
void QtPassSettings::setUseMonospace (
            const bool & useMonospace )  [static]
```

Definition at line 270 of file qtpasssettings.cpp.

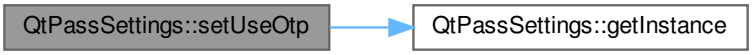
Here is the call graph for this function:

13.20.2.98 setUseOtp()

```
void QtPassSettings::setUseOtp (
            const bool & useOtp )  [static]
```

Definition at line 450 of file qtpasssettings.cpp.

Here is the call graph for this function:

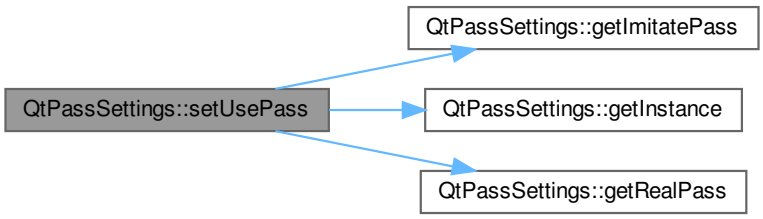


13.20.2.99 setUsePass()

```
void QtPassSettings::setUsePass (
            const bool & usePass )  [static]
```

Definition at line 174 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

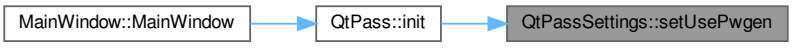
13.20.2.100 setUsePwgen()

```
void QtPassSettings::setUsePwgen (
            const bool & usePwgen )  [static]
```

Definition at line 479 of file qtpasssettings.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.20.2.101 setUseQrencode()

```
void QtPassSettings::setUseQrencode (
            const bool & useQrencode )  [static]
```

Definition at line 460 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.102 setUseSelection()

```
void QtPassSettings::setUseSelection (
            const bool & useSelection )  [static]
```

Definition at line 203 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.103 setUseSymbols()

```
void QtPassSettings::setUseSymbols (
            const bool & useSymbols )  [static]
```

Definition at line 515 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.104 setUseTemplate()

```
void QtPassSettings::setUseTemplate (
            const bool & useTemplate )  [static]
```

Definition at line 599 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.105 setUseTrayIcon()

```
void QtPassSettings::setUseTrayIcon (
            const bool & useTrayIcon )  [static]
```

Definition at line 536 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.106 setUseWebDav()

```
void QtPassSettings::setUseWebDav (
            const bool & useWebDav )  [static]
```

Definition at line 399 of file qtpasssettings.cpp.

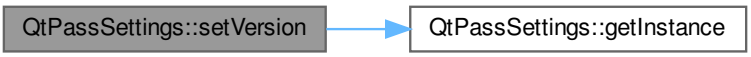
Here is the call graph for this function:

13.20.2.107 setVersion()

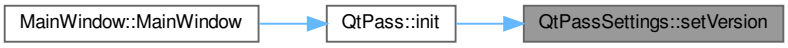
```
void QtPassSettings::setVersion (
            const QString & version )  [static]
```

Definition at line 124 of file qtpasssettings.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

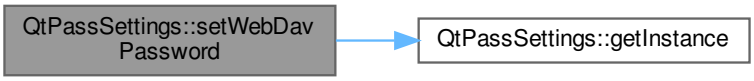


13.20.2.108 setWebDavPassword()

```
void QtPassSettings::setWebDavPassword (
            const QString & webDavPassword )  [static]
```

Definition at line 426 of file qtpasssettings.cpp.

Here is the call graph for this function:



13.20.2.109 setWebDavUrl()

```
void QtPassSettings::setWebDavUrl (
            const QString & webDavUrl )  [static]
```

Definition at line 408 of file qtpasssettings.cpp.

Here is the call graph for this function:

13.20.2.110 setWebDavUser()

```
void QtPassSettings::setWebDavUser (
            const QString & webDavUser )  [static]
```

Definition at line 417 of file qtpasssettings.cpp.

Here is the call graph for this function:

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

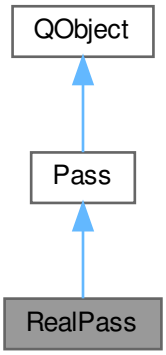
- src/qtpasssettings.h
- src/qtpasssettings.cpp

13.21 RealPass Class Reference

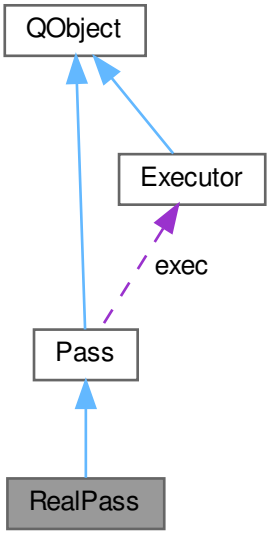
Wrapper for executing pass to handle the password-store.

`#include <realpass.h>`

Inheritance diagram for RealPass:



Collaboration diagram for RealPass:



Public Member Functions

- RealPass ()
- virtual ~RealPass ()
- virtual void GitInit () Q_DECL_OVERRIDE

  RealPass::GitInit pass git init wrapper.
- virtual void GitPull () Q_DECL_OVERRIDE

  RealPass::GitPull pass git pull wrapper.
- virtual void GitPull_b () Q_DECL_OVERRIDE

  RealPass::GitInit pass git pull wrapper which blocks until process finishes.
- virtual void GitPush () Q_DECL_OVERRIDE

  RealPass::GitPush pass git push wrapper.
- virtual void Show (QString file) Q_DECL_OVERRIDE

  RealPass::Show pass show.
- virtual void OtpGenerate (QString file) Q_DECL_OVERRIDE

  RealPass::OtpGenerate pass otp.
- virtual void Insert (QString file, QString newValue, bool overwrite=false) Q_DECL_OVERRIDE

  RealPass::Insert pass insert.
- virtual void Remove (QString file, bool isDir=false) Q_DECL_OVERRIDE

  RealPass::Remove pass remove wrapper.
- virtual void Init (QString path, const QList< UserInfo > &users) Q_DECL_OVERRIDE

  RealPass::Init initialize pass repository.
- void Move (const QString src, const QString dest, const bool force=false) Q_DECL_OVERRIDE

  RealPass::Move move a file (or folder)
- void Copy (const QString src, const QString dest, const bool force=false) Q_DECL_OVERRIDE

  RealPass::Copy copy a file (or folder)

Public Member Functions inherited from Pass

- Pass ()

  Pass::Pass wrapper for using either pass or the pass imitation.
- void init ()
- virtual ~Pass ()
- virtual void GitInit ()=0
- virtual void GitPull ()=0
- virtual void GitPull_b ()=0
- virtual void GitPush ()=0
- virtual void Show (QString file)=0
- virtual void OtpGenerate (QString file)=0
- virtual void Insert (QString file, QString value, bool force)=0
- virtual void Remove (QString file, bool isDir)=0
- virtual void Move (const QString srcDir, const QString dest, const bool force=false)=0
- virtual void Copy (const QString srcDir, const QString dest, const bool force=false)=0
- virtual void Init (QString path, const QList< UserInfo > &users)=0
- virtual QString Generate_b (unsigned int length, const QString &charset)

  Pass::Generate use either pwgen or internal password generator.
- void GenerateGPGKeys (QString batch)

  Pass::GenerateGPGKeys internal gpg keypair generator . .
- QList< UserInfo > listKeys (QStringList keystrings, bool secret=false)

  Pass::listKeys list users.
- QList< UserInfo > listKeys (QString keystring="", bool secret=false)

  Pass::listKeys list users.
- void updateEnv ()

  Pass::updateEnv update the execution environment (used when switching profiles)

Additional Inherited Members

Signals inherited from Pass

- void error (QProcess::ProcessError)
- void startingExecuteWrapper ()
- void statusMsg (QString, int)
- void critical (QString, QString)
- void processErrorExit (int exitCode, const QString &err)
- void finishedAny (const QString &, const QString &)
- void finishedGitInit (const QString &, const QString &)
- void finishedGitPull (const QString &, const QString &)
- void finishedGitPush (const QString &, const QString &)
- void finishedShow (const QString &)
- void finishedOtpGenerate (const QString &)
- void finishedInsert (const QString &, const QString &)
- void finishedRemove (const QString &, const QString &)
- void finishedInit (const QString &, const QString &)
- void finishedMove (const QString &, const QString &)
- void finishedCopy (const QString &, const QString &)
- void finishedGenerate (const QString &, const QString &)
- void finishedGenerateGPGKeys (const QString &, const QString &)

Static Public Member Functions inherited from Pass

- static QString getGpgIdPath (QString for_file)

  Pass::getGpgIdPath return gpgid file path for some file (folder).
- static QStringList getRecipientList (QString for_file)

  Pass::getRecipientList return list of gpg-id's to encrypt for.
- static QStringList getRecipientString (QString for_file, QString separator=" ", int *count=NULL)

  Pass::getRecipientString formated string for use with GPG.

Protected Types inherited from Pass

- typedef Enums::PROCESS PROCESS

Protected Slots inherited from Pass

- virtual void finished (int id, int exitCode, const QString &out, const QString &err)

  Pass::processFinished reemits specific signal based on what process has finished.

Protected Member Functions inherited from Pass

- void executeWrapper (PROCESS id, const QString &app, const QStringList &args, bool readStdout=true, bool readStderr=true)
- QString generateRandomPassword (const QString &charset, unsigned int length)
- quint32 boundedRandom (quint32 bound)
- virtual void executeWrapper (PROCESS id, const QString &app, const QStringList &args, QString input, bool readStdout=true, bool readStderr=true)

Protected Attributes inherited from Pass

- Executor exec

13.21.1 Detailed Description

Wrapper for executing pass to handle the password-store.

Definition at line 10 of file realpass.h.

13.21.2 Constructor & Destructor Documentation

13.21.2.1 RealPass()

```
RealPass::RealPass ( )  [default]
```

13.21.2.2 ~RealPass()

```
virtual RealPass::~RealPass ( )  [inline], [virtual]
```

Definition at line 17 of file realpass.h.

13.21.3 Member Function Documentation

13.21.3.1 Copy()

```
void RealPass::Copy (
            const QString src,
            const QString dest,
            const bool force = false )  [virtual]
```

RealPass::Copy copy a file (or folder)

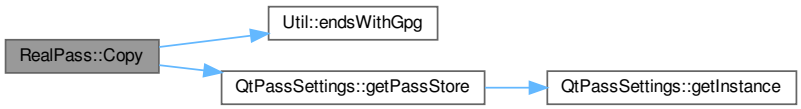
Parameters

src	source file or folder
dest	destination file or folder
force	overwrite

Implements Pass.

Definition at line 143 of file realpass.cpp.

Here is the call graph for this function:



13.21.3.2 GitInit()

```
void RealPass::GitInit ( )  [virtual]
```

RealPass::GitInit pass git init wrapper.

Implements Pass.

Definition at line 17 of file realpass.cpp.

13.21.3.3 GitPull()

```
void RealPass::GitPull ( )  [virtual]
```

RealPass::GitPull pass git pull wrapper.

Implements Pass.

Definition at line 30 of file realpass.cpp.

13.21.3.4 GitPull_b()

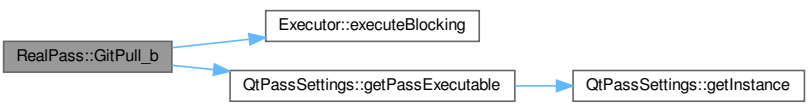
```
void RealPass::GitPull_b ( )  [virtual]
```

RealPass::GitInit pass git pull wrapper which blocks until process finishes.

Implements Pass.

Definition at line 23 of file realpass.cpp.

Here is the call graph for this function:



13.21.3.5 GitPush()

```
void RealPass::GitPush ( )  [virtual]
```

RealPass::GitPush pass git push wrapper.

Implements Pass.

Definition at line 35 of file realpass.cpp.

13.21.3.6 Init()

```
void RealPass::Init (
            QString path,
            const QList< UserInfo > & users )  [virtual]
```

RealPass::Init initialize pass repository.

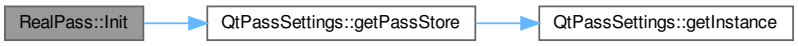
Parameters

path	Absolute path to new password-store
users	list of users with ability to decrypt new password-store

Implements Pass.

Definition at line 82 of file realpass.cpp.

Here is the call graph for this function:



13.21.3.7 Insert()

```
void RealPass::Insert (
            QString file,
            QString newValue,
            bool overwrite = false )  [virtual]
```

RealPass::Insert pass insert.

Implements Pass.

Definition at line 61 of file realpass.cpp.

13.21.3.8 Move()

```
void RealPass::Move (
            const QString src,
            const QString dest,
            const bool force = false )  [virtual]
```

RealPass::Move move a file (or folder)

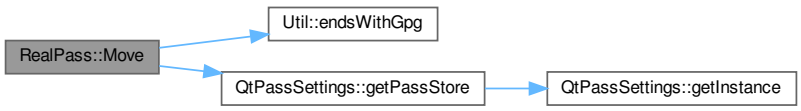
Parameters

src	source file or folder
dest	destination file or folder
force	overwrite

Implements Pass.

Definition at line 102 of file realpass.cpp.

Here is the call graph for this function:



13.21.3.9 OtpGenerate()

```
void RealPass::OtpGenerate (
            QString file )  [virtual]
```

RealPass::OtpGenerate pass otp.

Parameters

file	file containig OTP uri

Implements Pass.

Definition at line 54 of file realpass.cpp.

13.21.3.10 Remove()

```
void RealPass::Remove (
            QString file,
            bool isDir = false )  [virtual]
```

RealPass::Remove pass remove wrapper.

Implements Pass.

Definition at line 72 of file realpass.cpp.

13.21.3.11 Show()

```
void RealPass::Show (
            QString file )  [virtual]
```

RealPass::Show pass show.

Parameters

file	file to decrypt

Returns

if block is set, returns exit status of internal decryption process otherwise returns QProcess::NormalExit

Implements Pass.

Definition at line 46 of file realpass.cpp.

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

- src/realpass.h
- src/realpass.cpp

13.22 SettingsConstants Class Reference

Table for the naming of configuration items.

```
#include <settingsconstants.h>
```

Static Public Attributes

- static const QString version = "version"
- static const QString groupMainwindow = "mainwindow"
- static const QString geometry
- static const QString savestate
- static const QString pos
- static const QString size
- static const QString splitterLeft
- static const QString splitterRight
- static const QString maximized
- static const QString usePass = "usePass"
- static const QString useAutoclear = "useAutoclear"
- static const QString useSelection = "useSelection"
- static const QString autoclearSeconds = "autoclearSeconds"
- static const QString useAutoclearPanel = "useAutoclearPanel"
- static const QString autoclearPanelSeconds
- static const QString hidePassword = "hidePassword"
- static const QString hideContent = "hideContent"
- static const QString useMonospace = "useMonospace"
- static const QString displayAsIs = "displayAsIs"
- static const QString noLineWrapping = "noLineWrapping"
- static const QString addGPGId = "addGPGId"
- static const QString passStore = "passStore"
- static const QString passSigningKey = "passSigningKey"
- static const QString passExecutable = "passExecutable"
- static const QString gitExecutable = "gitExecutable"
- static const QString gpgExecutable = "gpgExecutable"
- static const QString pwgenExecutable = "pwgenExecutable"
- static const QString gpgHome = "gpgHome"
- static const QString useWebDav = "useWebDav"
- static const QString webDavUrl = "webDavUrl"
- static const QString webDavUser = "webDavUser"
- static const QString webDavPassword = "webDavPassword"
- static const QString profile = "profile"
- static const QString groupProfiles = "profiles"
- static const QString useGit = "useGit"
- static const QString useOtp = "useOtp"
- static const QString useQrencode = "useQrencode"
- static const QString qrencodeExecutable = "qrencodeExecutable"
- static const QString useClipboard = "useClipboard"
- static const QString usePwgen = "usePwgen"
- static const QString avoidCapitals = "avoidCapitals"
- static const QString avoidNumbers = "avoidNumbers"
- static const QString lessRandom = "lessRandom"
- static const QString useSymbols = "useSymbols"
- static const QString passwordLength = "passwordLength"
- static const QString passwordCharsselection
- static const QString passwordChars = "passwordChars"
- static const QString useTrayIcon = "useTrayIcon"
- static const QString hideOnClose = "hideOnClose"
- static const QString startMinimized = "startMinimized"
- static const QString alwaysOnTop = "alwaysOnTop"
- static const QString autoPull = "autoPull"
- static const QString autoPush = "autoPush"
- static const QString passTemplate = "passTemplate"
- static const QString useTemplate = "useTemplate"
- static const QString templateAllFields = "templateAllFields"
- static const QString clipBoardType = "clipBoardType"

13.22.1 Detailed Description

Table for the naming of configuration items.

Definition at line 10 of file settingsconstants.h.

13.22.2 Member Data Documentation

13.22.2.1 addGPGId

```
const QString SettingsConstants::addGPGId = "addGPGId"  [static]
```

Definition at line 32 of file settingsconstants.h.

13.22.2.2 alwaysOnTop

```
const QString SettingsConstants::alwaysOnTop = "alwaysOnTop"  [static]
```

Definition at line 62 of file settingsconstants.h.

13.22.2.3 autoclearPanelSeconds

```
const QString SettingsConstants::autoclearPanelSeconds  [static]
```

Initial value:

```
=
    "autoclearPanelSeconds"
```

Definition at line 26 of file settingsconstants.h.

13.22.2.4 autoclearSeconds

```
const QString SettingsConstants::autoclearSeconds = "autoclearSeconds"  [static]
```

Definition at line 24 of file settingsconstants.h.

13.22.2.5 autoPull

```
const QString SettingsConstants::autoPull = "autoPull"  [static]
```

Definition at line 63 of file settingsconstants.h.

13.22.2.6 autoPush

```
const QString SettingsConstants::autoPush = "autoPush"  [static]
```

Definition at line 64 of file settingsconstants.h.

13.22.2.7 avoidCapitals

```
const QString SettingsConstants::avoidCapitals = "avoidCapitals"  [static]
```

Definition at line 52 of file settingsconstants.h.

13.22.2.8 avoidNumbers

```
const QString SettingsConstants::avoidNumbers = "avoidNumbers"  [static]
```

Definition at line 53 of file settingsconstants.h.

13.22.2.9 clipBoardType

```
const QString SettingsConstants::clipBoardType = "clipBoardType"  [static]
```

Definition at line 68 of file settingsconstants.h.

13.22.2.10 displayAsIs

```
const QString SettingsConstants::displayAsIs = "displayAsIs"  [static]
```

Definition at line 30 of file settingsconstants.h.

13.22.2.11 geometry

```
const QString SettingsConstants::geometry  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/geometry"
```

Definition at line 14 of file settingsconstants.h.

13.22.2.12 gitExecutable

```
const QString SettingsConstants::gitExecutable = "gitExecutable"  [static]
```

Definition at line 36 of file settingsconstants.h.

13.22.2.13 gpgExecutable

```
const QString SettingsConstants::gpgExecutable = "gpgExecutable"  [static]
```

Definition at line 37 of file settingsconstants.h.

13.22.2.14 gpgHome

```
const QString SettingsConstants::gpgHome = "gpgHome"  [static]
```

Definition at line 39 of file settingsconstants.h.

13.22.2.15 groupMainwindow

```
const QString SettingsConstants::groupMainwindow = "mainwindow"  [static]
```

Definition at line 13 of file settingsconstants.h.

13.22.2.16 groupProfiles

```
const QString SettingsConstants::groupProfiles = "profiles"  [static]
```

Definition at line 45 of file settingsconstants.h.

13.22.2.17 hideContent

```
const QString SettingsConstants::hideContent = "hideContent"  [static]
```

Definition at line 28 of file settingsconstants.h.

13.22.2.18 hideOnClose

```
const QString SettingsConstants::hideOnClose = "hideOnClose"  [static]
```

Definition at line 60 of file settingsconstants.h.

13.22.2.19 hidePassword

```
const QString SettingsConstants::hidePassword = "hidePassword"  [static]
```

Definition at line 27 of file settingsconstants.h.

13.22.2.20 lessRandom

```
const QString SettingsConstants::lessRandom = "lessRandom"  [static]
```

Definition at line 54 of file settingsconstants.h.

13.22.2.21 maximized

```
const QString SettingsConstants::maximized  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/maximized"
```

Definition at line 20 of file settingsconstants.h.

13.22.2.22 noLineWrapping

```
const QString SettingsConstants::noLineWrapping = "noLineWrapping"  [static]
```

Definition at line 31 of file settingsconstants.h.

13.22.2.23 passExecutable

```
const QString SettingsConstants::passExecutable = "passExecutable"  [static]
```

Definition at line 35 of file settingsconstants.h.

13.22.2.24 passSigningKey

```
const QString SettingsConstants::passSigningKey = "passSigningKey"  [static]
```

Definition at line 34 of file settingsconstants.h.

13.22.2.25 passStore

```
const QString SettingsConstants::passStore = "passStore"  [static]
```

Definition at line 33 of file settingsconstants.h.

13.22.2.26 passTemplate

```
const QString SettingsConstants::passTemplate = "passTemplate"  [static]
```

Definition at line 65 of file settingsconstants.h.

13.22.2.27 passwordChars

```
const QString SettingsConstants::passwordChars = "passwordChars"  [static]
```

Definition at line 58 of file settingsconstants.h.

13.22.2.28 passwordCharsselection

```
const QString SettingsConstants::passwordCharsselection  [static]
```

Initial value:

```
=
    "passwordCharsselection"
```

Definition at line 57 of file settingsconstants.h.

13.22.2.29 passwordLength

```
const QString SettingsConstants::passwordLength = "passwordLength"  [static]
```

Definition at line 56 of file settingsconstants.h.

13.22.2.30 pos

```
const QString SettingsConstants::pos  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/pos"
```

Definition at line 16 of file settingsconstants.h.

13.22.2.31 profile

```
const QString SettingsConstants::profile = "profile"  [static]
```

Definition at line 44 of file settingsconstants.h.

13.22.2.32 pwgenExecutable

```
const QString SettingsConstants::pwgenExecutable = "pwgenExecutable"  [static]
```

Definition at line 38 of file settingsconstants.h.

13.22.2.33 qrencodeExecutable

```
const QString SettingsConstants::qrencodeExecutable = "qrencodeExecutable"  [static]
```

Definition at line 49 of file settingsconstants.h.

13.22.2.34 savestate

```
const QString SettingsConstants::savestate  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/savestate"
```

Definition at line 15 of file settingsconstants.h.

13.22.2.35 size

```
const QString SettingsConstants::size  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/size"
```

Definition at line 17 of file settingsconstants.h.

13.22.2.36 splitterLeft

```
const QString SettingsConstants::splitterLeft  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/splitterLeft"
```

Definition at line 18 of file settingsconstants.h.

13.22.2.37 splitterRight

```
const QString SettingsConstants::splitterRight  [static]
```

Initial value:

```
=
    SettingsConstants::groupMainwindow + "/splitterRight"
```

Definition at line 19 of file settingsconstants.h.

13.22.2.38 startMinimized

```
const QString SettingsConstants::startMinimized = "startMinimized"  [static]
```

Definition at line 61 of file settingsconstants.h.

13.22.2.39 templateAllFields

```
const QString SettingsConstants::templateAllFields = "templateAllFields"  [static]
```

Definition at line 67 of file settingsconstants.h.

13.22.2.40 useAutoclear

```
const QString SettingsConstants::useAutoclear = "useAutoclear"  [static]
```

Definition at line 22 of file settingsconstants.h.

13.22.2.41 useAutoclearPanel

```
const QString SettingsConstants::useAutoclearPanel = "useAutoclearPanel"  [static]
```

Definition at line 25 of file settingsconstants.h.

13.22.2.42 useClipboard

```
const QString SettingsConstants::useClipboard = "useClipboard"  [static]
```

Definition at line 50 of file settingsconstants.h.

13.22.2.43 useGit

```
const QString SettingsConstants::useGit = "useGit"  [static]
```

Definition at line 46 of file settingsconstants.h.

13.22.2.44 useMonospace

```
const QString SettingsConstants::useMonospace = "useMonospace"  [static]
```

Definition at line 29 of file settingsconstants.h.

13.22.2.45 useOtp

```
const QString SettingsConstants::useOtp = "useOtp"  [static]
```

Definition at line 47 of file settingsconstants.h.

13.22.2.46 usePass

```
const QString SettingsConstants::usePass = "usePass"  [static]
```

Definition at line 21 of file settingsconstants.h.

13.22.2.47 usePwgen

```
const QString SettingsConstants::usePwgen = "usePwgen"  [static]
```

Definition at line 51 of file settingsconstants.h.

13.22.2.48 useQrencode

```
const QString SettingsConstants::useQrencode = "useQrencode"  [static]
```

Definition at line 48 of file settingsconstants.h.

13.22.2.49 useSelection

```
const QString SettingsConstants::useSelection = "useSelection"  [static]
```

Definition at line 23 of file settingsconstants.h.

13.22.2.50 useSymbols

```
const QString SettingsConstants::useSymbols = "useSymbols"  [static]
```

Definition at line 55 of file settingsconstants.h.

13.22.2.51 useTemplate

```
const QString SettingsConstants::useTemplate = "useTemplate"  [static]
```

Definition at line 66 of file settingsconstants.h.

13.22.2.52 useTrayIcon

```
const QString SettingsConstants::useTrayIcon = "useTrayIcon"  [static]
```

Definition at line 59 of file settingsconstants.h.

13.22.2.53 useWebDav

```
const QString SettingsConstants::useWebDav = "useWebDav"  [static]
```

Definition at line 40 of file settingsconstants.h.

13.22.2.54 version

```
const QString SettingsConstants::version = "version"  [static]
```

Definition at line 12 of file settingsconstants.h.

13.22.2.55 webDavPassword

```
const QString SettingsConstants::webDavPassword = "webDavPassword"  [static]
```

Definition at line 43 of file settingsconstants.h.

13.22.2.56 webDavUrl

```
const QString SettingsConstants::webDavUrl = "webDavUrl"  [static]
```

Definition at line 41 of file settingsconstants.h.

13.22.2.57 webDavUser

```
const QString SettingsConstants::webDavUser = "webDavUser"  [static]
```

Definition at line 42 of file settingsconstants.h.

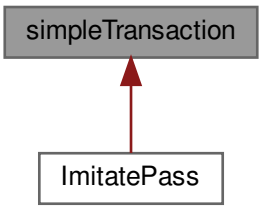
The documentation for this class was generated from the following files:

- src/settingsconstants.h
- src/settingsconstants.cpp

13.23 simpleTransaction Class Reference

`#include <simpletransaction.h>`

Inheritance diagram for simpleTransaction:



Public Member Functions

- simpleTransaction ()
- void transactionStart ()

  transactionStart this function is used to mark start of the sequence of processes that shall be treated as one operation.
- void transactionAdd (Enums::PROCESS)

  transactionAdd If called after call to transactionStart() and before transactionEnd(), this method marks given process as next step in transaction. Otherwise it marks given process as the only step in transaction(it's value is treated as transaction result).
- void transactionEnd (Enums::PROCESS)

  transactionEnd marks end of transaction
- Enums::PROCESS transactionIsOver (Enums::PROCESS)

  transactionIsOver checks wheather currently finished process is last in current transaction

13.23.1 Detailed Description

Definition at line 7 of file simpletransaction.h.

13.23.2 Constructor & Destructor Documentation

13.23.2.1 simpleTransaction()

`simpleTransaction::simpleTransaction ( )  [inline]`

Definition at line 13 of file simpletransaction.h.

13.23.3 Member Function Documentation

13.23.3.1 transactionAdd()

```
void simpleTransaction::transactionAdd (
            Enums::PROCESS id )
```

transactionAdd If called after call to transactionStart() and before transactionEnd(), this method marks given process as next step in transaction. Otherwise it marks given process as the only step in transaction(it's value is treated as transaction result).

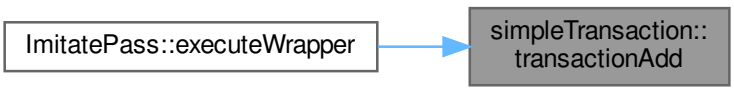
simpleTransaction::transactionAdd

Parameters

id	process that shall be treated as part of transaction
id	

Definition at line 25 of file simpletransaction.cpp.

Here is the caller graph for this function:



13.23.3.2 transactionEnd()

```
void simpleTransaction::transactionEnd (
            Enums::PROCESS pid )
```

transactionEnd marks end of transaction

simpleTransaction::transactionEnd

Parameters

pid	value that will be used as a result of transaction
pid	

Definition at line 40 of file simpletransaction.cpp.

13.23.3.3 transactionIsOver()

```
PROCESS simpleTransaction::transactionIsOver (
            Enums::PROCESS id )
```

transactionIsOver checks wheather currently finished process is last in current transaction

simpleTransaction::transactionIsOver

Returns

result of transaction as set by transactionAdd or transactionEnd if the transaction is over or PROCESS::↩ INVALID if it's not yet over

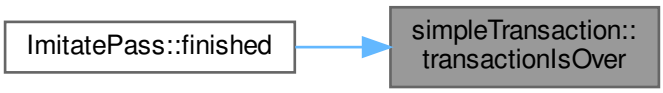
Parameters

id	

Returns

Definition at line 58 of file simpletransaction.cpp.

Here is the caller graph for this function:



13.23.3.4 transactionStart()

```
void simpleTransaction::transactionStart ( )
```

transactionStart this function is used to mark start of the sequence of processes that shall be treated as one operation.

simpleTransaction::transactionStart

Definition at line 14 of file simpletransaction.cpp.

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

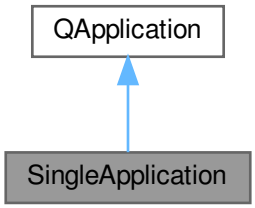
- src/simpletransaction.h
- src/simpletransaction.cpp

13.24 SingleApplication Class Reference

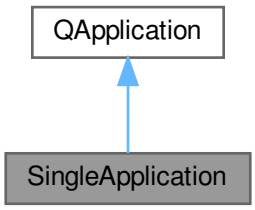
The SingleApplication class is used for commandline intergration.

`#include <singleapplication.h>`

Inheritance diagram for SingleApplication:



Collaboration diagram for SingleApplication:



Public Slots

- void receiveMessage ()

  SingleApplication::receiveMessage we have received (a command line) message.

Signals

- void messageAvailable (QString message)

  messageAvailable notification from commandline

Public Member Functions

- SingleApplication (int &argc, char ∗argv[ ], QString uniqueKey)

  SingleApplication::SingleApplication this replaces the QApplication allowing for local socket based communications.
- bool isRunning ()

  SingleApplication::isRunning is there already a QtPass instance running, to check wether to be server or client.
- bool sendMessage (const QString &message)

  SingleApplication::sendMessage send a message (from commandline) to an already running QtPass instance.

13.24.1 Detailed Description

The SingleApplication class is used for commandline intergration.

This class needs a bit of work or possibly replacement.

Definition at line 14 of file singleapplication.h.

13.24.2 Constructor & Destructor Documentation

13.24.2.1 SingleApplication()

```
SingleApplication::SingleApplication (
            int & argc,
            char * argv[],
            QString uniqueKey )
```

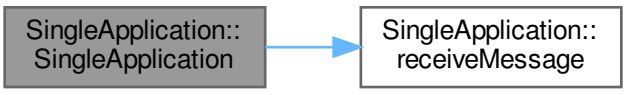
SingleApplication::SingleApplication this replaces the QApplication allowing for local socket based communications.

Parameters

argc	
argv	
uniqueKey	

Definition at line 15 of file singleapplication.cpp.

Here is the call graph for this function:



13.24.3 Member Function Documentation

13.24.3.1 isRunning()

```
bool SingleApplication::isRunning ( )
```

SingleApplication::isRunning is there already a QtPass instance running, to check wether to be server or client.

Returns

Definition at line 64 of file singleapplication.cpp.

Here is the caller graph for this function:

13.24.3.2 messageAvailable

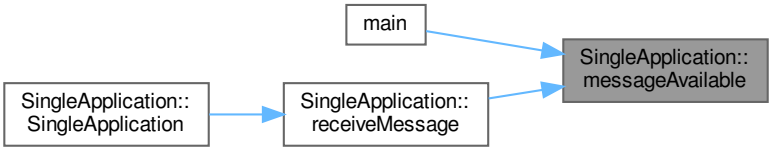
```
void SingleApplication::messageAvailable (
            QString message )  [signal]
```

messageAvailable notification from commandline

Parameters

message	args sent to qtpass executable

Here is the caller graph for this function:



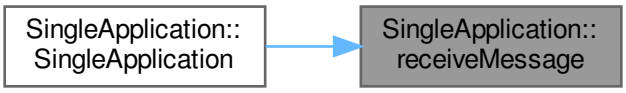
13.24.3.3 receiveMessage

```
void SingleApplication::receiveMessage ( )  [slot]
```

SingleApplication::receiveMessage we have received (a command line) message.

Definition at line 44 of file singleapplication.cpp.

Here is the caller graph for this function:



13.24.3.4 sendMessage()

```
bool SingleApplication::sendMessage (
            const QString & message )
```

SingleApplication::sendMessage send a message (from commandline) to an already running QtPass instance.

Parameters

message	

Returns

Definition at line 72 of file singleapplication.cpp.

Here is the caller graph for this function:

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

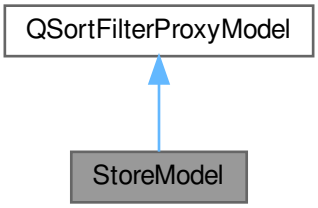
- src/singleapplication.h
- src/singleapplication.cpp

13.25 StoreModel Class Reference

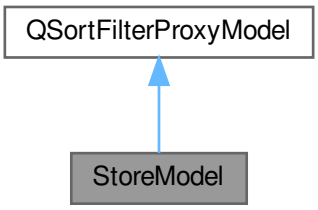
The QSortFilterProxyModel for handling filesystem searches.

```
#include <storemodel.h>
```

Inheritance diagram for StoreModel:



Collaboration diagram for StoreModel:



Public Member Functions

- StoreModel ()

  StoreModel::StoreModel SubClass of QSortFilterProxyModel via http://www.qtcentre.org/threads/46471-QTreeView-Filter.
- bool filterAcceptsRow (int, const QModelIndex &) const override

  StoreModel::filterAcceptsRow should row be shown, wrapper for StoreModel::ShowThis method.
- bool ShowThis (const QModelIndex) const

  StoreModel::ShowThis should a row be shown, based on our search criteria.
- void setModelAndStore (QFileSystemModel ∗sourceModel, QString passStore)

  StoreModel::setModelAndStore update the source model and store.
- QVariant data (const QModelIndex &index, int role) const override

  StoreModel::data don't show the .gpg at the end of a file.
- bool lessThan (const QModelIndex &source_left, const QModelIndex &source_right) const override

  StoreModel::lessThan.
- Qt::DropActions supportedDropActions () const override

  StoreModel::supportedDropActions enable drop.
- Qt::DropActions supportedDragActions () const override

  StoreModel::supportedDragActions enable drag.
- Qt::ItemFlags flags (const QModelIndex &index) const override

  StoreModel::flags.
- QStringList mimeTypes () const override

  StoreModel::mimeTypes.
- QMimeData ∗ mimeData (const QModelIndexList &indexes) const override

  StoreModel::mimeData.
- bool canDropMimeData (const QMimeData ∗data, Qt::DropAction action, int row, int column, const QModelIndex &parent) const override

  StoreModel::canDropMimeData.
- bool dropMimeData (const QMimeData ∗data, Qt::DropAction action, int row, int column, const QModelIndex &parent) override

  StoreModel::dropMimeData.

13.25.1 Detailed Description

The QSortFilterProxyModel for handling filesystem searches.

Definition at line 11 of file storemodel.h.

13.25.2 Constructor & Destructor Documentation

13.25.2.1 StoreModel()

```
StoreModel::StoreModel ( )
```

StoreModel::StoreModel SubClass of QSortFilterProxyModel via http://www.qtcentre.org/threads/46471-QTreeView-Filter.

Definition at line 34 of file storemodel.cpp.

13.25.3 Member Function Documentation

13.25.3.1 canDropMimeData()

```
bool StoreModel::canDropMimeData (
            const QMimeData * data,
            Qt::DropAction action,
            int row,
            int column,
            const QModelIndex & parent ) const  [override]
```

StoreModel::canDropMimeData.

Parameters

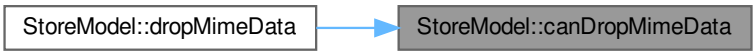
data	
action	
row	
column	
parent	

Returns

Definition at line 189 of file storemodel.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:



13.25.3.2 data()

```
QVariant StoreModel::data (
            const QModelIndex & index,
            int role ) const  [override]
```

StoreModel::data don't show the .gpg at the end of a file.

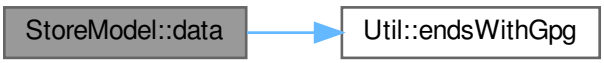
Parameters

index	
role	

Returns

Definition at line 97 of file storemodel.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

13.25.3.3 dropMimeData()

```
bool StoreModel::dropMimeData (
            const QMimeData * data,
            Qt::DropAction action,
            int row,
            int column,
            const QModelIndex & parent )  [override]
```

StoreModel::dropMimeData.

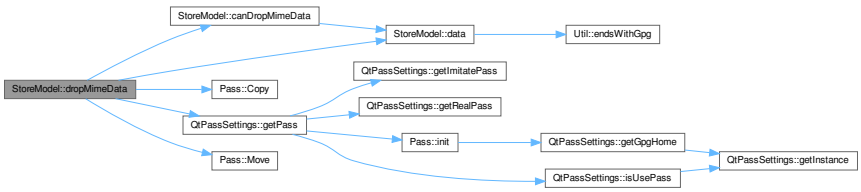
Parameters

data	
action	
row	
column	
parent	

Returns

Definition at line 238 of file storemodel.cpp.

Here is the call graph for this function:



13.25.3.4 filterAcceptsRow()

```
bool StoreModel::filterAcceptsRow (
            int sourceRow,
            const QModelIndex & sourceParent ) const  [override]
```

StoreModel::filterAcceptsRow should row be shown, wrapper for StoreModel::ShowThis method.

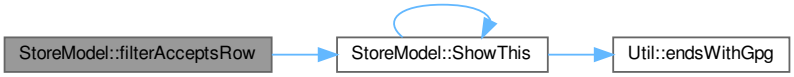
Parameters

sourceRow	
sourceParent	

Returns

Definition at line 43 of file storemodel.cpp.

Here is the call graph for this function:



13.25.3.5 flags()

```
Qt::ItemFlags StoreModel::flags (
            const QModelIndex & index ) const  [override]
```

StoreModel::flags.

Parameters

index	

Returns

Definition at line 134 of file storemodel.cpp.

13.25.3.6 lessThan()

```
bool StoreModel::lessThan (
            const QModelIndex & source_left,
            const QModelIndex & source_right ) const  [override]
```

StoreModel::lessThan.

Parameters

source_left	
source_right	

Returns

Definition at line 300 of file storemodel.cpp.

13.25.3.7 mimeData()

```
QMimeData * StoreModel::mimeData (
            const QModelIndexList & indexes ) const  [override]
```

StoreModel::mimeData.

Parameters

indexes	

Returns

Definition at line 158 of file storemodel.cpp.

Here is the call graph for this function:

Here is the caller graph for this function:

13.25.3.8 mimeTypes()

```
QStringList StoreModel::mimeTypes ( ) const  [override]
```

StoreModel::mimeTypes.

Returns

Definition at line 147 of file storemodel.cpp.

13.25.3.9 setModelAndStore()

```
void StoreModel::setModelAndStore (
            QFileSystemModel * sourceModel,
            QString passStore )
```

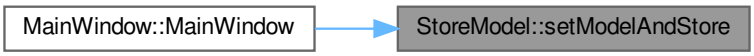
StoreModel::setModelAndStore update the source model and store.

Parameters

sourceModel	
passStore	

Definition at line 84 of file storemodel.cpp.

Here is the caller graph for this function:



13.25.3.10 ShowThis()

```
bool StoreModel::ShowThis (
            const QModelIndex index ) const
```

StoreModel::ShowThis should a row be shown, based on our search criteria.

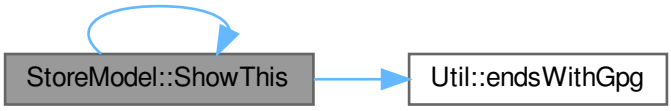
Parameters

index	

Returns

Definition at line 55 of file storemodel.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

13.25.3.11 supportedDragActions()

```
Qt::DropActions StoreModel::supportedDragActions ( ) const  [override]
```

StoreModel::supportedDragActions enable drag.

Returns

Definition at line 125 of file storemodel.cpp.

13.25.3.12 supportedDropActions()

```
Qt::DropActions StoreModel::supportedDropActions ( ) const  [override]
```

StoreModel::supportedDropActions enable drop.

Returns

Definition at line 117 of file storemodel.cpp.

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

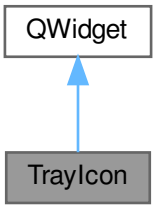
- src/storemodel.h
- src/storemodel.cpp

13.26 TrayIcon Class Reference

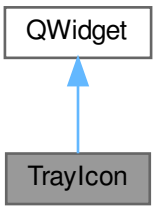
Handles the systemtray icon and menu.

```
#include <trayicon.h>
```

Inheritance diagram for TrayIcon:



Collaboration diagram for TrayIcon:



Public Slots

- void showHideParent ()

  TrayIcon::showHideParent toggle app visibility.
- void iconActivated (QSystemTrayIcon::ActivationReason reason)

  TrayIcon::iconActivated you clicked on the trayicon.

Public Member Functions

- TrayIcon (QMainWindow ∗parent)

  TrayIcon::TrayIcon use a (system) tray icon with a nice QtPass logo on it (currently) only Quits.
- void showMessage (const QString &title, const QString &msg, int time)

  TrayIcon::showMessage show a systray message for notification.
- void setVisible (bool visible)

  TrayIcon::setVisible show or hide the icon.
- bool getIsAllocated ()

  TrayIcon::getIsAllocated return if TrayIcon is allocated.

13.26.1 Detailed Description

Handles the systemtray icon and menu.

Definition at line 14 of file trayicon.h.

13.26.2 Constructor & Destructor Documentation

13.26.2.1 TrayIcon()

```
TrayIcon::TrayIcon (
            QMainWindow * parent )  [explicit]
```

TrayIcon::TrayIcon use a (system) tray icon with a nice QtPass logo on it (currently) only Quits.

Parameters

parent	

Definition at line 16 of file trayicon.cpp.

Here is the call graph for this function:

13.26.3 Member Function Documentation

13.26.3.1 getIsAllocated()

```
bool TrayIcon::getIsAllocated ( )
```

TrayIcon::getIsAllocated return if TrayIcon is allocated.

Definition at line 57 of file trayicon.cpp.

13.26.3.2 iconActivated

```
void TrayIcon::iconActivated (
            QSystemTrayIcon::ActivationReason reason )  [slot]
```

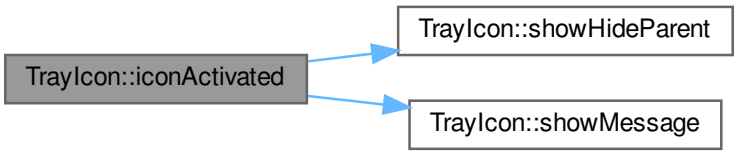
TrayIcon::iconActivated you clicked on the trayicon.

Parameters

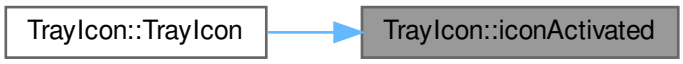
reason	

Definition at line 112 of file trayicon.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.26.3.3 setVisible()

```
void TrayIcon::setVisible (
            bool visible )
```

TrayIcon::setVisible show or hide the icon.

Parameters

visible	

Definition at line 47 of file trayicon.cpp.

13.26.3.4 showHideParent

```
void TrayIcon::showHideParent ( )  [slot]
```

TrayIcon::showHideParent toggle app visibility.

Definition at line 101 of file trayicon.cpp.

Here is the caller graph for this function:

13.26.3.5 showMessage()

```
void TrayIcon::showMessage (
            const QString & title,
            const QString & msg,
            int time )
```

TrayIcon::showMessage show a systray message for notification.

Parameters

title	
msg	
time	

Definition at line 132 of file trayicon.cpp.

Here is the caller graph for this function:

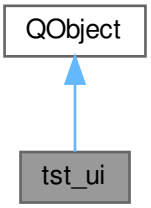
The documentation for this class was generated from the following files:

- src/trayicon.h
- src/trayicon.cpp

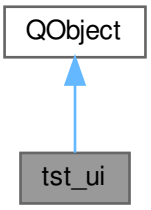
13.27 tst_ui Class Reference

The tst_ui class is our first unit test.

Inheritance diagram for tst_ui:



Collaboration diagram for tst_ui:



13.27.1 Detailed Description

The tst_ui class is our first unit test.

Definition at line 9 of file tst_ui.cpp.

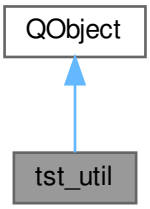
The documentation for this class was generated from the following file:

- tests/auto/ui/tst_ui.cpp

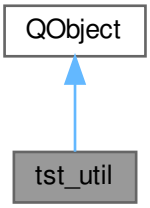
13.28 tst_util Class Reference

The tst_util class is our first unit test.

Inheritance diagram for tst_util:



Collaboration diagram for tst_util:



Public Slots

- void init ()

  tst_util::init unit test init method
- void cleanup ()

  tst_util::cleanup unit test cleanup method

Public Member Functions

- tst_util ()

  tst_util::tst_util basic constructor
- ~tst_util () override

  tst_util::~tst_util basic destructor

13.28.1 Detailed Description

The tst_util class is our first unit test.

Definition at line 10 of file tst_util.cpp.

13.28.2 Constructor & Destructor Documentation

13.28.2.1 tst_util()

```
tst_util::tst_util ( )  [default]
```

tst_util::tst_util basic constructor

13.28.2.2 ~tst_util()

```
tst_util::~tst_util ( )  [override], [default]
```

tst_util::~tst_util basic destructor

13.28.3 Member Function Documentation

13.28.3.1 cleanup

```
void tst_util::cleanup ( )  [slot]
```

tst_util::cleanup unit test cleanup method

Definition at line 50 of file tst_util.cpp.

13.28.3.2 init

```
void tst_util::init ( )  [slot]
```

tst_util::init unit test init method

Definition at line 45 of file tst_util.cpp.

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

- tests/auto/util/tst_util.cpp

13.29 UserInfo Struct Reference

Stores key info lines including validity, creation date and more.

`#include <userinfo.h>`

Public Member Functions

- UserInfo ()
- bool fullyValid ()

  UserInfo::fullyValid when validity is f or u. http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=doc/DETAILS.
- bool marginallyValid ()

  UserInfo::marginallyValid when validity is m. http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=doc/DETAILS.
- bool isValid ()

  UserInfo::isValid when fullyValid or marginallyValid.

Public Attributes

- QString name

  UserInfo::name full name.
- QString key_id

  UserInfo::key_id hexadecimal representation.
- char validity

  UserInfo::validity GnuPG representation of validity http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=doc/DETAILS.
- bool have_secret

  UserInfo::have_secret secret key is available (can decrypt with this key)
- bool enabled

  UserInfo::enabled.
- QDateTime expiry

  UserInfo::expiry date/time key expires.
- QDateTime created

  UserInfo::created date/time key was created.

13.29.1 Detailed Description

Stores key info lines including validity, creation date and more.

Definition at line 11 of file userinfo.h.

13.29.2 Constructor & Destructor Documentation

13.29.2.1 UserInfo()

```
UserInfo::UserInfo ( )  [inline]
```

Definition at line 12 of file userinfo.h.

13.29.3 Member Function Documentation

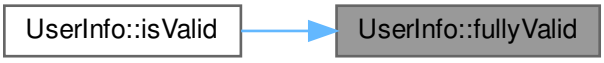
13.29.3.1 fullyValid()

```
bool UserInfo::fullyValid ( )  [inline]
```

UserInfo::fullyValid when validity is f or u. http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=doc/DETAILS.

Definition at line 18 of file userinfo.h.

Here is the caller graph for this function:



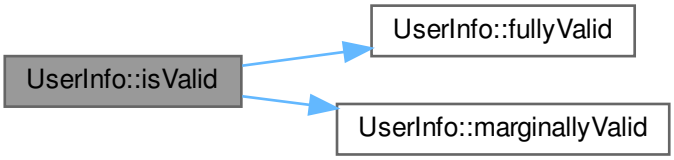
13.29.3.2 isValid()

```
bool UserInfo::isValid ( )  [inline]
```

UserInfo::isValid when fullyValid or marginallyValid.

Definition at line 27 of file userinfo.h.

Here is the call graph for this function:



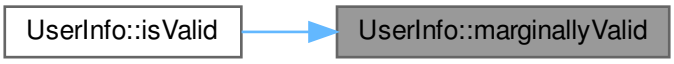
13.29.3.3 marginallyValid()

```
bool UserInfo::marginallyValid ( )  [inline]
```

UserInfo::marginallyValid when validity is m. http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=doc/DETAILS.

Definition at line 23 of file userinfo.h.

Here is the caller graph for this function:



13.29.4 Member Data Documentation

13.29.4.1 created

```
QDateTime UserInfo::created
```

UserInfo::created date/time key was created.

Definition at line 58 of file userinfo.h.

13.29.4.2 enabled

```
bool UserInfo::enabled
```

UserInfo::enabled.

Definition at line 50 of file userinfo.h.

13.29.4.3 expiry

```
QDateTime UserInfo::expiry
```

UserInfo::expiry date/time key expires.

Definition at line 54 of file userinfo.h.

13.29.4.4 have_secret

```
bool UserInfo::have_secret
```

UserInfo::have_secret secret key is available (can decrypt with this key)

Definition at line 46 of file userinfo.h.

13.29.4.5 key_id

```
QString UserInfo::key_id
```

UserInfo::key_id hexadecimal representation.

Definition at line 36 of file userinfo.h.

13.29.4.6 name

```
QString UserInfo::name
```

UserInfo::name full name.

Definition at line 32 of file userinfo.h.

13.29.4.7 validity

```
char UserInfo::validity
```

UserInfo::validity GnuPG representation of validity http://git.gnupg.org/cgi-bin/gitweb.cgi?p=gnupg.git;a=blob_plain;f=doc/DETAILS.

Definition at line 41 of file userinfo.h.

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

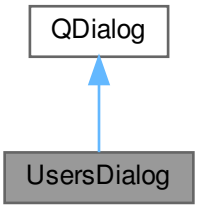
- src/userinfo.h

13.30 UsersDialog Class Reference

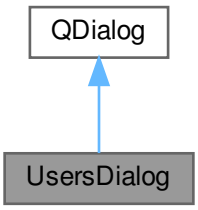
Handles listing and editing of GPG users.

`#include <usersdialog.h>`

Inheritance diagram for UsersDialog:



Collaboration diagram for UsersDialog:



Public Slots

- void accept ()
  UsersDialog::accept.

Public Member Functions

- UsersDialog (QString dir, QWidget *parent=nullptr)
  UsersDialog::UsersDialog basic constructor.
- ~UsersDialog ()
  UsersDialog::~UsersDialog basic destructor.

Protected Member Functions

- void closeEvent (QCloseEvent *event)

  UsersDialog::closeEvent might have to store size and location if that is wanted.
- void keyPressEvent (QKeyEvent *event)

  UsersDialog::keyPressEvent clear the lineEdit when escape is pressed. No action for Enter currently.

13.30.1 Detailed Description

Handles listing and editing of GPG users.

Selection of whom to encrypt to.

Definition at line 23 of file usersdialog.h.

13.30.2 Constructor & Destructor Documentation

13.30.2.1 UsersDialog()

```
UsersDialog::UsersDialog (
            QString dir,
            QWidget * parent = nullptr )  [explicit]
```

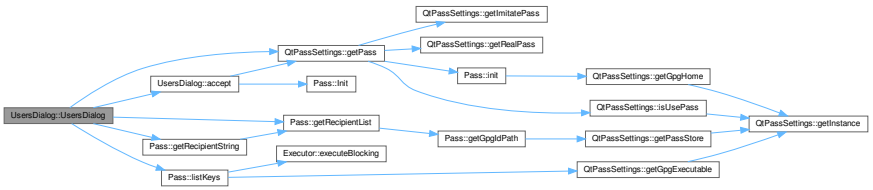
UsersDialog::UsersDialog basic constructor.

Parameters

parent	

Definition at line 18 of file usersdialog.cpp.

Here is the call graph for this function:



13.30.2.2 ~UsersDialog()

```
UsersDialog::~UsersDialog ( )
```

UsersDialog::~UsersDialog basic destructor.

Definition at line 80 of file usersdialog.cpp.

13.30.3 Member Function Documentation

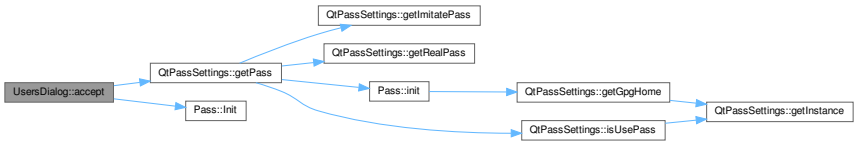
13.30.3.1 accept

```
void UsersDialog::accept ( )  [slot]
```

UsersDialog::accept.

Definition at line 87 of file usersdialog.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

13.30.3.2 closeEvent()

```
void UsersDialog::closeEvent (
            QCloseEvent * event )  [protected]
```

UsersDialog::closeEvent might have to store size and location if that is wanted.

Parameters

event	

Definition at line 98 of file usersdialog.cpp.

13.30.3.3 keyPressEvent()

```
void UsersDialog::keyPressEvent (
            QKeyEvent * event )  [protected]
```

UsersDialog::keyPressEvent clear the lineEdit when escape is pressed. No action for Enter currently.

Parameters

event	

Definition at line 108 of file usersdialog.cpp.

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

- src/usersdialog.h
- src/usersdialog.cpp

13.31 Util Class Reference

Some static utilities to be used elsewhere.

```
#include <util.h>
```

Static Public Member Functions

- static QString findBinaryInPath (QString binary)

  Util::findBinaryInPath search for executables.
- static QString findPasswordStore ()

  Util::findPasswordStore look for common .password-store folder location.
- static QString normalizeFolderPath (QString path)

  Util::normalizeFolderPath let's always end folders with a QDir::separator()
- static bool checkConfig ()

  Util::checkConfig do we have prequisite settings?
- static QString getDir (const QModelIndex &index, bool forPass, const QFileSystemModel &model, const StoreModel &storeModel)

  Util::getDir get selectd folder path.
- static void copyDir (const QString &src, const QString &dest)

  Util::copyDir.
- static const QRegularExpression & endsWithGpg ()
- static const QRegularExpression & protocolRegex ()

13.31.1 Detailed Description

Some static utilities to be used elsewhere.

Definition at line 16 of file util.h.

13.31.2 Member Function Documentation

13.31.2.1 checkConfig()

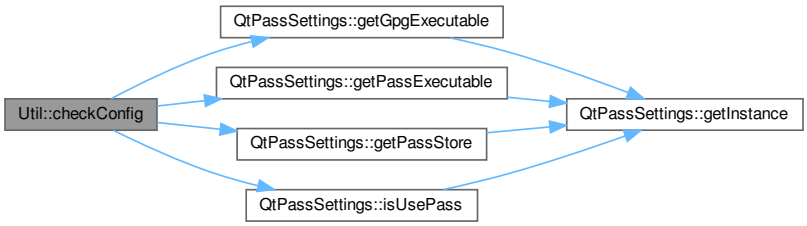
```
bool Util::checkConfig ( )  [static]
```

Util::checkConfig do we have prequisite settings?

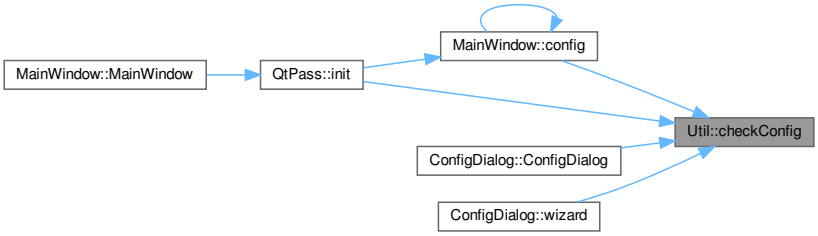
Returns

Definition at line 142 of file util.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.31.2.2 copyDir()

```
void Util::copyDir (
            const QString & src,
            const QString & dest )  [static]
```

Util::copyDir.

Parameters

src	
dest	

Definition at line 182 of file util.cpp.

Here is the call graph for this function:

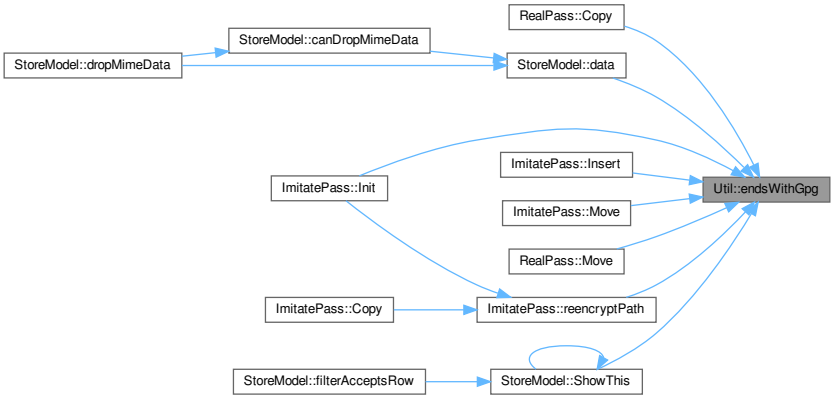
Here is the caller graph for this function:

13.31.2.3 endsWithGpg()

```
const QRegularExpression & Util::endsWithGpg ( )  [static]
```

Definition at line 198 of file util.cpp.

Here is the caller graph for this function:



13.31.2.4 findBinaryInPath()

```
QString Util::findBinaryInPath (
            QString binary )  [static]
```

Util::findBinaryInPath search for executables.

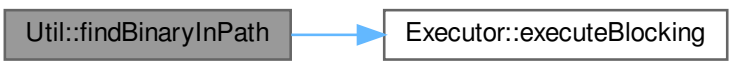
Parameters

binary	

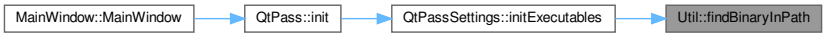
Returns

Definition at line 90 of file util.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.31.2.5 findPasswordStore()

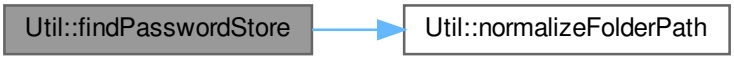
```
QString Util::findPasswordStore ( )  [static]
```

Util::findPasswordStore look for common .password-store folder location.

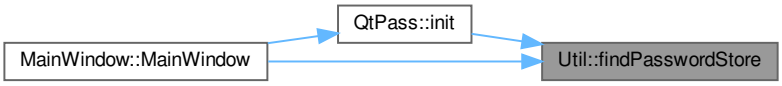
Returns

Definition at line 56 of file util.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.31.2.6 getDir()

```
QString Util::getDir (
            const QModelIndex & index,
            bool forPass,
            const QFileSystemModel & model,
            const StoreModel & storeModel )  [static]
```

Util::getDir get selectd folder path.

Parameters

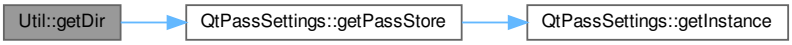
index	
forPass	short or full path
model	the filesystem model to operate on
storeModel	our storemodel to operate on

Returns

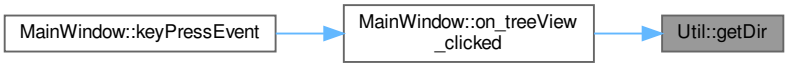
path

Definition at line 160 of file util.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



13.31.2.7 normalizeFolderPath()

```
QString Util::normalizeFolderPath (
            QString path )  [static]
```

Util::normalizeFolderPath let's always end folders with a QDir::separator()

Parameters

path	

Returns

Definition at line 79 of file util.cpp.

Here is the caller graph for this function:

13.31.2.8 protocolRegex()

```
const QRegularExpression & Util::protocolRegex ( )  [static]
```

Definition at line 203 of file util.cpp.

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

- src/util.h
- src/util.cpp

Chapter 14

File Documentation

14.1 CHANGELOG.md File Reference

14.2 CODE_OF_CONDUCT.md File Reference

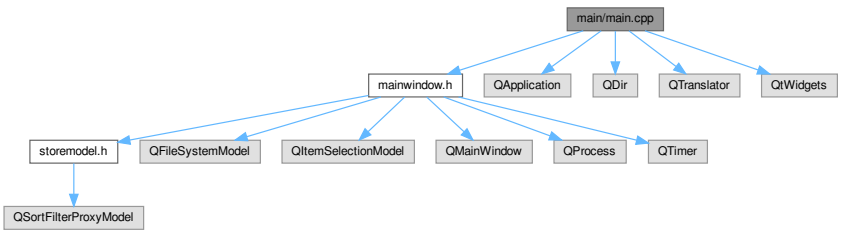
14.3 CONTRIBUTING.md File Reference

14.4 FAQ.md File Reference

14.5 main/main.cpp File Reference

```
#include "mainwindow.h"
#include <QApplication>
#include <QDir>
#include <QTranslator>
#include <QtWidgets>
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

  main

14.5.1 Function Documentation

14.5.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

main

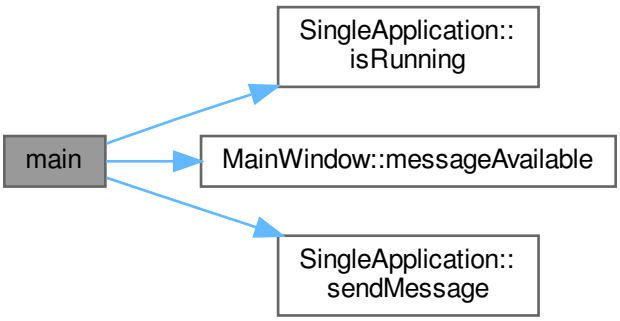
Parameters

argc	
argv	

Returns

Definition at line 43 of file main.cpp.

Here is the call graph for this function:



14.6 main.cpp

Go to the documentation of this file.

```
00001 #include "mainwindow.h"
00002 #if SINGLE_APP
00003 #include "singleapplication.h"
00004 #endif
00005
00006 #include <QApplication>
00007 #include <QDir>
```

```
00008 #include <QTranslator>
00009 #include <QtWidgets>
00010
00043 int main(int argc, char *argv[]) {
00044 #if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
00045   QApplication::setAttribute(Qt::AA_EnableHighDpiScaling);
00046   QApplication::setAttribute(Qt::AA_UseHighDpiPixmaps);
00047 #endif
00048   QString text = "";
00049   for (int i = 1; i < argc; ++i) {
00050     if (i > 1)
00051       text += " ";
00052     text += argv[i];
00053   }
00054
00055   if ((text.indexOf("-psn_") == 0) || (text.indexOf("-session") == 0)) {
00056     text.clear();
00057   }
00058
00059 #if SINGLE_APP
00060   QString name = qgetenv("USER");
00061   if (name.isEmpty())
00062     name = qgetenv("USERNAME");
00063   SingleApplication app(argc, argv, name + "QtPass");
00064   if (app.isRunning()) {
00065     if (text.length() > 0)
00066       app.sendMessage(text);
00067     return 0;
00068   }
00069 #else
00070   QApplication app(argc, argv);
00071 #endif
00072
00073   Q_INIT_RESOURCE(resources);
00074   Q_INIT_RESOURCE(qmake_qmake_qm_files); // qmake names the file
00075
00076   QCoreApplication::setOrganizationName("IJHack");
00077   QCoreApplication::setOrganizationDomain("ijhack.org");
00078   QCoreApplication::setApplicationName("QtPass");
00079   QCoreApplication::setApplicationVersion(VERSION);
00080
00081   // Setup and load translator for localization
00082   QTranslator translator;
00083   QString locale = QLocale::system().name();
00084   // locale = "nl_NL";
00085   // locale = "he_IL";
00086   // locale = "ar_MA";
00087   if (translator.load(
00088           QString(":localization/localization_%1.qm").arg(locale))) {
00089     SingleApplication::installTranslator(&translator);
00090     SingleApplication::setLayoutDirection(
00091         QObject::tr("LTR") == "RTL" ? Qt::RightToLeft : Qt::LeftToRight);
00092   }
00093
00094   MainWindow w(text);
00095
00096   SingleApplication::setActiveWindow(&w);
00097   SingleApplication::setWindowIcon(QIcon(":artwork/icon.png"));
00098
00099 #if SINGLE_APP
00100   QObject::connect(&app, &SingleApplication::messageAvailable, &w,
00101                    &MainWindow::messageAvailable);
00102 #endif
00103
00104   QGuiApplication::setDesktopFileName("qtpass.desktop");
00105
00106   // Center the MainWindow on the screen the mouse pointer is currently on
00107 #if QT_VERSION < QT_VERSION_CHECK(5, 12, 0)
00108   static int cursorScreen =
00109       app.desktop()->screenNumber(app.desktop()->cursor().pos());
00110   QPoint cursorScreenCenter =
00111       app.desktop()->screenGeometry(cursorScreen).center();
00112 #else
00113   QScreen *screen = QGuiApplication::screenAt(QCursor::pos());
00114   QPoint cursorScreenCenter = screen->geometry().center();
00115 #endif
00116   QRect windowFrameGeo = w.frameGeometry();
00117   windowFrameGeo.moveCenter(cursorScreenCenter);
00118   w.move(windowFrameGeo.topLeft());
00119
00120   w.show();
00121
00122   return SingleApplication::exec();
00123 }
```

14.7 README.md File Reference

14.8 src/configdialog.cpp File Reference

```
#include "configdialog.h"
#include "keygendialog.h"
#include "mainwindow.h"
#include "qtpasssettings.h"
#include "ui_configdialog.h"
#include "util.h"
#include <QClipboard>
#include <QDir>
#include <QFileDialog>
#include <QMessageBox>
#include <QPushButton>
#include <QSystemTrayIcon>
#include <QTableWidgetItem>
```

Include dependency graph for configdialog.cpp:

14.9 configdialog.cpp

Go to the documentation of this file.

```
00001 #include "configdialog.h"
00002 #include "keygendialog.h"
00003 #include "mainwindow.h"
00004 #include "qtpasssettings.h"
00005 #include "ui_configdialog.h"
00006 #include "util.h"
00007 #include <QClipboard>
00008 #include <QDir>
00009 #include <QFileDialog>
00010 #include <QMessageBox>
00011 #include <QPushButton>
00012 #include <QSystemTrayIcon>
00013 #include <QTableWidgetItem>
00014 #ifdef Q_OS_WIN
00015 #include <windows.h>
00016 #endif
00017
00018 #ifdef QT_DEBUG
00019 #include "debughelper.h"
00020 #endif
00021
00026 ConfigDialog::ConfigDialog(MainWindow *parent)
00027     : QDialog(parent), ui(new Ui::ConfigDialog) {
00028   mainWindow = parent;
00029   ui->setupUi(this);
00030
00031   ui->passPath->setText(QtPassSettings::getPassExecutable());
00032   setGitPath(QtPassSettings::getGitExecutable());
00033   ui->gpgPath->setText(QtPassSettings::getGpgExecutable());
00034   ui->storePath->setText(QtPassSettings::getPassStore());
00035
00036   ui->spinBoxAutoclearSeconds->setValue(QtPassSettings::getAutoclearSeconds());
00037   ui->spinBoxAutoclearPanelSeconds->setValue(
00038       QtPassSettings::getAutoclearPanelSeconds());
00039   ui->checkBoxHidePassword->setChecked(QtPassSettings::isHidePassword());
00040   ui->checkBoxHideContent->setChecked(QtPassSettings::isHideContent());
```

```
00041   ui->checkBoxUseMonospace->setChecked(QtPassSettings::isUseMonospace());
00042   ui->checkBoxDisplayAsIs->setChecked(QtPassSettings::isDisplayAsIs());
00043   ui->checkBoxNoLineWrapping->setChecked(QtPassSettings::isNoLineWrapping());
00044   ui->checkBoxAddGPGId->setChecked(QtPassSettings::isAddGPGId(true));
00045
00046   if (QSystemTrayIcon::isSystemTrayAvailable()) {
00047     ui->checkBoxHideOnClose->setChecked(QtPassSettings::isHideOnClose());
00048     ui->checkBoxStartMinimized->setChecked(QtPassSettings::isStartMinimized());
00049   } else {
00050     ui->checkBoxUseTrayIcon->setEnabled(false);
00051     ui->checkBoxUseTrayIcon->setToolTip(tr("System tray is not available"));
00052     ui->checkBoxHideOnClose->setEnabled(false);
00053     ui->checkBoxStartMinimized->setEnabled(false);
00054   }
00055
00056   ui->checkBoxAvoidCapitals->setChecked(QtPassSettings::isAvoidCapitals());
00057   ui->checkBoxAvoidNumbers->setChecked(QtPassSettings::isAvoidNumbers());
00058   ui->checkBoxLessRandom->setChecked(QtPassSettings::isLessRandom());
00059   ui->checkBoxUseSymbols->setChecked(QtPassSettings::isUseSymbols());
00060   ui->plainTextEditTemplate->setPlainText(QtPassSettings::getPassTemplate());
00061   ui->checkBoxTemplateAllFields->setChecked(
00062       QtPassSettings::isTemplateAllFields());
00063   ui->checkBoxAutoPull->setChecked(QtPassSettings::isAutoPull());
00064   ui->checkBoxAutoPush->setChecked(QtPassSettings::isAutoPush());
00065   ui->checkBoxAlwaysOnTop->setChecked(QtPassSettings::isAlwaysOnTop());
00066
00067 #if defined(Q_OS_WIN)
00068   ui->checkBoxUseOtp->hide();
00069   ui->checkBoxUseQrencode->hide();
00070   ui->label_10->hide();
00071 #endif
00072
00073   if (!isPassOtpAvailable()) {
00074     ui->checkBoxUseOtp->setEnabled(false);
00075     ui->checkBoxUseOtp->setToolTip(
00076         tr("Pass OTP extension needs to be installed"));
00077   }
00078
00079   if (!isQrencodeAvailable()) {
00080     ui->checkBoxUseQrencode->setEnabled(false);
00081     ui->checkBoxUseQrencode->setToolTip(tr("qrencode needs to be installed"));
00082   }
00083
00084   setProfiles(QtPassSettings::getProfiles(), QtPassSettings::getProfile());
00085   setPwgenPath(QtPassSettings::getPwgenExecutable());
00086   setPasswordConfiguration(QtPassSettings::getPasswordConfiguration());
00087
00088   usePass(QtPassSettings::isUsePass());
00089   useAutoclear(QtPassSettings::isUseAutoclear());
00090   useAutoclearPanel(QtPassSettings::isUseAutoclearPanel());
00091   useTrayIcon(QtPassSettings::isUseTrayIcon());
00092   useGit(QtPassSettings::isUseGit());
00093
00094   useOtp(QtPassSettings::isUseOtp());
00095   useQrencode(QtPassSettings::isUseQrencode());
00096
00097   usePwgen(QtPassSettings::isUsePwgen());
00098   useTemplate(QtPassSettings::isUseTemplate());
00099
00100   ui->profileTable->verticalHeader()->hide();
00101   ui->profileTable->horizontalHeader()->setSectionResizeMode(
00102       1, QHeaderView::Stretch);
00103   ui->label->setText(ui->label->text() + VERSION);
00104   ui->comboBoxClipboard->clear();
00105
00106   ui->comboBoxClipboard->addItem(tr("No Clipboard"));
00107   ui->comboBoxClipboard->addItem(tr("Always copy to clipboard"));
00108   ui->comboBoxClipboard->addItem(tr("On-demand copy to clipboard"));
00109
00110   int currentIndex = QtPassSettings::getClipBoardTypeRaw();
00111   ui->comboBoxClipboard->setCurrentIndex(currentIndex);
00112   on_comboBoxClipboard_activated(currentIndex);
00113
00114   QClipboard *clip = QApplication::clipboard();
00115   if (!clip->supportsSelection()) {
00116     useSelection(false);
00117     ui->checkBoxSelection->setVisible(false);
00118   } else {
00119     useSelection(QtPassSettings::isUseSelection());
00120   }
00121
00122   if (Util::checkConfig()) {
00123     // Show Programs tab, which is likely
00124     // what the user needs to fix now.
00125     ui->tabWidget->setCurrentIndex(1);
00126   }
00127
```

```
00128   connect(ui->profileTable, &QTableWidget::itemChanged, this,
00129           &ConfigDialog::onProfileTableItemChanged);
00130   connect(this, &ConfigDialog::accepted, this, &ConfigDialog::on_accepted);
00131 }
00132
00137 ConfigDialog::~ConfigDialog() {
00138   QtPassSettings::setGitExecutable(ui->gitPath->text());
00139   QtPassSettings::setGpgExecutable(ui->gpgPath->text());
00140   QtPassSettings::setPassExecutable(ui->passPath->text());
00141 }
00142
00148 void ConfigDialog::setGitPath(QString path) {
00149   ui->gitPath->setText(path);
00150   ui->checkBoxUseGit->setEnabled(!path.isEmpty());
00151   if (path.isEmpty()) {
00152     useGit(false);
00153   }
00154 }
00155
00161 void ConfigDialog::usePass(bool usePass) {
00162   ui->radioButtonNative->setChecked(!usePass);
00163   ui->radioButtonPass->setChecked(usePass);
00164   setGroupBoxState();
00165 }
00166
00167 void ConfigDialog::validate(QTableWidgetItem *item) {
00168   bool status = true;
00169
00170   if (item == nullptr) {
00171     for (int i = 0; i < ui->profileTable->rowCount(); i++) {
00172       for (int j = 0; j < ui->profileTable->columnCount(); j++) {
00173         QTableWidgetItem *_item = ui->profileTable->item(i, j);
00174
00175         if (_item->text().isEmpty() && j != 2) {
00176           _item->setBackground(Qt::red);
00177           status = false;
00178           break;
00179         }
00180       }
00181
00182       if (!status)
00183         break;
00184     }
00185   } else {
00186     if (item->text().isEmpty() && item->column() != 2) {
00187       item->setBackground(Qt::red);
00188       status = false;
00189     }
00190   }
00191
00192   ui->buttonBox->button(QDialogButtonBox::Ok)->setEnabled(status);
00193 }
00194
00195 void ConfigDialog::on_accepted() {
00196   QtPassSettings::setPassExecutable(ui->passPath->text());
00197   QtPassSettings::setGitExecutable(ui->gitPath->text());
00198   QtPassSettings::setGpgExecutable(ui->gpgPath->text());
00199   QtPassSettings::setPassStore(
00200       Util::normalizeFolderPath(ui->storePath->text()));
00201   QtPassSettings::setUsePass(ui->radioButtonPass->isChecked());
00202   QtPassSettings::setClipBoardType(ui->comboBoxClipboard->currentIndex());
00203   QtPassSettings::setUseSelection(ui->checkBoxSelection->isChecked());
00204   QtPassSettings::setUseAutoclear(ui->checkBoxAutoclear->isChecked());
00205   QtPassSettings::setAutoclearSeconds(ui->spinBoxAutoclearSeconds->value());
00206   QtPassSettings::setUseAutoclearPanel(ui->checkBoxAutoclearPanel->isChecked());
00207   QtPassSettings::setAutoclearPanelSeconds(
00208       ui->spinBoxAutoclearPanelSeconds->value());
00209   QtPassSettings::setHidePassword(ui->checkBoxHidePassword->isChecked());
00210   QtPassSettings::setHideContent(ui->checkBoxHideContent->isChecked());
00211   QtPassSettings::setUseMonospace(ui->checkBoxUseMonospace->isChecked());
00212   QtPassSettings::setDisplayAsIs(ui->checkBoxDisplayAsIs->isChecked());
00213   QtPassSettings::setNoLineWrapping(ui->checkBoxNoLineWrapping->isChecked());
00214   QtPassSettings::setAddGPGId(ui->checkBoxAddGPGId->isChecked());
00215   QtPassSettings::setUseTrayIcon(ui->checkBoxUseTrayIcon->isEnabled() &&
00216                                  ui->checkBoxUseTrayIcon->isChecked());
00217   QtPassSettings::setHideOnClose(ui->checkBoxHideOnClose->isEnabled() &&
00218                                  ui->checkBoxHideOnClose->isChecked());
00219   QtPassSettings::setStartMinimized(ui->checkBoxStartMinimized->isEnabled() &&
00220                                     ui->checkBoxStartMinimized->isChecked());
00221   QtPassSettings::setProfiles(getProfiles());
00222   QtPassSettings::setUseGit(ui->checkBoxUseGit->isChecked());
00223   QtPassSettings::setUseOtp(ui->checkBoxUseOtp->isChecked());
00224   QtPassSettings::setUseQrencode(ui->checkBoxUseQrencode->isChecked());
00225   QtPassSettings::setPwgenExecutable(ui->pwgenPath->text());
00226   QtPassSettings::setUsePwgen(ui->checkBoxUsePwgen->isChecked());
00227   QtPassSettings::setAvoidCapitals(ui->checkBoxAvoidCapitals->isChecked());
00228   QtPassSettings::setAvoidNumbers(ui->checkBoxAvoidNumbers->isChecked());
```

```
00229   QtPassSettings::setLessRandom(ui->checkBoxLessRandom->isChecked());
00230   QtPassSettings::setUseSymbols(ui->checkBoxUseSymbols->isChecked());
00231   QtPassSettings::setPasswordConfiguration(getPasswordConfiguration());
00232   QtPassSettings::setUseTemplate(ui->checkBoxUseTemplate->isChecked());
00233   QtPassSettings::setPassTemplate(ui->plainTextEditTemplate->toPlainText());
00234   QtPassSettings::setTemplateAllFields(
00235       ui->checkBoxTemplateAllFields->isChecked());
00236   QtPassSettings::setAutoPush(ui->checkBoxAutoPush->isChecked());
00237   QtPassSettings::setAutoPull(ui->checkBoxAutoPull->isChecked());
00238   QtPassSettings::setAlwaysOnTop(ui->checkBoxAlwaysOnTop->isChecked());
00239
00240   QtPassSettings::setVersion(VERSION);
00241 }
00242
00243 void ConfigDialog::on_autodetectButton_clicked() {
00244   QString pass = Util::findBinaryInPath("pass");
00245   if (!pass.isEmpty())
00246     ui->passPath->setText(pass);
00247   usePass(!pass.isEmpty());
00248   QString gpg = Util::findBinaryInPath("gpg2");
00249   if (gpg.isEmpty())
00250     gpg = Util::findBinaryInPath("gpg");
00251   if (!gpg.isEmpty())
00252     ui->gpgPath->setText(gpg);
00253   QString git = Util::findBinaryInPath("git");
00254   if (!git.isEmpty())
00255     ui->gitPath->setText(git);
00256   QString pwgen = Util::findBinaryInPath("pwgen");
00257   if (!pwgen.isEmpty())
00258     ui->pwgenPath->setText(pwgen);
00259 }
00260
00265 void ConfigDialog::on_radioButtonNative_clicked() { setGroupBoxState(); }
00266
00271 void ConfigDialog::on_radioButtonPass_clicked() { setGroupBoxState(); }
00272
00277 QStringList ConfigDialog::getSecretKeys() {
00278   QList<UserInfo> keys = QtPassSettings::getPass()->listKeys("", true);
00279   QStringList names;
00280
00281   if (keys.empty())
00282     return names;
00283
00284   foreach (const UserInfo &sec, keys)
00285     names << sec.name;
00286
00287   return names;
00288 }
00289
00293 void ConfigDialog::setGroupBoxState() {
00294   bool state = ui->radioButtonPass->isChecked();
00295   ui->groupBoxNative->setEnabled(!state);
00296   ui->groupBoxPass->setEnabled(state);
00297 }
00298
00303 QString ConfigDialog::selectExecutable() {
00304   QFileDialog dialog(this);
00305   dialog.setFileMode(QFileDialog::ExistingFile);
00306   dialog.setOption(QFileDialog::ReadOnly);
00307   if (dialog.exec())
00308     return dialog.selectedFiles().constFirst();
00309
00310   return QString();
00311 }
00312
00317 QString ConfigDialog::selectFolder() {
00318   QFileDialog dialog(this);
00319   dialog.setFileMode(QFileDialog::Directory);
00320   dialog.setFilter(QDir::NoFilter);
00321   dialog.setOption(QFileDialog::ShowDirsOnly);
00322   if (dialog.exec())
00323     return dialog.selectedFiles().constFirst();
00324
00325   return QString();
00326 }
00327
00332 void ConfigDialog::on_toolButtonGit_clicked() {
00333   QString git = selectExecutable();
00334   bool state = !git.isEmpty();
00335   if (state) {
00336     ui->gitPath->setText(git);
00337   } else {
00338     useGit(false);
00339   }
00340
00341   ui->checkBoxUseGit->setEnabled(state);
00342 }
```

```
00343
00347 void ConfigDialog::on_toolButtonGpg_clicked() {
00348   QString gpg = selectExecutable();
00349   if (!gpg.isEmpty())
00350     ui->gpgPath->setText(gpg);
00351 }
00352
00356 void ConfigDialog::on_toolButtonPass_clicked() {
00357   QString pass = selectExecutable();
00358   if (!pass.isEmpty())
00359     ui->passPath->setText(pass);
00360 }
00361
00366 void ConfigDialog::on_toolButtonStore_clicked() {
00367   QString store = selectFolder();
00368   if (!store.isEmpty()) // TODO(annejan) call check
00369     ui->storePath->setText(store);
00370 }
00371
00376 void ConfigDialog::on_comboBoxClipboard_activated(int index) {
00377   bool state = index > 0;
00378
00379   ui->checkBoxSelection->setEnabled(state);
00380   ui->checkBoxAutoclear->setEnabled(state);
00381   ui->checkBoxHidePassword->setEnabled(state);
00382   ui->checkBoxHideContent->setEnabled(state);
00383   if (state) {
00384     ui->spinBoxAutoclearSeconds->setEnabled(ui->checkBoxAutoclear->isChecked());
00385     ui->labelSeconds->setEnabled(ui->checkBoxAutoclear->isChecked());
00386   } else {
00387     ui->spinBoxAutoclearSeconds->setEnabled(false);
00388     ui->labelSeconds->setEnabled(false);
00389   }
00390 }
00391
00396 void ConfigDialog::on_checkBoxAutoclearPanel_clicked() {
00397   bool state = ui->checkBoxAutoclearPanel->isChecked();
00398   ui->spinBoxAutoclearPanelSeconds->setEnabled(state);
00399   ui->labelPanelSeconds->setEnabled(state);
00400 }
00401
00407 void ConfigDialog::useSelection(bool useSelection) {
00408   ui->checkBoxSelection->setChecked(useSelection);
00409   on_checkBoxSelection_clicked();
00410 }
00411
00417 void ConfigDialog::useAutoclear(bool useAutoclear) {
00418   ui->checkBoxAutoclear->setChecked(useAutoclear);
00419   on_checkBoxAutoclear_clicked();
00420 }
00421
00427 void ConfigDialog::useAutoclearPanel(bool useAutoclearPanel) {
00428   ui->checkBoxAutoclearPanel->setChecked(useAutoclearPanel);
00429   on_checkBoxAutoclearPanel_clicked();
00430 }
00431
00436 void ConfigDialog::on_checkBoxSelection_clicked() {
00437   on_comboBoxClipboard_activated(ui->comboBoxClipboard->currentIndex());
00438 }
00439
00444 void ConfigDialog::on_checkBoxAutoclear_clicked() {
00445   on_comboBoxClipboard_activated(ui->comboBoxClipboard->currentIndex());
00446 }
00447
00455 void ConfigDialog::genKey(QString batch, QDialog *dialog) {
00456   mainWindow->generateKeyPair(batch, dialog);
00457 }
00458
00465 void ConfigDialog::setProfiles(QHash<QString, QHash<QString, QString» profiles,
00466                                QString currentProfile) {
00467   // dbg()« profiles;
00468   if (profiles.contains("")) {
00469     profiles.remove("");
00470     // remove weird "" key value pairs
00471   }
00472
00473   ui->profileTable->setRowCount(profiles.count());
00474   QHashIterator<QString, QHash<QString, QString» i(profiles);
00475   int n = 0;
00476   while (i.hasNext()) {
00477     i.next();
00478     if (!i.value().isEmpty() && !i.key().isEmpty()) {
00479       ui->profileTable->setItem(n, 0, new QTableWidgetItem(i.key()));
00480       ui->profileTable->setItem(n, 1,
00481                                 new QTableWidgetItem(i.value().value("path")));
00482       ui->profileTable->setItem(
00483           n, 2, new QTableWidgetItem(i.value().value("signingKey")));
```

```
00484       // dbg()« "naam:" + i.key();
00485       if (i.key() == currentProfile)
00486         ui->profileTable->selectRow(n);
00487     }
00488     ++n;
00489   }
00490 }
00491
00496 QHash<QString, QHash<QString, QString» ConfigDialog::getProfiles() {
00497   QHash<QString, QHash<QString, QString» profiles;
00498   // Check?
00499   for (int i = 0; i < ui->profileTable->rowCount(); ++i) {
00500     QHash<QString, QString> profile;
00501     QTableWidgetItem *pathItem = ui->profileTable->item(i, 1);
00502     if (nullptr != pathItem) {
00503       QTableWidgetItem *item = ui->profileTable->item(i, 0);
00504       if (item == nullptr) {
00505         continue;
00506       }
00507       profile["path"] = pathItem->text();
00508       QTableWidgetItem *signingKeyItem = ui->profileTable->item(i, 2);
00509       if (nullptr != signingKeyItem) {
00510         profile["signingKey"] = signingKeyItem->text();
00511       }
00512       profiles.insert(item->text(), profile);
00513     }
00514   }
00515   return profiles;
00516 }
00517
00521 void ConfigDialog::on_addButton_clicked() {
00522   int n = ui->profileTable->rowCount();
00523   ui->profileTable->insertRow(n);
00524   ui->profileTable->setItem(n, 0, new QTableWidgetItem());
00525   ui->profileTable->setItem(n, 1, new QTableWidgetItem(ui->storePath->text()));
00526   ui->profileTable->setItem(n, 2, new QTableWidgetItem());
00527   ui->profileTable->selectRow(n);
00528   ui->deleteButton->setEnabled(true);
00529
00530   validate();
00531 }
00532
00536 void ConfigDialog::on_deleteButton_clicked() {
00537   QSet<int> selectedRows; // we use a set to prevent doubles
00538   QList<QTableWidgetItem *> itemList = ui->profileTable->selectedItems();
00539   if (itemList.count() == 0) {
00540     QMessageBox::warning(this, tr("No profile selected"),
00541                          tr("No profile selected to delete"));
00542     return;
00543   }
00544   QTableWidgetItem *item;
00545   foreach (item, itemList)
00546     selectedRows.insert(item->row());
00547   // get a list, and sort it big to small
00548   QList<int> rows = selectedRows.values();
00549   std::sort(rows.begin(), rows.end());
00550   // now actually do the removing:
00551   foreach (int row, rows)
00552     ui->profileTable->removeRow(row);
00553   if (ui->profileTable->rowCount() < 1)
00554     ui->deleteButton->setEnabled(false);
00555
00556   validate();
00557 }
00558
00565 void ConfigDialog::criticalMessage(const QString &title, const QString &text) {
00566   QMessageBox::critical(this, title, text, QMessageBox::Ok, QMessageBox::Ok);
00567 }
00568
00569 bool ConfigDialog::isQrencodeAvailable() {
00570 #ifdef Q_OS_WIN
00571   return false;
00572 #else
00573   QProcess which;
00574   which.start("which", QStringList() « "qrencode");
00575   which.waitForFinished();
00576   QtPassSettings::setQrencodeExecutable(
00577       which.readAllStandardOutput().trimmed());
00578   return which.exitCode() == 0;
00579 #endif
00580 }
00581
00582 bool ConfigDialog::isPassOtpAvailable() {
00583 #ifdef Q_OS_WIN
00584   return false;
00585 #else
00586   return true;
```

```
00587 #endif
00588 }
00589
00594 void ConfigDialog::wizard() {
00595   Util::checkConfig();
00596   on_autodetectButton_clicked();
00597   bool clean = false;
00598
00599   QString gpg = ui->gpgPath->text();
00600   if (!gpg.startsWith("wsl ") && !QFile(gpg).exists()) {
00601     criticalMessage(
00602         tr("GnuPG not found"),
00603 #ifdef Q_OS_WIN
00604 #ifdef WINSTORE
00605         tr("Please install GnuPG on your system.<br>Install "
00606            "<strong>Ubuntu</strong> from the Microsoft Store to get it.<br>"
00607            "If you already did so, make sure you started it once and<br>"
00608            "click \"Autodetect\" in the next dialog.")
00609 #else
00610         tr("Please install GnuPG on your system.<br>Install "
00611            "<strong>Ubuntu</strong> from the Microsoft Store<br>or <a "
00612            "href=\"https://www.gnupg.org/download/#sec-1-2\">download</a> it "
00613            "from GnuPG.org")
00614 #endif
00615 #else
00616         tr("Please install GnuPG on your system.<br>Install "
00617            "<strong>gpg</strong> using your favorite package manager<br>or "
00618            "<a "
00619            "href=\"https://www.gnupg.org/download/#sec-1-2\">download</a> it "
00620            "from GnuPG.org")
00621 #endif
00622     );
00623     clean = true;
00624   }
00625
00626   QStringList names = getSecretKeys();
00627
00628 #ifdef QT_DEBUG
00629   dbg() « names;
00630 #endif
00631
00632   if ((gpg.startsWith("wsl ") || QFile(gpg).exists()) && names.empty()) {
00633     KeygenDialog d(this);
00634     if (!d.exec())
00635       return;
00636   }
00637
00638   QString passStore = ui->storePath->text();
00639
00640   if (!QFile(passStore).exists()) {
00641     // TODO(annejan) pass version?
00642     if (QMessageBox::question(
00643             this, tr("Create password-store?"),
00644             tr("Would you like to create a password-store at %1?")
00645                 .arg(passStore),
00646             QMessageBox::Yes | QMessageBox::No) == QMessageBox::Yes) {
00647       QDir().mkdir(passStore);
00648 #ifdef Q_OS_WIN
00649       SetFileAttributes(passStore.toStdWString().c_str(),
00650                         FILE_ATTRIBUTE_HIDDEN);
00651 #endif
00652       if (ui->checkBoxUseGit->isChecked())
00653         emit mainWindow->passGitInitNeeded();
00654       mainWindow->userDialog(passStore);
00655     }
00656   }
00657
00658   if (!QFile(QDir(passStore).filePath(".gpg-id")).exists()) {
00659 #ifdef QT_DEBUG
00660     dbg() « ".gpg-id file does not exist";
00661 #endif
00662     if (!clean) {
00663       criticalMessage(tr("Password store not initialised"),
00664                       tr("The folder %1 doesn't seem to be a password store or "
00665                          "is not yet initialised.")
00666                           .arg(passStore));
00667     }
00668     while (!QFile(passStore).exists()) {
00669       on_toolButtonStore_clicked();
00670       // allow user to cancel
00671       if (passStore == ui->storePath->text())
00672         return;
00673       passStore = ui->storePath->text();
00674     }
00675     if (!QFile(passStore + ".gpg-id").exists()) {
00676 #ifdef QT_DEBUG
00677       dbg() « ".gpg-id file still does not exist :/";
```

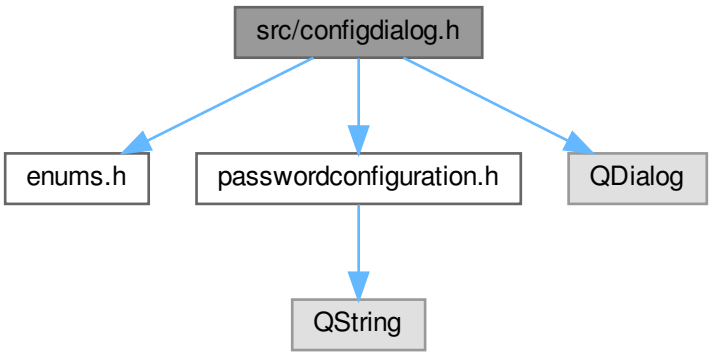
```
00678 #endif
00679       // appears not to be store
00680       // init yes / no ?
00681       mainWindow->userDialog(passStore);
00682     }
00683   }
00684
00685   ui->checkBoxHidePassword->setCheckState(Qt::Checked);
00686 }
00687
00693 void ConfigDialog::useTrayIcon(bool useSystray) {
00694   if (QSystemTrayIcon::isSystemTrayAvailable()) {
00695     ui->checkBoxUseTrayIcon->setChecked(useSystray);
00696     ui->checkBoxHideOnClose->setEnabled(useSystray);
00697     ui->checkBoxStartMinimized->setEnabled(useSystray);
00698
00699     if (!useSystray) {
00700       ui->checkBoxHideOnClose->setChecked(false);
00701       ui->checkBoxStartMinimized->setChecked(false);
00702     }
00703   }
00704 }
00705
00710 void ConfigDialog::on_checkBoxUseTrayIcon_clicked() {
00711   bool state = ui->checkBoxUseTrayIcon->isChecked();
00712   ui->checkBoxHideOnClose->setEnabled(state);
00713   ui->checkBoxStartMinimized->setEnabled(state);
00714 }
00715
00720 void ConfigDialog::closeEvent(QCloseEvent *event) {
00721   // TODO(annejan) save window size or something?
00722   event->accept();
00723 }
00724
00729 void ConfigDialog::useGit(bool useGit) {
00730   ui->checkBoxUseGit->setChecked(useGit);
00731   on_checkBoxUseGit_clicked();
00732 }
00733
00738 void ConfigDialog::useOtp(bool useOtp) {
00739   ui->checkBoxUseOtp->setChecked(useOtp);
00740 }
00741
00746 void ConfigDialog::useQrencode(bool useQrencode) {
00747   ui->checkBoxUseQrencode->setChecked(useQrencode);
00748 }
00749
00754 void ConfigDialog::on_checkBoxUseGit_clicked() {
00755   ui->checkBoxAddGPGId->setEnabled(ui->checkBoxUseGit->isChecked());
00756   ui->checkBoxAutoPull->setEnabled(ui->checkBoxUseGit->isChecked());
00757   ui->checkBoxAutoPush->setEnabled(ui->checkBoxUseGit->isChecked());
00758 }
00759
00764 void ConfigDialog::on_toolButtonPwgen_clicked() {
00765   QString pwgen = selectExecutable();
00766   if (!pwgen.isEmpty()) {
00767     ui->pwgenPath->setText(pwgen);
00768     ui->checkBoxUsePwgen->setEnabled(true);
00769   } else {
00770     ui->checkBoxUsePwgen->setEnabled(false);
00771     ui->checkBoxUsePwgen->setChecked(false);
00772   }
00773 }
00774
00780 void ConfigDialog::setPwgenPath(QString pwgen) {
00781   ui->pwgenPath->setText(pwgen);
00782   if (pwgen.isEmpty()) {
00783     ui->checkBoxUsePwgen->setChecked(false);
00784     ui->checkBoxUsePwgen->setEnabled(false);
00785   }
00786   on_checkBoxUsePwgen_clicked();
00787 }
00788
00793 void ConfigDialog::on_checkBoxUsePwgen_clicked() {
00794   bool usePwgen = ui->checkBoxUsePwgen->isChecked();
00795   ui->checkBoxAvoidCapitals->setEnabled(usePwgen);
00796   ui->checkBoxAvoidNumbers->setEnabled(usePwgen);
00797   ui->checkBoxLessRandom->setEnabled(usePwgen);
00798   ui->checkBoxUseSymbols->setEnabled(usePwgen);
00799   ui->lineEditPasswordChars->setEnabled(!usePwgen);
00800   ui->labelPasswordChars->setEnabled(!usePwgen);
00801   ui->passwordCharTemplateSelector->setEnabled(!usePwgen);
00802 }
00803
00811 void ConfigDialog::usePwgen(bool usePwgen) {
00812   if (ui->pwgenPath->text().isEmpty())
00813     usePwgen = false;
```

```
00814   ui->checkBoxUsePwgen->setChecked(usePwgen);
00815   on_checkBoxUsePwgen_clicked();
00816 }
00817
00818 void ConfigDialog::setPasswordConfiguration(
00819     const PasswordConfiguration &config) {
00820   ui->spinBoxPasswordLength->setValue(config.length);
00821   ui->passwordCharTemplateSelector->setCurrentIndex(config.selected);
00822   if (config.selected != PasswordConfiguration::CUSTOM)
00823     ui->lineEditPasswordChars->setEnabled(false);
00824   ui->lineEditPasswordChars->setText(config.Characters[config.selected]);
00825 }
00826
00827 PasswordConfiguration ConfigDialog::getPasswordConfiguration() {
00828   PasswordConfiguration config;
00829   config.length = ui->spinBoxPasswordLength->value();
00830   config.selected = static_cast<PasswordConfiguration::characterSet>(
00831       ui->passwordCharTemplateSelector->currentIndex());
00832   config.Characters[PasswordConfiguration::CUSTOM] =
00833       ui->lineEditPasswordChars->text();
00834   return config;
00835 }
00836
00843 void ConfigDialog::on_passwordCharTemplateSelector_activated(int index) {
00844   ui->lineEditPasswordChars->setText(
00845       QtPassSettings::getPasswordConfiguration().Characters[index]);
00846   if (index == 3) {
00847     ui->lineEditPasswordChars->setEnabled(true);
00848   } else {
00849     ui->lineEditPasswordChars->setEnabled(false);
00850   }
00851 }
00852
00857 void ConfigDialog::on_checkBoxUseTemplate_clicked() {
00858   ui->plainTextEditTemplate->setEnabled(ui->checkBoxUseTemplate->isChecked());
00859   ui->checkBoxTemplateAllFields->setEnabled(
00860       ui->checkBoxUseTemplate->isChecked());
00861 }
00862
00863 void ConfigDialog::onProfileTableItemChanged(QTableWidgetItem *item) {
00864   validate(item);
00865 }
00866
00871 void ConfigDialog::useTemplate(bool useTemplate) {
00872   ui->checkBoxUseTemplate->setChecked(useTemplate);
00873   on_checkBoxUseTemplate_clicked();
00874 }
```

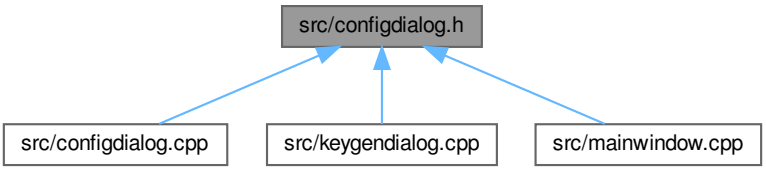
14.10 src/configdialog.h File Reference

```
#include "enums.h"
#include "passwordconfiguration.h"
#include <QDialog>
```

Include dependency graph for configdialog.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ConfigDialog

  The ConfigDialog handles the configuration interface.

Namespaces

- namespace Ui

14.11 configdialog.h

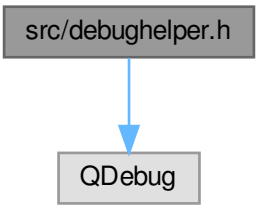
Go to the documentation of this file.

```
00001 #ifndef CONFIGDIALOG_H_
00002 #define CONFIGDIALOG_H_
00003
00004 #include "enums.h"
00005 #include "passwordconfiguration.h"
00006
```

```
00007 #include <QDialog>
00008
00009 namespace Ui {
00010 struct UserInfo;
00011
00012 class ConfigDialog;
00013 } // namespace Ui
00014
00021 class MainWindow;
00022 class QCloseEvent;
00023 class QTableWidgetItem;
00024 class ConfigDialog : public QDialog {
00025   Q_OBJECT
00026
00027 public:
00028   explicit ConfigDialog(MainWindow *parent);
00029   ~ConfigDialog();
00030
00031   void useSelection(bool useSelection);
00032   void useAutoclear(bool useAutoclear);
00033   void useAutoclearPanel(bool useAutoclearPanel);
00034   QHash<QString, QHash<QString, QString>> getProfiles();
00035   void wizard();
00036   void genKey(QString, QDialog *);
00037   void useTrayIcon(bool useSystray);
00038   void useGit(bool useGit);
00039   void useOtp(bool useOtp);
00040   void useQrencode(bool useQrencode);
00041   void setPwgenPath(QString);
00042   void usePwgen(bool usePwgen);
00043   void setPasswordConfiguration(const PasswordConfiguration &config);
00044   PasswordConfiguration getPasswordConfiguration();
00045   void useTemplate(bool useTemplate);
00046
00047 protected:
00048   void closeEvent(QCloseEvent *event);
00049
00050 private slots:
00051   void on_accepted();
00052   void on_autodetectButton_clicked();
00053   void on_radioButtonNative_clicked();
00054   void on_radioButtonPass_clicked();
00055   void on_toolButtonGit_clicked();
00056   void on_toolButtonGpg_clicked();
00057   void on_toolButtonPwgen_clicked();
00058   void on_toolButtonPass_clicked();
00059   void on_toolButtonStore_clicked();
00060   void on_comboBoxClipboard_activated(int);
00061   void on_passwordCharTemplateSelector_activated(int);
00062   void on_checkBoxSelection_clicked();
00063   void on_checkBoxAutoclear_clicked();
00064   void on_checkBoxAutoclearPanel_clicked();
00065   void on_addButton_clicked();
00066   void on_deleteButton_clicked();
00067   void on_checkBoxUseTrayIcon_clicked();
00068   void on_checkBoxUseGit_clicked();
00069   void on_checkBoxUsePwgen_clicked();
00070   void on_checkBoxUseTemplate_clicked();
00071   void onProfileTableItemChanged(QTableWidgetItem *item);
00072
00073 private:
00074   QScopedPointer<Ui::ConfigDialog> ui;
00075
00076   QStringList getSecretKeys();
00077
00078   void setGitPath(QString);
00079   void setProfiles(QHash<QString, QHash<QString, QString>, QString);
00080   void usePass(bool usePass);
00081
00082   void setGroupBoxState();
00083   QString selectExecutable();
00084   QString selectFolder();
00085   // QMessageBox::critical with hack to avoid crashes with
00086   // Qt 5.4.1 when QApplication::exec was not yet called
00087   void criticalMessage(const QString &title, const QString &text);
00088
00089   bool isPassOtpAvailable();
00090   bool isQrencodeAvailable();
00091   void validate(QTableWidgetItem *item = nullptr);
00092
00093   MainWindow *mainWindow;
00094 };
00095
00096 #endif // CONFIGDIALOG_H_
```

14.12 src/debughelper.h File Reference

`#include <QDebug>`
Include dependency graph for debughelper.h:



Macros

- #define dbg() qDebug() << __FILE__ ":" << __LINE__

14.12.1 Macro Definition Documentation

14.12.1.1 dbg

```
#define dbg( ) qDebug() << __FILE__ ":" << __LINE__
```

Definition at line 7 of file debughelper.h.

14.13 debughelper.h

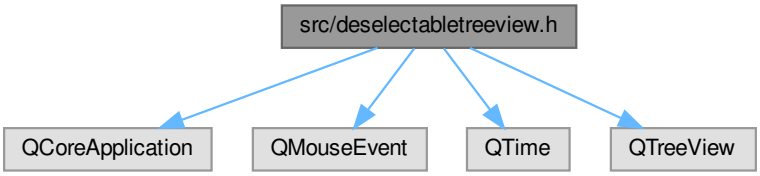
Go to the documentation of this file.

```
00001 #ifndef DEBUG_H
00002 #define DEBUG_H
00003
00004 #include <QDebug>
00005
00006 //  this is soooooo ugly...
00007 #define dbg() qDebug() « __FILE__ ":" « __LINE__
00008
00009 #endif // DEBUG_H
```

14.14 src/deselectabletreeview.h File Reference

```
#include <QCoreApplication>
#include <QMouseEvent>
#include <QTime>
#include <QTreeView>
```

Include dependency graph for deselectabletreeview.h:



Classes

- class DeselectableTreeView

  The DeselectableTreeView class loosly based on http://stackoverflow.com/questions/2761284/ thanks to Yassir Ennazk.

14.15 deselectabletreeview.h

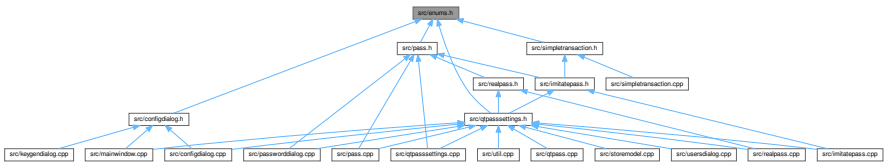
Go to the documentation of this file.

```
00001 #ifndef DESELECTABLETREEVIEW_H
00002 #define DESELECTABLETREEVIEW_H
00003
00004 #include <QCoreApplication>
00005 #include <QMouseEvent>
00006 #include <QTime>
00007 #include <QTreeView>
00008
00014 class DeselectableTreeView : public QTreeView {
00015   Q_OBJECT
00016
00017 public:
00022   DeselectableTreeView(QWidget *parent) : QTreeView(parent) {}
00026   virtual ~DeselectableTreeView() {}
00027
00028 signals:
00032   void emptyClicked();
00033
00034 private:
00035   bool doubleClickHappened = false;
00036   bool clickSelected = false;
00037
00042   virtual void mousePressEvent(QMouseEvent *event) {
00043     clickSelected = selectionModel()->isSelected(indexAt(event->pos()));
00044     QTreeView::mousePressEvent(event);
00045   }
00046
00051   void mouseReleaseEvent(QMouseEvent *event) {
00052     doubleClickHappened = false;
00053     // The timer is to distinguish between single and double click
00054     QTime dieTime = QTime::currentTime().addMSecs(200);
00055     while (QTime::currentTime() < dieTime)
00056       QCoreApplication::processEvents(QEventLoop::AllEvents, 100);
00057     // could this be done nicer?
00058     if (!doubleClickHappened && clickSelected) {
```

```
00059       QModelIndex item = indexAt(event->pos());
00060       bool selected = selectionModel()->isSelected(indexAt(event->pos()));
00061       if ((item.row() == -1 && item.column() == -1) || selected) {
00062         clearSelection();
00063         const QModelIndex index;
00064         selectionModel()->setCurrentIndex(index, QItemSelectionModel::Select);
00065         emit emptyClicked();
00066       } else {
00067         QTreeView::mouseReleaseEvent(event);
00068       }
00069     } else {
00070       QTreeView::mouseReleaseEvent(event);
00071     }
00072     clickSelected = false;
00073   }
00074
00079   void mouseDoubleClickEvent(QMouseEvent *event) {
00080     doubleClickHappened = true;
00081     QTreeView::mouseDoubleClickEvent(event);
00082   }
00083 };
00084
00085 #endif // DESELECTABLETREEVIEW_H
```

14.16 src/enums.h File Reference

This graph shows which files directly or indirectly include this file:



Namespaces

- namespace Enums

  Enumerators for configuration and runtime items.

Enumerations

- enum Enums::clipBoardType { Enums::CLIPBOARD_NEVER = 0 , Enums::CLIPBOARD_ALWAYS = 1 , Enums::CLIPBOARD_ON_DEMAND = 2 }
- enum Enums::PROCESS {
  Enums::GIT_INIT = 0 , Enums::GIT_ADD , Enums::GIT_COMMIT , Enums::GIT_RM ,
  Enums::GIT_PULL , Enums::GIT_PUSH , Enums::PASS_SHOW , Enums::PASS_INSERT ,
  Enums::PASS_REMOVE , Enums::PASS_INIT , Enums::GPG_GENKEYS , Enums::PASS_MOVE ,
  Enums::PASS_COPY , Enums::GIT_MOVE , Enums::GIT_COPY , Enums::PROCESS_COUNT ,
  Enums::INVALID , Enums::PASS_OTP_GENERATE }

14.17 enums.h

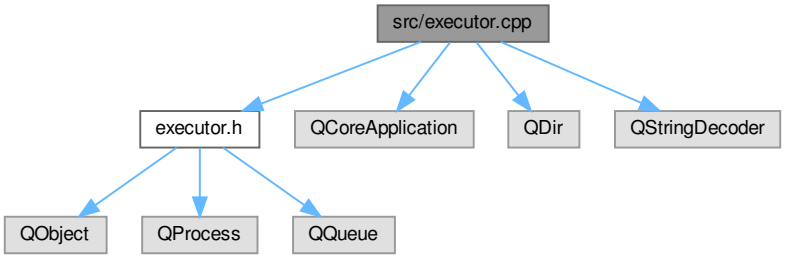
Go to the documentation of this file.

```
00001 #ifndef ENUMS_H
00002 #define ENUMS_H
00003
00008 namespace Enums {
00009
00010 enum clipBoardType {
00011   CLIPBOARD_NEVER = 0,
00012   CLIPBOARD_ALWAYS = 1,
00013   CLIPBOARD_ON_DEMAND = 2
00014 };
00015
00016 enum PROCESS {
00017   GIT_INIT = 0,
00018   GIT_ADD,
00019   GIT_COMMIT,
00020   GIT_RM,
00021   GIT_PULL,
00022   GIT_PUSH,
00023   PASS_SHOW,
00024   PASS_INSERT,
00025   PASS_REMOVE,
00026   PASS_INIT,
00027   GPG_GENKEYS,
00028   PASS_MOVE,
00029   PASS_COPY,
00030   GIT_MOVE,
00031   GIT_COPY,
00032   PROCESS_COUNT,
00033   INVALID,
00034   PASS_OTP_GENERATE,
00035 };
00036
00037 } // namespace Enums
00038
00039 #endif // ENUMS_H
```

14.18 src/executor.cpp File Reference

```
#include "executor.h"
#include <QCoreApplication>
#include <QDir>
#include <QStringDecoder>
```

Include dependency graph for executor.cpp:



14.19 executor.cpp

Go to the documentation of this file.

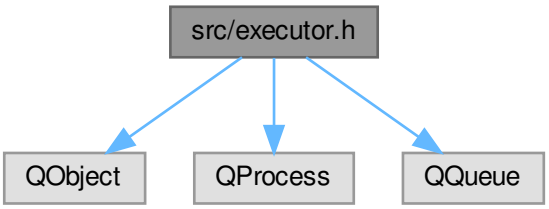
```
00001 #include "executor.h"
00002 #include <QCoreApplication>
00003 #include <QDir>
00004 #if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
00005 #include <QTextCodec>
00006 #endif
00007 #if QT_VERSION >= QT_VERSION_CHECK(6, 0, 0)
00008 #include <QStringDecoder>
00009 #endif
00010
00011 #ifdef QT_DEBUG
00012 #include "debughelper.h"
00013 #endif
00014
00019 Executor::Executor(QObject *parent) : QObject(parent), running(false) {
00020   connect(&m_process,
00021           static_cast<void (QProcess::*)(int, QProcess::ExitStatus)>(
00022               &QProcess::finished),
00023           this,
00024           static_cast<void (Executor::*)(int, QProcess::ExitStatus)>(
00025               &Executor::finished));
00026   connect(&m_process, &QProcess::started, this, &Executor::starting);
00027 }
00028
00032 void Executor::executeNext() {
00033   if (!running) {
00034     if (!m_execQueue.isEmpty()) {
00035       const execQueueItem &i = m_execQueue.head();
00036       running = true;
00037       if (!i.workingDir.isEmpty())
00038         m_process.setWorkingDirectory(i.workingDir);
00039       if (i.app.startsWith("wsl ")) {
00040         QStringList tmp = i.args;
00041         QString app = i.app;
00042         tmp.prepend(app.remove(0, 4));
00043         m_process.start("wsl", tmp);
00044       } else
00045         m_process.start(i.app, i.args);
00046       if (!i.input.isEmpty()) {
00047         m_process.waitForStarted(-1);
00048         QByteArray data = i.input.toUtf8();
00049         if (m_process.write(data) != data.length()) {
00050 #ifdef QT_DEBUG
00051           dbg() << "Not all data written to process:" << i.id << " " << i.app;
00052 #endif
00053         }
00054       }
00055       m_process.closeWriteChannel();
00056     }
00057   }
00058 }
00059
00068 void Executor::execute(int id, const QString &app, const QStringList &args,
00069                        bool readStdout, bool readStderr) {
00070   execute(id, QString(), app, args, QString(), readStdout, readStderr);
00071 }
00072
00082 void Executor::execute(int id, const QString &workDir, const QString &app,
00083                        const QStringList &args, bool readStdout,
00084                        bool readStderr) {
00085   execute(id, workDir, app, args, QString(), readStdout, readStderr);
00086 }
00087
00097 void Executor::execute(int id, const QString &app, const QStringList &args,
00098                        QString input, bool readStdout, bool readStderr) {
00099   execute(id, QString(), app, args, input, readStdout, readStderr);
00100 }
00101
00113 void Executor::execute(int id, const QString &workDir, const QString &app,
00114                        const QStringList &args, QString input, bool readStdout,
00115                        bool readStderr) {
00116   // Happens a lot if e.g. git binary is not set.
00117   // This will result in bogus "QProcess::FailedToStart" messages,
00118   // also hiding legitimate errors from the gpg commands.
00119   if (app.isEmpty()) {
00120 #ifdef QT_DEBUG
00121     dbg() << "Trying to execute nothing...";
00122 #endif
00123     return;
00124   }
00125   QString appPath = app;
00126   if (!appPath.startsWith("wsl "))
00127     appPath =
00128         QDir(QCoreApplication::applicationDirPath()).absoluteFilePath(app);
00129   m_execQueue.push_back(
00130       {id, appPath, args, input, readStdout, readStderr, workDir});
00131   executeNext();
```

```
00132 }
00133
00144 static QString decodeAssumingUtf8(QByteArray in) {
00145 #if QT_VERSION < QT_VERSION_CHECK(6, 0, 0)
00146   QTextCodec *codec = QTextCodec::codecForName("UTF-8");
00147   QTextCodec::ConverterState state;
00148   QString out = codec->toUnicode(in.constData(), in.size(), &state);
00149   if (!state.invalidChars)
00150     return out;
00151   codec = QTextCodec::codecForUtfText(in);
00152   return codec->toUnicode(in);
00153 #else
00154   auto converter = QStringDecoder(QStringDecoder::Utf8);
00155   return converter(in);
00156 #endif
00157 }
00158
00171 int Executor::executeBlocking(QString app, const QStringList &args,
00172                               QString input, QString *process_out,
00173                               QString *process_err) {
00174   QProcess internal;
00175   if (app.startsWith("wsl ")) {
00176     QStringList tmp = args;
00177     tmp.prepend(app.remove(0, 4));
00178     internal.start("wsl", tmp);
00179   } else
00180     internal.start(app, args);
00181   if (!input.isEmpty()) {
00182     QByteArray data = input.toUtf8();
00183     internal.waitForStarted(-1);
00184     if (internal.write(data) != data.length()) {
00185 #ifdef QT_DEBUG
00186       dbg() « "Not all input written:" « app;
00187 #endif
00188     }
00189     internal.closeWriteChannel();
00190   }
00191   internal.waitForFinished(-1);
00192   if (internal.exitStatus() == QProcess::NormalExit) {
00193     QString pout = decodeAssumingUtf8(internal.readAllStandardOutput());
00194     QString perr = decodeAssumingUtf8(internal.readAllStandardError());
00195     if (process_out != Q_NULLPTR)
00196       *process_out = pout;
00197     if (process_err != Q_NULLPTR)
00198       *process_err = perr;
00199     return internal.exitCode();
00200   }
00201   //  TODO(bezet): emit error() ?
00202   return -1; //    QProcess error code + qDebug error?
00203 }
00204
00213 int Executor::executeBlocking(QString app, const QStringList &args,
00214                               QString *process_out, QString *process_err) {
00215   return executeBlocking(app, args, QString(), process_out, process_err);
00216 }
00217
00223 void Executor::setEnvironment(const QStringList &env) {
00224   m_process.setEnvironment(env);
00225 }
00226
00233 int Executor::cancelNext() {
00234   if (running || m_execQueue.isEmpty())
00235     return -1; //  TODO(bezet): definitely throw here
00236   return m_execQueue.dequeue().id;
00237 }
00238
00244 void Executor::finished(int exitCode, QProcess::ExitStatus exitStatus) {
00245   execQueueItem i = m_execQueue.dequeue();
00246   running = false;
00247   if (exitStatus == QProcess::NormalExit) {
00248     QString output, err;
00249     if (i.readStdout)
00250       output = decodeAssumingUtf8(m_process.readAllStandardOutput());
00251     if (i.readStderr || exitCode != 0) {
00252       err = decodeAssumingUtf8(m_process.readAllStandardError());
00253       if (exitCode != 0) {
00254 #ifdef QT_DEBUG
00255         dbg() « exitCode « err;
00256 #endif
00257       }
00258     }
00259     emit finished(i.id, exitCode, output, err);
00260   }
00261   //    else: emit crashed with ID, which may give a chance to recover ?
00262   executeNext();
00263 }
```

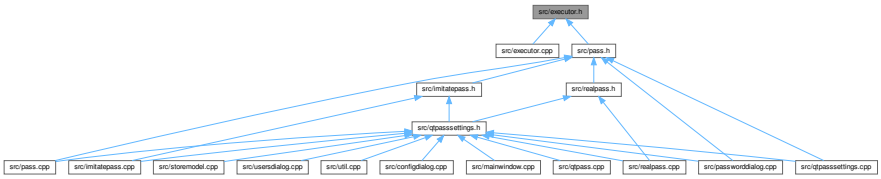
14.20 src/executor.h File Reference

```
#include <QObject>
#include <QProcess>
#include <QQueue>
```

Include dependency graph for executor.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Executor

  Executes external commands for handleing password, git and other data.

14.21 executor.h

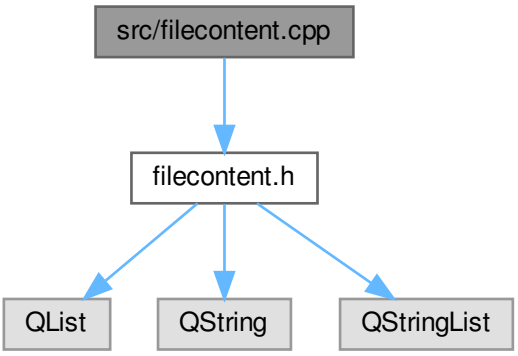
Go to the documentation of this file.

```
00001 #ifndef EXECUTOR_H
00002 #define EXECUTOR_H
00003
00004 #include <QObject>
00005 #include <QProcess>
00006 #include <QQueue>
00007
00012 class Executor : public QObject {
00013   Q_OBJECT
00014
00019   struct execQueueItem {
00023     int id;
00027     QString app;
00031     QStringList args;
00035     QString input;
00039     bool readStdout;
00045     bool readStderr;
```

```
00050     QString workingDir;
00051   };
00052
00053   QQueue<execQueueItem> m_execQueue;
00054   QProcess m_process;
00055   bool running;
00056   void executeNext();
00057
00058 public:
00059   explicit Executor(QObject *parent = 0);
00060
00061   void execute(int id, const QString &app, const QStringList &args,
00062                bool readStdout, bool readStderr = true);
00063
00064   void execute(int id, const QString &workDir, const QString &app,
00065                const QStringList &args, bool readStdout,
00066                bool readStderr = true);
00067
00068   void execute(int id, const QString &app, const QStringList &args,
00069                QString input = QString(), bool readStdout = false,
00070                bool readStderr = true);
00071
00072   void execute(int id, const QString &workDir, const QString &app,
00073                const QStringList &args, QString input = QString(),
00074                bool readStdout = false, bool readStderr = true);
00075
00076   static int executeBlocking(QString app, const QStringList &args,
00077                              QString input = QString(),
00078                              QString *process_out = Q_NULLPTR,
00079                              QString *process_err = Q_NULLPTR);
00080
00081   static int executeBlocking(QString app, const QStringList &args,
00082                              QString *process_out,
00083                              QString *process_err = Q_NULLPTR);
00084
00085   void setEnvironment(const QStringList &env);
00086
00087   int cancelNext();
00088 private slots:
00089   void finished(int exitCode, QProcess::ExitStatus exitStatus);
00090 signals:
00099   void finished(int id, int exitCode, const QString &output,
00100                 const QString &errout);
00104   void starting();
00114   void error(int id, int exitCode, const QString &output,
00115              const QString &errout);
00116 };
00117
00118 #endif // EXECUTOR_H
```

14.22 src/filecontent.cpp File Reference

`#include "filecontent.h"`
Include dependency graph for filecontent.cpp:



14.23 filecontent.cpp

Go to the documentation of this file.

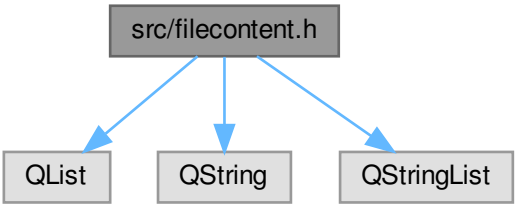
```
00001 #include "filecontent.h"
00002
00003 static bool isLineHidden(const QString &line) {
00004   return line.startsWith("otpauth://", Qt::CaseInsensitive);
00005 }
00006
00007 FileContent FileContent::parse(const QString &fileContent,
00008                                const QStringList &templateFields,
00009                                bool allFields) {
00010   QStringList lines = fileContent.split("\n");
00011   QString password = lines.takeFirst();
00012   QStringList remainingData, remainingDataDisplay;
00013   NamedValues namedValues;
00014   for (const QString &line : qAsConst(lines)) {
00015     if (line.contains(":")) {
00016       int colon = line.indexOf(':');
00017       QString name = line.left(colon);
00018       QString value = line.right(line.length() - colon - 1);
00019       if ((allFields &&
00020            !value.startsWith(
00021                "//")) // if value startswith // colon is probably from a url
00022           || templateFields.contains(name)) {
00023         namedValues.append({name.trimmed(), value.trimmed()});
00024         continue;
00025       }
00026     }
00027
00028     remainingData.append(line);
00029     if (!isLineHidden(line))
00030       remainingDataDisplay.append(line);
00031   }
00032   return FileContent(password, namedValues, remainingData.join("\n"),
00033                      remainingDataDisplay.join("\n"));
00034 }
00035
00036 QString FileContent::getPassword() const { return this->password; }
00037
00038 NamedValues FileContent::getNamedValues() const { return this->namedValues; }
00039
00040 QString FileContent::getRemainingData() const { return this->remainingData; }
00041
```

```
00042 QString FileContent::getRemainingDataForDisplay() const {
00043   return this->remainingDataDisplay;
00044 }
00045
00046 FileContent::FileContent(const QString &password,
00047                          const NamedValues &namedValues,
00048                          const QString &remainingData,
00049                          const QString &remainingDataDisplay)
00050     : password(password), namedValues(namedValues),
00051       remainingData(remainingData), remainingDataDisplay(remainingDataDisplay) {
00052 }
00053
00054 NamedValues::NamedValues() : QList() {}
00055
00056 NamedValues::NamedValues(std::initializer_list<NamedValue> values)
00057     : QList(values) {}
00058
00059 QString NamedValues::takeValue(const QString &name) {
00060   for (int i = 0; i < length(); ++i) {
00061     if (at(i).name == name) {
00062       return takeAt(i).value;
00063     }
00064   }
00065   return QString();
00066 }
```

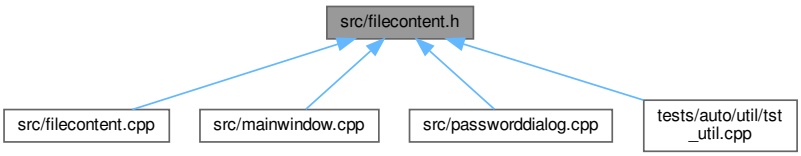
14.24 src/filecontent.h File Reference

```
#include <QList>
#include <QString>
#include <QStringList>
```

Include dependency graph for filecontent.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct NamedValue
- class NamedValues

  The NamedValues class is mostly a list of NamedValue but also has a method to take a specific NamedValue pair out of the list.
- class FileContent

14.25 filecontent.h

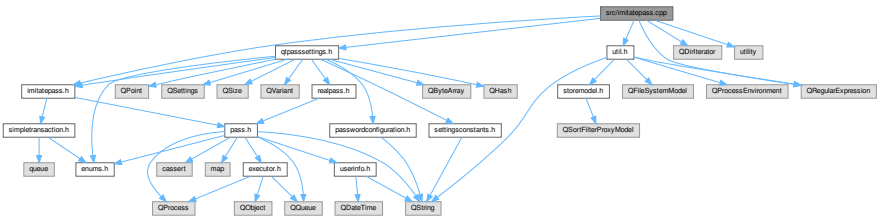
Go to the documentation of this file.

```
00001 #ifndef FILECONTENT_H
00002 #define FILECONTENT_H
00003
00004 #include <QList>
00005 #include <QString>
00006 #include <QStringList>
00007
00008 struct NamedValue {
00009   QString name;
00010   QString value;
00011 };
00012
00017 class NamedValues : public QList<NamedValue> {
00018 public:
00019   NamedValues();
00020   NamedValues(std::initializer_list<NamedValue> values);
00021
00022   QString takeValue(const QString &name);
00023 };
00024
00025 class FileContent {
00026 public:
00045   static FileContent parse(const QString &fileContent,
00046                            const QStringList &templateFields, bool allFields);
00047
00051   QString getPassword() const;
00052
00057   NamedValues getNamedValues() const;
00058
00062   QString getRemainingData() const;
00063
00068   QString getRemainingDataForDisplay() const;
00069
00070 private:
00071   FileContent(const QString &password, const NamedValues &namedValues,
00072               const QString &remainingData,
00073               const QString &remainingDataDisplay);
00074
00075   QString password;
00076   NamedValues namedValues;
00077   QString remainingData, remainingDataDisplay;
00078 };
00079
00080 #endif // FILECONTENT_H
```

14.26 src/imitatepass.cpp File Reference

```
#include "imitatepass.h"
#include "qtpasssettings.h"
#include "util.h"
#include <QDirIterator>
#include <QRegularExpression>
```

#include <utility>
Include dependency graph for imitatepass.cpp:



14.27 imitatepass.cpp

Go to the documentation of this file.

```
00001 #include "imitatepass.h"
00002 #include "qtpasssettings.h"
00003 #include "util.h"
00004 #include <QDirIterator>
00005 #include <QRegularExpression>
00006 #include <utility>
00007
00008 #ifdef QT_DEBUG
00009 #include "debughelper.h"
00010 #endif
00011
00012 using namespace Enums;
00013
00018 ImitatePass::ImitatePass() = default;
00019
00020 static QString pgit(const QString &path) {
00021   if (!QtPassSettings::getGitExecutable().startsWith("wsl "))
00022     return path;
00023   QString res = "$(wslpath " + path + ")";
00024   return res.replace('\\', '/');
00025 }
00026
00027 static QString pgpg(const QString &path) {
00028   if (!QtPassSettings::getGpgExecutable().startsWith("wsl "))
00029     return path;
00030   QString res = "$(wslpath " + path + ")";
00031   return res.replace('\\', '/');
00032 }
00033
00037 void ImitatePass::GitInit() {
00038   executeGit(GIT_INIT, {"init", pgit(QtPassSettings::getPassStore())});
00039 }
00040
00044 void ImitatePass::GitPull() { executeGit(GIT_PULL, {"pull"}); }
00045
00049 void ImitatePass::GitPull_b() {
00050   exec.executeBlocking(QtPassSettings::getGitExecutable(), {"pull"});
00051 }
00052
00056 void ImitatePass::GitPush() {
00057   if (QtPassSettings::isUseGit()) {
00058     executeGit(GIT_PUSH, {"push"});
00059   }
00060 }
00061
00065 void ImitatePass::Show(QString file) {
00066   file = QtPassSettings::getPassStore() + file + ".gpg";
00067   QStringList args = {"-d",      "--quiet",     "--yes",   "--no-encrypt-to",
00068                       "--batch", "--use-agent", pgpg(file)};
00069   executeGpg(PASS_SHOW, args);
00070 }
00071
00075 void ImitatePass::OtpGenerate(QString file) {
00076 #ifdef QT_DEBUG
00077   dbg() << "No OTP generation code for fake pass yet, attempting for file: " +
00078                file;
00079 #else
00080   Q_UNUSED(file)
00081 #endif
```

```
00082 }
00083
00091 void ImitatePass::Insert(QString file, QString newValue, bool overwrite) {
00092   file = file + ".gpg";
00093   QString gpgIdPath = Pass::getGpgIdPath(file);
00094   if (!verifyGpgIdFile(gpgIdPath)) {
00095     emit critical(tr("Check .gpgid file signature!"),
00096                   tr("Signature for %1 is invalid.").arg(gpgIdPath));
00097     return;
00098   }
00099   transactionHelper trans(this, PASS_INSERT);
00100   QStringList recipients = Pass::getRecipientList(file);
00101   if (recipients.isEmpty()) {
00102     //  TODO(bezet): probably throw here
00103     emit critical(tr("Can not edit"),
00104                   tr("Could not read encryption key to use, .gpg-id "
00105                      "file missing or invalid."));
00106     return;
00107   }
00108   QStringList args = {"--batch", "-eq", "--output", pgpg(file)};
00109   for (auto &r : recipients) {
00110     args.append("-r");
00111     args.append(r);
00112   };
00113   if (overwrite)
00114     args.append("--yes");
00115   args.append("-");
00116   executeGpg(PASS_INSERT, args, newValue);
00117   if (!QtPassSettings::isUseWebDav() && QtPassSettings::isUseGit()) {
00118     //    TODO(bezet) why not?
00119     if (!overwrite)
00120       executeGit(GIT_ADD, {"add", pgit(file)});
00121     QString path = QDir(QtPassSettings::getPassStore()).relativeFilePath(file);
00122     path.replace(Util::endsWithGpg(), "");
00123     QString msg =
00124         QString(overwrite ? "Edit" : "Add") + " for " + path + " using QtPass.";
00125     GitCommit(file, msg);
00126   }
00127 }
00128
00135 void ImitatePass::GitCommit(const QString &file, const QString &msg) {
00136   if (file.isEmpty())
00137     executeGit(GIT_COMMIT, {"commit", "-m", msg});
00138   else
00139     executeGit(GIT_COMMIT, {"commit", "-m", msg, "--", pgit(file)});
00140 }
00141
00145 void ImitatePass::Remove(QString file, bool isDir) {
00146   file = QtPassSettings::getPassStore() + file;
00147   transactionHelper trans(this, PASS_REMOVE);
00148   if (!isDir)
00149     file += ".gpg";
00150   if (QtPassSettings::isUseGit()) {
00151     executeGit(GIT_RM, {"rm", (isDir ? "-rf" : "-f"), pgit(file)});
00152     //  TODO(bezet): commit message used to have pass-like file name inside(ie.
00153     //  getFile(file, true)
00154     GitCommit(file, "Remove for " + file + " using QtPass.");
00155   } else {
00156     if (isDir) {
00157       QDir dir(file);
00158       dir.removeRecursively();
00159     } else
00160       QFile(file).remove();
00161   }
00162 }
00163
00171 void ImitatePass::Init(QString path, const QList<UserInfo> &users) {
00172 #if QT_VERSION >= QT_VERSION_CHECK(5, 15, 0)
00173   QStringList signingKeys =
00174       QtPassSettings::getPassSigningKey().split(" ", Qt::SkipEmptyParts);
00175 #else
00176   QStringList signingKeys =
00177       QtPassSettings::getPassSigningKey().split(" ", QString::SkipEmptyParts);
00178 #endif
00179   QString gpgIdSigFile = path + ".gpg-id.sig";
00180   bool addSigFile = false;
00181   if (!signingKeys.isEmpty()) {
00182     QString out;
00183     QStringList args =
00184         QStringList{"--status-fd=1", "--list-secret-keys"} + signingKeys;
00185     exec.executeBlocking(QtPassSettings::getGpgExecutable(), args, &out);
00186     bool found = false;
00187     for (auto &key : signingKeys) {
00188       if (out.contains("[GNUPG:] KEY_CONSIDERED " + key)) {
00189         found = true;
00190         break;
00191       }
```

```
00192     }
00193     if (!found) {
00194       emit critical(tr("No signing key!"),
00195                     tr("None of the secret signing keys is available.\n"
00196                        "You will not be able to change the user list!"));
00197       return;
00198     }
00199     QFileInfo checkFile(gpgIdSigFile);
00200     if (!checkFile.exists() || !checkFile.isFile())
00201       addSigFile = true;
00202   }
00203
00204   QString gpgIdFile = path + ".gpg-id";
00205   QFile gpgId(gpgIdFile);
00206   bool addFile = false;
00207   transactionHelper trans(this, PASS_INIT);
00208   if (QtPassSettings::isAddGPGId(true)) {
00209     QFileInfo checkFile(gpgIdFile);
00210     if (!checkFile.exists() || !checkFile.isFile())
00211       addFile = true;
00212   }
00213   if (!gpgId.open(QIODevice::WriteOnly | QIODevice::Text)) {
00214     emit critical(tr("Cannot update"),
00215                   tr("Failed to open .gpg-id for writing."));
00216     return;
00217   }
00218   bool secret_selected = false;
00219   foreach (const UserInfo &user, users) {
00220     if (user.enabled) {
00221       gpgId.write((user.key_id + "\n").toUtf8());
00222       secret_selected |= user.have_secret;
00223     }
00224   }
00225   gpgId.close();
00226   if (!secret_selected) {
00227     emit critical(
00228         tr("Check selected users!"),
00229         tr("None of the selected keys have a secret key available.\n"
00230            "You will not be able to decrypt any newly added passwords!"));
00231     return;
00232   }
00233
00234   if (!signingKeys.isEmpty()) {
00235     QStringList args;
00236     for (auto &key : signingKeys) {
00237       args.append(QStringList{"--default-key", key});
00238     }
00239     args.append(QStringList{"--yes", "--detach-sign", gpgIdFile});
00240     exec.executeBlocking(QtPassSettings::getGpgExecutable(), args);
00241     if (!verifyGpgIdFile(gpgIdFile)) {
00242       emit critical(tr("Check .gpgid file signature!"),
00243                     tr("Signature for %1 is invalid.").arg(gpgIdFile));
00244       return;
00245     }
00246   }
00247
00248   if (!QtPassSettings::isUseWebDav() && QtPassSettings::isUseGit() &&
00249       !QtPassSettings::getGitExecutable().isEmpty()) {
00250     if (addFile)
00251       executeGit(GIT_ADD, {"add", pgit(gpgIdFile)});
00252     QString commitPath = gpgIdFile;
00253     commitPath.replace(Util::endsWithGpg(), "");
00254     GitCommit(gpgIdFile, "Added " + commitPath + " using QtPass.");
00255     if (!signingKeys.isEmpty()) {
00256       if (addSigFile)
00257         executeGit(GIT_ADD, {"add", pgit(gpgIdSigFile)});
00258       commitPath = gpgIdSigFile;
00259       commitPath.replace(QRegularExpression("\\.gpg$"), "");
00260       GitCommit(gpgIdSigFile, "Added " + commitPath + " using QtPass.");
00261     }
00262   }
00263   reencryptPath(path);
00264 }
00265
00271 bool ImitatePass::verifyGpgIdFile(const QString &file) {
00272 #if QT_VERSION >= QT_VERSION_CHECK(5, 15, 0)
00273   QStringList signingKeys =
00274       QtPassSettings::getPassSigningKey().split(" ", Qt::SkipEmptyParts);
00275 #else
00276   QStringList signingKeys =
00277       QtPassSettings::getPassSigningKey().split(" ", QString::SkipEmptyParts);
00278 #endif
00279   if (signingKeys.isEmpty())
00280     return true;
00281   QString out;
00282   QStringList args =
00283       QStringList{"--verify", "--status-fd=1", pgpg(file) + ".sig", pgpg(file)};
```

```
00284   exec.executeBlocking(QtPassSettings::getGpgExecutable(), args, &out);
00285   QRegularExpression re(
00286       "^\\[GNUPG:\\] VALIDSIG ([A-F0-9]{40}) .* ([A-F0-9]{40})\\r?$",
00287       QRegularExpression::MultilineOption);
00288   QRegularExpressionMatch m = re.match(out);
00289   if (!m.hasMatch())
00290     return false;
00291   QStringList fingerprints = m.capturedTexts();
00292   fingerprints.removeFirst();
00293   for (auto &key : signingKeys) {
00294     if (fingerprints.contains(key))
00295       return true;
00296   }
00297   return false;
00298 }
00299
00305 bool ImitatePass::removeDir(const QString &dirName) {
00306   bool result = true;
00307   QDir dir(dirName);
00308
00309   if (dir.exists(dirName)) {
00310     Q_FOREACH (QFileInfo info,
00311                dir.entryInfoList(QDir::NoDotAndDotDot | QDir::System |
00312                                  QDir::Hidden | QDir::AllDirs | QDir::Files,
00313                                  QDir::DirsFirst)) {
00314       if (info.isDir())
00315         result = removeDir(info.absoluteFilePath());
00316       else
00317         result = QFile::remove(info.absoluteFilePath());
00318
00319       if (!result)
00320         return result;
00321     }
00322     result = dir.rmdir(dirName);
00323   }
00324   return result;
00325 }
00326
00334 void ImitatePass::reencryptPath(const QString &dir) {
00335   emit statusMsg(tr("Re-encrypting from folder %1").arg(dir), 3000);
00336   emit startReencryptPath();
00337   if (QtPassSettings::isAutoPull()) {
00338     //  TODO(bezet): move statuses inside actions?
00339     emit statusMsg(tr("Updating password-store"), 2000);
00340     GitPull_b();
00341   }
00342   QDir currentDir;
00343   QDirIterator gpgFiles(dir, QStringList() << "*.gpg", QDir::Files,
00344                         QDirIterator::Subdirectories);
00345   QStringList gpgIdFilesVerified;
00346   QStringList gpgId;
00347   while (gpgFiles.hasNext()) {
00348     QString fileName = gpgFiles.next();
00349     if (gpgFiles.fileInfo().path() != currentDir.path()) {
00350       QString gpgIdPath = Pass::getGpgIdPath(fileName);
00351       if (!gpgIdFilesVerified.contains(gpgIdPath)) {
00352         if (!verifyGpgIdFile(gpgIdPath)) {
00353           emit critical(tr("Check .gpgid file signature!"),
00354                         tr("Signature for %1 is invalid.").arg(gpgIdPath));
00355           emit endReencryptPath();
00356           return;
00357         }
00358         gpgIdFilesVerified.append(gpgIdPath);
00359       }
00360       gpgId = getRecipientList(fileName);
00361       gpgId.sort();
00362     }
00363     //  TODO(bezet): enable --with-colons for better future-proofness?
00364     QStringList args = {
00365         "-v",           "--no-secmem-warning", "--no-permission-warning",
00366         "--list-only", "--keyid-format=long", pgpg(fileName)};
00367     QString keys, err;
00368     exec.executeBlocking(QtPassSettings::getGpgExecutable(), args, &keys, &err);
00369     QStringList actualKeys;
00370     keys += err;
00371     static const QRegularExpression newLines{"[\r\n]"};
00372 #if QT_VERSION >= QT_VERSION_CHECK(5, 15, 0)
00373     QStringList key = keys.split(newLines, Qt::SkipEmptyParts);
00374 #else
00375     QStringList key = keys.split(newLines, QString::SkipEmptyParts);
00376 #endif
00377     QListIterator<QString> itr(key);
00378     while (itr.hasNext()) {
00379       QString current = itr.next();
00380       QStringList cur = current.split(" ");
00381       if (cur.length() > 4) {
00382         QString actualKey = cur.takeAt(4);
```

```
00383         if (actualKey.length() == 16) {
00384           actualKeys << actualKey;
00385         }
00386       }
00387     }
00388     actualKeys.sort();
00389     if (actualKeys != gpgId) {
00390       // dbg()<< actualKeys << gpgId << getRecipientList(fileName);
00391 #ifdef QT_DEBUG
00392       dbg() << "reencrypt " << fileName << " for " << gpgId;
00393 #endif
00394       QString local_lastDecrypt = "Could not decrypt";
00395       args = QStringList{
00396           "-d",       "--quiet",      "--yes",        "--no-encrypt-to",
00397           "--batch", "--use-agent", pgpg(fileName)};
00398       exec.executeBlocking(QtPassSettings::getGpgExecutable(), args,
00399                            &local_lastDecrypt);
00400
00401       if (!local_lastDecrypt.isEmpty() &&
00402           local_lastDecrypt != "Could not decrypt") {
00403         if (local_lastDecrypt.right(1) != "\n")
00404           local_lastDecrypt += "\n";
00405
00406         QStringList recipients = Pass::getRecipientList(fileName);
00407         if (recipients.isEmpty()) {
00408           emit critical(tr("Can not edit"),
00409                         tr("Could not read encryption key to use, .gpg-id "
00410                            "file missing or invalid."));
00411           return;
00412         }
00413         args =
00414             QStringList{"--yes", "--batch", "-eq", "--output", pgpg(fileName)};
00415         for (auto &i : recipients) {
00416           args.append("-r");
00417           args.append(i);
00418         }
00419         args.append("-");
00420         exec.executeBlocking(QtPassSettings::getGpgExecutable(), args,
00421                              local_lastDecrypt);
00422
00423         if (!QtPassSettings::isUseWebDav() && QtPassSettings::isUseGit()) {
00424           exec.executeBlocking(QtPassSettings::getGitExecutable(),
00425                                {"add", pgit(fileName)});
00426           QString path =
00427               QDir(QtPassSettings::getPassStore()).relativeFilePath(fileName);
00428           path.replace(Util::endsWithGpg(), "");
00429           exec.executeBlocking(QtPassSettings::getGitExecutable(),
00430                                {"commit", pgit(fileName), "-m",
00431                                 "Edit for " + path + " using QtPass."});
00432         }
00433
00434       } else {
00435 #ifdef QT_DEBUG
00436         dbg() << "Decrypt error on re-encrypt";
00437 #endif
00438       }
00439     }
00440   }
00441   if (QtPassSettings::isAutoPush()) {
00442     emit statusMsg(tr("Updating password-store"), 2000);
00443     //  TODO(bezet): this is non-blocking and shall be done outside
00444     GitPush();
00445   }
00446   emit endReencryptPath();
00447 }
00448
00449 void ImitatePass::Move(const QString src, const QString dest,
00450                        const bool force) {
00451   transactionHelper trans(this, PASS_MOVE);
00452   QFileInfo srcFileInfo(src);
00453   QFileInfo destFileInfo(dest);
00454   QString destFile;
00455   QString srcFileBaseName = srcFileInfo.fileName();
00456
00457   if (srcFileInfo.isFile()) {
00458     if (destFileInfo.isFile()) {
00459       if (!force) {
00460 #ifdef QT_DEBUG
00461         dbg() << "Destination file already exists";
00462 #endif
00463         return;
00464       }
00465     } else if (destFileInfo.isDir()) {
00466       destFile = QDir(dest).filePath(srcFileBaseName);
00467     } else {
00468       destFile = dest;
00469     }
```

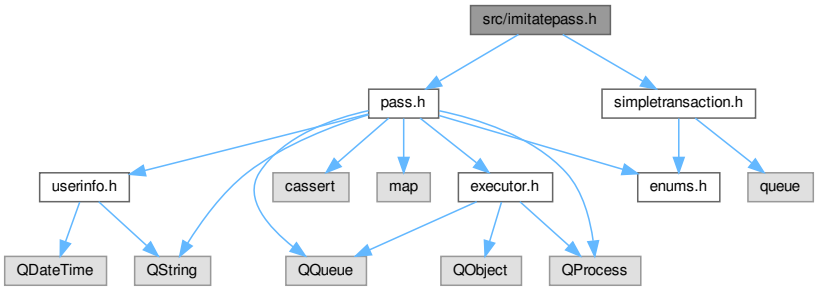
```
00470
00471     if (destFile.endsWith(".gpg", Qt::CaseInsensitive))
00472       destFile.chop(4); // make sure suffix is lowercase
00473     destFile.append(".gpg");
00474   } else if (srcFileInfo.isDir()) {
00475     if (destFileInfo.isDir()) {
00476       destFile = QDir(dest).filePath(srcFileBaseName);
00477     } else if (destFileInfo.isFile()) {
00478 #ifdef QT_DEBUG
00479       dbg() « "Destination is a file";
00480 #endif
00481       return;
00482     } else {
00483       destFile = dest;
00484     }
00485   } else {
00486 #ifdef QT_DEBUG
00487     dbg() « "Source file does not exist";
00488 #endif
00489     return;
00490   }
00491
00492 #ifdef QT_DEBUG
00493   dbg() « "Move Source: " « src;
00494   dbg() « "Move Destination: " « destFile;
00495 #endif
00496
00497   if (QtPassSettings::isUseGit()) {
00498     QStringList args;
00499     args « "mv";
00500     if (force) {
00501       args « "-f";
00502     }
00503     args « pgit(src);
00504     args « pgit(destFile);
00505     executeGit(GIT_MOVE, args);
00506
00507     QString relSrc = QDir(QtPassSettings::getPassStore()).relativeFilePath(src);
00508     relSrc.replace(Util::endsWithGpg(), "");
00509     QString relDest =
00510         QDir(QtPassSettings::getPassStore()).relativeFilePath(destFile);
00511     relDest.replace(Util::endsWithGpg(), "");
00512     QString message = QString("Moved for %1 to %2 using QtPass.");
00513     message = message.arg(relSrc, relDest);
00514     GitCommit("", message);
00515   } else {
00516     QDir qDir;
00517     if (force) {
00518       qDir.remove(destFile);
00519     }
00520     qDir.rename(src, destFile);
00521   }
00522 }
00523
00524 void ImitatePass::Copy(const QString src, const QString dest,
00525                        const bool force) {
00526   QFileInfo destFileInfo(dest);
00527   transactionHelper trans(this, PASS_COPY);
00528   if (QtPassSettings::isUseGit()) {
00529     QStringList args;
00530     args « "cp";
00531     if (force) {
00532       args « "-f";
00533     }
00534     args « pgit(src);
00535     args « pgit(dest);
00536     executeGit(GIT_COPY, args);
00537
00538     QString message = QString("copied from %1 to %2 using QTPass.");
00539     message = message.arg(src, dest);
00540     GitCommit("", message);
00541   } else {
00542     QDir qDir;
00543     if (force) {
00544       qDir.remove(dest);
00545     }
00546     QFile::copy(src, dest);
00547   }
00548   // reecrypt all files under the new folder
00549   if (destFileInfo.isDir()) {
00550     reencryptPath(destFileInfo.absoluteFilePath());
00551   } else if (destFileInfo.isFile()) {
00552     reencryptPath(destFileInfo.dir().path());
00553   }
00554 }
00555
00560 void ImitatePass::executeGpg(PROCESS id, const QStringList &args, QString input,
```

```
00561                             bool readStdout, bool readStderr) {
00562   executeWrapper(id, QtPassSettings::getGpgExecutable(), args, std::move(input),
00563                  readStdout, readStderr);
00564 }
00569 void ImitatePass::executeGit(PROCESS id, const QStringList &args, QString input,
00570                              bool readStdout, bool readStderr) {
00571   executeWrapper(id, QtPassSettings::getGitExecutable(), args, std::move(input),
00572                  readStdout, readStderr);
00573 }
00574
00585 void ImitatePass::finished(int id, int exitCode, const QString &out,
00586                            const QString &err) {
00587 #ifdef QT_DEBUG
00588   dbg() « "Imitate Pass";
00589 #endif
00590   static QString transactionOutput;
00591   PROCESS pid = transactionIsOver(static_cast<PROCESS>(id));
00592   transactionOutput.append(out);
00593
00594   if (exitCode == 0) {
00595     if (pid == INVALID)
00596       return;
00597   } else {
00598     while (pid == INVALID) {
00599       id = exec.cancelNext();
00600       if (id == -1) {
00601         //  this is probably irrecoverable and shall not happen
00602 #ifdef QT_DEBUG
00603         dbg() « "No such transaction!";
00604 #endif
00605         return;
00606       }
00607       pid = transactionIsOver(static_cast<PROCESS>(id));
00608     }
00609   }
00610   Pass::finished(pid, exitCode, transactionOutput, err);
00611   transactionOutput.clear();
00612 }
00613
00623 void ImitatePass::executeWrapper(PROCESS id, const QString &app,
00624                                  const QStringList &args, QString input,
00625                                  bool readStdout, bool readStderr) {
00626   transactionAdd(id);
00627   Pass::executeWrapper(id, app, args, input, readStdout, readStderr);
00628 }
```

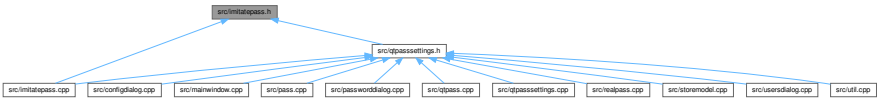
14.28 src/imitatepass.h File Reference

```
#include "pass.h"
#include "simpletransaction.h"
```

Include dependency graph for imitatepass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class ImitatePass

  Imitates pass features when pass is not enabled or available.

14.29 imitatepass.h

Go to the documentation of this file.

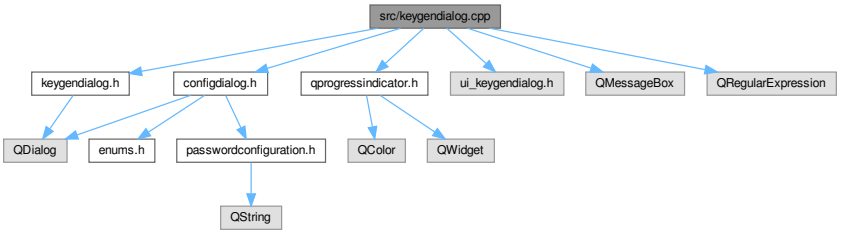
```
00001 #ifndef IMITATEPASS_H
00002 #define IMITATEPASS_H
00003
00004 #include "pass.h"
00005 #include "simpletransaction.h"
00006
00011 class ImitatePass : public Pass, private simpleTransaction {
00012   Q_OBJECT
00013
00014   bool verifyGpgIdFile(const QString &file);
00015   bool removeDir(const QString &dirName);
00016
00017   void GitCommit(const QString &file, const QString &msg);
00018
00019   void executeGit(PROCESS id, const QStringList &args,
00020                   QString input = QString(), bool readStdout = true,
00021                   bool readStderr = true);
00022   void executeGpg(PROCESS id, const QStringList &args,
00023                   QString input = QString(), bool readStdout = true,
00024                   bool readStderr = true);
00025
00026   class transactionHelper {
00027     simpleTransaction *m_transaction;
00028     PROCESS m_result;
00029
00030   public:
00031     transactionHelper(simpleTransaction *trans, PROCESS result)
00032         : m_transaction(trans), m_result(result) {
00033       m_transaction->transactionStart();
00034     }
00035     ~transactionHelper() { m_transaction->transactionEnd(m_result); }
00036   };
00037
00038 protected:
00039   virtual void finished(int id, int exitCode, const QString &out,
00040                         const QString &err) Q_DECL_OVERRIDE;
00041
00042   virtual void executeWrapper(PROCESS id, const QString &app,
00043                               const QStringList &args, QString input,
00044                               bool readStdout = true,
00045                               bool readStderr = true) Q_DECL_OVERRIDE;
00046
00047 public:
00048   ImitatePass();
00049   virtual ~ImitatePass() {}
00050   virtual void GitInit() Q_DECL_OVERRIDE;
00051   virtual void GitPull() Q_DECL_OVERRIDE;
00052   virtual void GitPull_b() Q_DECL_OVERRIDE;
00053   virtual void GitPush() Q_DECL_OVERRIDE;
00054   virtual void Show(QString file) Q_DECL_OVERRIDE;
00055   virtual void OtpGenerate(QString file) Q_DECL_OVERRIDE;
00056   virtual void Insert(QString file, QString newValue,
00057                       bool overwrite = false) Q_DECL_OVERRIDE;
00058   virtual void Remove(QString file, bool isDir = false) Q_DECL_OVERRIDE;
00059   virtual void Init(QString path, const QList<UserInfo> &users) Q_DECL_OVERRIDE;
00060
```

```
00061   void reencryptPath(const QString &dir);
00062 signals:
00063   void startReencryptPath();
00064   void endReencryptPath();
00065
00066   // Pass interface
00067 public:
00068   void Move(const QString src, const QString dest,
00069             const bool force = false) Q_DECL_OVERRIDE;
00070   void Copy(const QString src, const QString dest,
00071             const bool force = false) Q_DECL_OVERRIDE;
00072 };
00073
00074 #endif // IMITATEPASS_H
```

14.30 src/keygendialog.cpp File Reference

```
#include "keygendialog.h"
#include "configdialog.h"
#include "qprogressindicator.h"
#include "ui_keygendialog.h"
#include <QMessageBox>
#include <QRegularExpression>
```

Include dependency graph for keygendialog.cpp:



14.31 keygendialog.cpp

Go to the documentation of this file.

```
00001 #include "keygendialog.h"
00002 #include "configdialog.h"
00003 #include "qprogressindicator.h"
00004 #include "ui_keygendialog.h"
00005 #include <QMessageBox>
00006 #include <QRegularExpression>
00007
00008 #ifdef QT_DEBUG
00009 #include "debughelper.h"
00010 #endif
00011
00016 KeygenDialog::KeygenDialog(ConfigDialog *parent)
00017     : QDialog(parent), ui(new Ui::KeygenDialog) {
00018   ui->setupUi(this);
00019   dialog = parent;
00020 }
00021
00025 KeygenDialog::~KeygenDialog() { delete ui; }
00026
00032 void KeygenDialog::on_passphrase1_textChanged(const QString &arg1) {
00033   bool state = ui->passphrase1->text() == ui->passphrase2->text();
00034   if (state) {
00035     replace("Passphrase", arg1);
00036     no_protection(arg1.isEmpty());
00037   }
00038
```

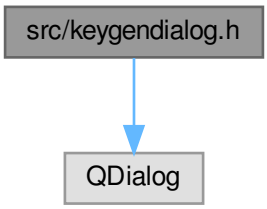
```
00039   ui->buttonBox->setEnabled(state);
00040 }
00041
00047 void KeygenDialog::on_passphrase2_textChanged(const QString &arg1) {
00048   on_passphrase1_textChanged(arg1);
00049 }
00050
00055 void KeygenDialog::on_checkBox_stateChanged(int arg1) {
00056   ui->plainTextEdit->setReadOnly(!arg1);
00057   ui->plainTextEdit->setEnabled(arg1);
00058 }
00059
00065 void KeygenDialog::on_email_textChanged(const QString &arg1) {
00066   replace("Name-Email", arg1);
00067 }
00068
00074 void KeygenDialog::on_name_textChanged(const QString &arg1) {
00075   replace("Name-Real", arg1);
00076 }
00077
00084 void KeygenDialog::replace(const QString &key, const QString &value) {
00085   QStringList clear;
00086   QString expert = ui->plainTextEdit->toPlainText();
00087   static const QRegularExpression newLines{"[\r\n]"};
00088 #if QT_VERSION >= QT_VERSION_CHECK(5, 15, 0)
00089   const QStringList lines = expert.split(newLines, Qt::SkipEmptyParts);
00090 #else
00091   const QStringList lines = expert.split(newLines, QString::SkipEmptyParts);
00092 #endif
00093   for (QString line : lines) {
00094     line.replace(QRegularExpression(key + ":.*"), key + ": " + value);
00095     if (key == "Passphrase")
00096       line.replace("%no-protection", "Passphrase: " + value);
00097     clear.append(line);
00098   }
00099   ui->plainTextEdit->setPlainText(clear.join("\n"));
00100 }
00101
00107 void KeygenDialog::no_protection(bool enable) {
00108   QStringList clear;
00109   QString expert = ui->plainTextEdit->toPlainText();
00110   static const QRegularExpression newLines{"[\r\n]"};
00111 #if QT_VERSION >= QT_VERSION_CHECK(5, 15, 0)
00112   const QStringList lines = expert.split(newLines, Qt::SkipEmptyParts);
00113 #else
00114   const QStringList lines = expert.split(newLines, QString::SkipEmptyParts);
00115 #endif
00116   for (QString line : lines) {
00117     bool remove = false;
00118     if (!enable) {
00119       if (line.indexOf("%no-protection") == 0)
00120         remove = true;
00121     } else {
00122       if (line.indexOf("Passphrase") == 0)
00123         line = "%no-protection";
00124     }
00125     if (!remove)
00126       clear.append(line);
00127   }
00128   ui->plainTextEdit->setPlainText(clear.join("\n"));
00129 }
00130
00136 void KeygenDialog::done(int r) {
00137   if (QDialog::Accepted == r) { // ok was pressed
00138     // check name
00139     if (ui->name->text().length() < 5) {
00140       QMessageBox::critical(this, tr("Invalid name"),
00141                             tr("Name must be at least 5 characters long."));
00142       return;
00143     }
00144
00145     // check email
00146     static const QRegularExpression mailre(
00147         QRegularExpression::anchoredPattern(
00148             R"(\b[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}\b)"),
00149         QRegularExpression::CaseInsensitiveOption);
00150     if (!mailre.match(ui->email->text()).hasMatch()) {
00151       QMessageBox::critical(
00152           this, tr("Invalid email"),
00153           tr("The email address you typed is not a valid email address."));
00154       return;
00155     }
00156
00157     ui->widget->setEnabled(false);
00158     ui->buttonBox->setEnabled(false);
00159     ui->checkBox->setEnabled(false);
00160     ui->plainTextEdit->setEnabled(false);
```

```
00161
00162     auto *pi = new QProgressIndicator();
00163     pi->startAnimation();
00164     pi->setSizePolicy(QSizePolicy::Expanding, QSizePolicy::Expanding);
00165
00166     ui->frame->hide();
00167     ui->label->setText(
00168         tr("This operation can take some minutes.<br />"
00169            "We need to generate a lot of random bytes. It is a good idea to "
00170            "perform some other action (type on the keyboard, move the mouse, "
00171            "utilize the disks) during the prime generation; this gives the "
00172            "random number generator a better chance to gain enough entropy."));
00173
00174     this->layout()->addWidget(pi);
00175
00176     this->show();
00177     dialog->genKey(ui->plainTextEdit->toPlainText(), this);
00178   } else { // cancel, close or exc was pressed
00179     QDialog::done(r);
00180     return;
00181   }
00182 }
00183
00188 void KeygenDialog::closeEvent(QCloseEvent *event) {
00189   // TODO(annejan) save window size or somethign
00190   event->accept();
00191 }
```

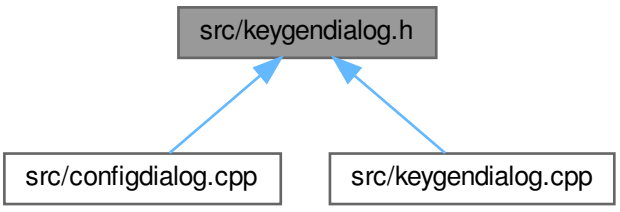
14.32 src/keygendialog.h File Reference

`#include <QDialog>`

Include dependency graph for keygendialog.h:



This graph shows which files directly or indirectly include this file:



Classes

- class KeygenDialog

  Handles GPG keypair generation.

Namespaces

- namespace Ui

14.33 keygendialog.h

Go to the documentation of this file.

```
00001 #ifndef KEYGENDIALOG_H_
00002 #define KEYGENDIALOG_H_
00003
00004 #include <QDialog>
00005
00006 namespace Ui {
00007 class KeygenDialog;
00008 }
00009
00014 class ConfigDialog;
00015 class QCloseEvent;
00016 class KeygenDialog : public QDialog {
00017   Q_OBJECT
00018
00019 public:
00020   explicit KeygenDialog(ConfigDialog *parent = 0);
00021   ~KeygenDialog();
00022
00023 protected:
00024   void closeEvent(QCloseEvent *event);
00025
00026 private slots:
00027   void on_passphrase1_textChanged(const QString &arg1);
00028   void on_passphrase2_textChanged(const QString &arg1);
00029   void on_checkBox_stateChanged(int arg1);
00030   void on_email_textChanged(const QString &arg1);
00031   void on_name_textChanged(const QString &arg1);
00032
00033 private:
00034   Ui::KeygenDialog *ui;
00035   void replace(const QString &, const QString &);
00036   void done(int r);
00037   void no_protection(bool enable);
00038   ConfigDialog *dialog;
00039 };
00040
00041 #endif // KEYGENDIALOG_H_
```

14.34 src/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "configdialog.h"
#include "filecontent.h"
#include "passworddialog.h"
#include "qpushbuttonasqrcode.h"
#include "qpushbuttonshowpassword.h"
#include "qpushbuttonwithclipboard.h"
#include "qtpass.h"
#include "qtpasssettings.h"
#include "trayicon.h"
#include "ui_mainwindow.h"
#include "usersdialog.h"
```

```
#include "util.h"
#include <QCloseEvent>
#include <QDesktopServices>
#include <QDialog>
#include <QFileInfo>
#include <QInputDialog>
#include <QLabel>
#include <QMenu>
#include <QMessageBox>
#include <QShortcut>
#include <QTimer>
```

Include dependency graph for mainwindow.cpp:

14.35 mainwindow.cpp

Go to the documentation of this file.

```
00001 #include "mainwindow.h"
00002
00003 #ifdef QT_DEBUG
00004 #include "debughelper.h"
00005 #endif
00006
00007 #include "configdialog.h"
00008 #include "filecontent.h"
00009 #include "passworddialog.h"
00010 #include "qpushbuttonasqrcode.h"
00011 #include "qpushbuttonshowpassword.h"
00012 #include "qpushbuttonwithclipboard.h"
00013 #include "qtpass.h"
00014 #include "qtpasssettings.h"
00015 #include "trayicon.h"
00016 #include "ui_mainwindow.h"
00017 #include "usersdialog.h"
00018 #include "util.h"
00019 #include <QCloseEvent>
00020 #include <QDesktopServices>
00021 #include <QDialog>
00022 #include <QFileInfo>
00023 #include <QInputDialog>
00024 #include <QLabel>
00025 #include <QMenu>
00026 #include <QMessageBox>
00027 #include <QShortcut>
00028 #include <QTimer>
00029
00036 MainWindow::MainWindow(const QString &searchText, QWidget *parent)
00037     : QMainWindow(parent), ui(new Ui::MainWindow), keygen(nullptr),
00038       tray(nullptr) {
00039 #ifdef __APPLE__
00040   // extra treatment for mac os
00041   // see http://doc.qt.io/qt-5/qkeysequence.html#qt_set_sequence_auto_mnemonic
00042   qt_set_sequence_auto_mnemonic(true);
00043 #endif
00044   ui->setupUi(this);
00045
00046   m_qtPass = new QtPass(this);
00047
00048   // register shortcut ctrl/cmd + Q to close the main window
00049   new QShortcut(QKeySequence(Qt::CTRL | Qt::Key_Q), this, SLOT(close()));
00050   // register shortcut ctrl/cmd + C to copy the currently selected password
00051   new QShortcut(QKeySequence(QKeySequence::StandardKey::Copy), this,
00052                 SLOT(copyPasswordFromTreeview()));
00053
00054   model.setNameFilters(QStringList() << "*.gpg");
00055   model.setNameFilterDisables(false);
00056
```

```
00057   /*
00058    * I added this to solve Windows bug but now on GNU/Linux the main folder,
00059    * if hidden, disappear
00060    *
00061    * model.setFilter(QDir::NoDot);
00062    */
00063
00064   QString passStore = QtPassSettings::getPassStore(Util::findPasswordStore());
00065
00066   QModelIndex rootDir = model.setRootPath(passStore);
00067   model.fetchMore(rootDir);
00068
00069   proxyModel.setModelAndStore(&model, passStore);
00070   selectionModel.reset(new QItemSelectionModel(&proxyModel));
00071
00072   ui->treeView->setModel(&proxyModel);
00073   ui->treeView->setRootIndex(proxyModel.mapFromSource(rootDir));
00074   ui->treeView->setColumnHidden(1, true);
00075   ui->treeView->setColumnHidden(2, true);
00076   ui->treeView->setColumnHidden(3, true);
00077   ui->treeView->setHeaderHidden(true);
00078   ui->treeView->setIndentation(15);
00079   ui->treeView->setHorizontalScrollBarPolicy(Qt::ScrollBarAsNeeded);
00080   ui->treeView->setContextMenuPolicy(Qt::CustomContextMenu);
00081   ui->treeView->header()->setSectionResizeMode(0, QHeaderView::Stretch);
00082   ui->treeView->sortByColumn(0, Qt::AscendingOrder);
00083   connect(ui->treeView, &QWidget::customContextMenuRequested, this,
00084           &MainWindow::showContextMenu);
00085   connect(ui->treeView, &DeselectableTreeView::emptyClicked, this,
00086           &MainWindow::deselect);
00087
00088   if (QtPassSettings::isUseMonospace()) {
00089     ui->textBrowser->setFont(QFont(QStringLiteral("Monospace")));
00090   }
00091   if (QtPassSettings::isNoLineWrapping()) {
00092     ui->textBrowser->setLineWrapMode(QTextBrowser::NoWrap);
00093   }
00094   ui->textBrowser->setOpenExternalLinks(true);
00095   ui->textBrowser->setContextMenuPolicy(Qt::CustomContextMenu);
00096   connect(ui->textBrowser, &QWidget::customContextMenuRequested, this,
00097           &MainWindow::showBrowserContextMenu);
00098
00099   updateProfileBox();
00100
00101   QtPassSettings::getPass()->updateEnv();
00102   clearPanelTimer.setInterval(1000 *
00103                               QtPassSettings::getAutoclearPanelSeconds());
00104   clearPanelTimer.setSingleShot(true);
00105   connect(&clearPanelTimer, SIGNAL(timeout()), this, SLOT(clearPanel()));
00106
00107   searchTimer.setInterval(350);
00108   searchTimer.setSingleShot(true);
00109
00110   connect(&searchTimer, &QTimer::timeout, this, &MainWindow::onTimeoutSearch);
00111
00112   initToolBarButtons();
00113   initStatusBar();
00114
00115   ui->lineEdit->setClearButtonEnabled(true);
00116
00117   setUiElementsEnabled(true);
00118
00119   QTimer::singleShot(10, this, SLOT(focusInput()));
00120
00121   ui->lineEdit->setText(searchText);
00122
00123   if (!m_qtPass->init())
00124     // no working config so this should just quit
00125     QApplication::quit();
00126 }
00127
00128 MainWindow::~MainWindow() { delete m_qtPass; }
00129
00136 void MainWindow::focusInput() {
00137   ui->lineEdit->selectAll();
00138   ui->lineEdit->setFocus();
00139 }
00140
00145 void MainWindow::changeEvent(QEvent *event) {
00146   QWidget::changeEvent(event);
00147   if (event->type() == QEvent::ActivationChange) {
00148     if (isActiveWindow()) {
00149       focusInput();
00150     }
00151   }
00152 }
00153
```

```
00157 void MainWindow::initToolBarButtons() {
00158   connect(ui->actionAddPassword, &QAction::triggered, this,
00159           &MainWindow::addPassword);
00160   connect(ui->actionAddFolder, &QAction::triggered, this,
00161           &MainWindow::addFolder);
00162   connect(ui->actionEdit, &QAction::triggered, this, &MainWindow::onEdit);
00163   connect(ui->actionDelete, &QAction::triggered, this, &MainWindow::onDelete);
00164   connect(ui->actionPush, &QAction::triggered, this, &MainWindow::onPush);
00165   connect(ui->actionUpdate, &QAction::triggered, this, &MainWindow::onUpdate);
00166   connect(ui->actionUsers, &QAction::triggered, this, &MainWindow::onUsers);
00167   connect(ui->actionConfig, &QAction::triggered, this, &MainWindow::onConfig);
00168   connect(ui->actionOtp, &QAction::triggered, this, &MainWindow::onOtp);
00169
00170   ui->actionAddPassword->setIcon(
00171       QIcon::fromTheme("document-new", QIcon(":/icons/document-new.svg")));
00172   ui->actionAddFolder->setIcon(
00173       QIcon::fromTheme("folder-new", QIcon(":/icons/folder-new.svg")));
00174   ui->actionEdit->setIcon(QIcon::fromTheme(
00175       "document-properties", QIcon(":/icons/document-properties.svg")));
00176   ui->actionDelete->setIcon(
00177       QIcon::fromTheme("edit-delete", QIcon(":/icons/edit-delete.svg")));
00178   ui->actionPush->setIcon(
00179       QIcon::fromTheme("go-up", QIcon(":/icons/go-top.svg")));
00180   ui->actionUpdate->setIcon(
00181       QIcon::fromTheme("go-down", QIcon(":/icons/go-bottom.svg")));
00182   ui->actionUsers->setIcon(QIcon::fromTheme(
00183       "x-office-address-book", QIcon(":/icons/x-office-address-book.svg")));
00184   ui->actionConfig->setIcon(QIcon::fromTheme(
00185       "applications-system", QIcon(":/icons/applications-system.svg")));
00186 }
00187
00191 void MainWindow::initStatusBar() {
00192   ui->statusBar->showMessage(tr("Welcome to QtPass %1").arg(VERSION), 2000);
00193
00194   QPixmap logo = QPixmap::fromImage(QImage(":/artwork/icon.svg"))
00195                      .scaledToHeight(statusBar()->height());
00196   QLabel *logoApp = new QLabel(statusBar());
00197   logoApp->setPixmap(logo);
00198   statusBar()->addPermanentWidget(logoApp);
00199 }
00200
00201 const QModelIndex MainWindow::getCurrentTreeViewIndex() {
00202   return ui->treeView->currentIndex();
00203 }
00204
00205 void MainWindow::cleanKeygenDialog() {
00206   this->keygen->close();
00207   this->keygen = nullptr;
00208 }
00209
00210 void MainWindow::flashText(const QString &text, const bool isError,
00211                            const bool isHtml) {
00212   if (isError)
00213     ui->textBrowser->setTextColor(Qt::red);
00214
00215   if (isHtml) {
00216     QString _text = text;
00217     if (!ui->textBrowser->toPlainText().isEmpty())
00218       _text = ui->textBrowser->toHtml() + _text;
00219     ui->textBrowser->setHtml(_text);
00220   } else {
00221     ui->textBrowser->setText(text);
00222     ui->textBrowser->setTextColor(Qt::black);
00223   }
00224 }
00225
00230 void MainWindow::config() {
00231   QScopedPointer<ConfigDialog> d(new ConfigDialog(this));
00232   d->setModal(true);
00233   // Automatically default to pass if it's available
00234   if (m_qtPass->isFreshStart() &&
00235       QFile(QtPassSettings::getPassExecutable()).exists()) {
00236     QtPassSettings::setUsePass(true);
00237   }
00238
00239   if (m_qtPass->isFreshStart())
00240     d->wizard(); // does shit
00241   if (d->exec()) {
00242     if (d->result() == QDialog::Accepted) {
00243       // Update the textBrowser font
00244       if (QtPassSettings::isUseMonospace()) {
00245         ui->textBrowser->setFont(QFont(QStringLiteral("Monospace")));
00246       } else {
00247         ui->textBrowser->setFont(QFont());
00248       }
00249       // Update the textBrowser line wrap mode
00250       if (QtPassSettings::isNoLineWrapping()) {
```

```
00251         ui->textBrowser->setLineWrapMode(QTextBrowser::NoWrap);
00252       } else {
00253         ui->textBrowser->setLineWrapMode(QTextBrowser::WidgetWidth);
00254       }
00255
00256       if (QtPassSettings::isAlwaysOnTop()) {
00257         Qt::WindowFlags flags = windowFlags();
00258         this->setWindowFlags(flags | Qt::WindowStaysOnTopHint);
00259       } else {
00260         this->setWindowFlags(Qt::Window);
00261       }
00262       this->show();
00263
00264       updateProfileBox();
00265       ui->treeView->setRootIndex(proxyModel.mapFromSource(
00266           model.setRootPath(QtPassSettings::getPassStore())));
00267
00268       if (m_qtPass->isFreshStart() && Util::checkConfig())
00269         config();
00270       QtPassSettings::getPass()->updateEnv();
00271       clearPanelTimer.setInterval(1000 *
00272                                   QtPassSettings::getAutoclearPanelSeconds());
00273       m_qtPass->setClipboardTimer();
00274
00275       updateGitButtonVisibility();
00276       updateOtpButtonVisibility();
00277       if (QtPassSettings::isUseTrayIcon() && tray == nullptr)
00278         initTrayIcon();
00279       else if (!QtPassSettings::isUseTrayIcon() && tray != nullptr) {
00280         destroyTrayIcon();
00281       }
00282     }
00283
00284     m_qtPass->setFreshStart(false);
00285   }
00286 }
00287
00291 void MainWindow::onUpdate(bool block) {
00292   ui->statusBar->showMessage(tr("Updating password-store"), 2000);
00293   if (block)
00294     QtPassSettings::getPass()->GitPull_b();
00295   else
00296     QtPassSettings::getPass()->GitPull();
00297 }
00298
00302 void MainWindow::onPush() {
00303   if (QtPassSettings::isUseGit()) {
00304     ui->statusBar->showMessage(tr("Updating password-store"), 2000);
00305     QtPassSettings::getPass()->GitPush();
00306   }
00307 }
00308
00316 QString MainWindow::getFile(const QModelIndex &index, bool forPass) {
00317   if (!index.isValid() ||
00318       !model.fileInfo(proxyModel.mapToSource(index)).isFile())
00319     return QString();
00320   QString filePath = model.filePath(proxyModel.mapToSource(index));
00321   if (forPass) {
00322     filePath = QDir(QtPassSettings::getPassStore()).relativeFilePath(filePath);
00323     filePath.replace(Util::endsWithGpg(), "");
00324   }
00325   return filePath;
00326 }
00327
00332 void MainWindow::on_treeView_clicked(const QModelIndex &index) {
00333   bool cleared = ui->treeView->currentIndex().flags() == Qt::NoItemFlags;
00334   currentDir =
00335       Util::getDir(ui->treeView->currentIndex(), false, model, proxyModel);
00336   //    TODO(bezet): "Could not decrypt";
00337   m_qtPass->clearClippedText();
00338   QString file = getFile(index, true);
00339   ui->passwordName->setText(getFile(index, true));
00340   if (!file.isEmpty() && !cleared) {
00341     QtPassSettings::getPass()->Show(file);
00342   } else {
00343     clearPanel(false);
00344     ui->actionEdit->setEnabled(false);
00345     ui->actionDelete->setEnabled(true);
00346   }
00347 }
00348
00354 void MainWindow::on_treeView_doubleClicked(const QModelIndex &index) {
00355   QFileInfo fileOrFolder =
00356       model.fileInfo(proxyModel.mapToSource(ui->treeView->currentIndex()));
00357
00358   if (fileOrFolder.isFile()) {
00359     editPassword(getFile(index, true));
```

```
00360   }
00361 }
00362
00366 void MainWindow::deselect() {
00367   currentDir = "";
00368   m_qtPass->clearClipboard();
00369   ui->treeView->clearSelection();
00370   ui->actionEdit->setEnabled(false);
00371   ui->actionDelete->setEnabled(false);
00372   ui->passwordName->setText("");
00373   clearPanel(false);
00374 }
00375
00376 void MainWindow::executeWrapperStarted() {
00377   clearTemplateWidgets();
00378   ui->textBrowser->clear();
00379   setUiElementsEnabled(false);
00380   clearPanelTimer.stop();
00381 }
00382
00383 void MainWindow::passShowHandler(const QString &p_output) {
00384   QStringList templ = QtPassSettings::isUseTemplate()
00385                           ? QtPassSettings::getPassTemplate().split("\n")
00386                           : QStringList();
00387   bool allFields =
00388       QtPassSettings::isUseTemplate() && QtPassSettings::isTemplateAllFields();
00389   FileContent fileContent = FileContent::parse(p_output, templ, allFields);
00390   QString output = p_output;
00391   QString password = fileContent.getPassword();
00392
00393   // set clipped text
00394   m_qtPass->setClippedText(password, p_output);
00395
00396   // first clear the current view:
00397   clearTemplateWidgets();
00398
00399   // show what is needed:
00400   if (QtPassSettings::isHideContent()) {
00401     output = "***" + tr("Content hidden") + "***";
00402   } else if (!QtPassSettings::isDisplayAsIs()) {
00403     if (!password.isEmpty()) {
00404       // set the password, it is hidden if needed in addToGridLayout
00405       addToGridLayout(0, tr("Password"), password);
00406     }
00407
00408     NamedValues namedValues = fileContent.getNamedValues();
00409     for (int j = 0; j < namedValues.length(); ++j) {
00410       NamedValue nv = namedValues.at(j);
00411       addToGridLayout(j + 1, nv.name, nv.value);
00412     }
00413     if (ui->gridLayout->count() == 0)
00414       ui->verticalLayoutPassword->setSpacing(0);
00415     else
00416       ui->verticalLayoutPassword->setSpacing(6);
00417
00418     output = fileContent.getRemainingDataForDisplay();
00419   }
00420
00421   if (QtPassSettings::isUseAutoclearPanel()) {
00422     clearPanelTimer.start();
00423   }
00424
00425   emit passShowHandlerFinished(output);
00426   setUiElementsEnabled(true);
00427 }
00428
00429 void MainWindow::passOtpHandler(const QString &p_output) {
00430   if (!p_output.isEmpty()) {
00431     addToGridLayout(ui->gridLayout->count() + 1, tr("OTP Code"), p_output);
00432     m_qtPass->copyTextToClipboard(p_output);
00433   }
00434   if (QtPassSettings::isUseAutoclearPanel()) {
00435     clearPanelTimer.start();
00436   }
00437   setUiElementsEnabled(true);
00438 }
00439
00443 void MainWindow::clearPanel(bool notify) {
00444   while (ui->gridLayout->count() > 0) {
00445     QLayoutItem *item = ui->gridLayout->takeAt(0);
00446     delete item->widget();
00447     delete item;
00448   }
00449   if (notify) {
00450     QString output = "***" + tr("Password and Content hidden") + "***";
00451     ui->textBrowser->setHtml(output);
00452   } else {
```

```
00453     ui->textBrowser->setHtml("");
00454   }
00455 }
00456
00462 void MainWindow::setUiElementsEnabled(bool state) {
00463   ui->treeView->setEnabled(state);
00464   ui->lineEdit->setEnabled(state);
00465   ui->lineEdit->installEventFilter(this);
00466   ui->actionAddPassword->setEnabled(state);
00467   ui->actionAddFolder->setEnabled(state);
00468   ui->actionUsers->setEnabled(state);
00469   ui->actionConfig->setEnabled(state);
00470   // is a file selected?
00471   state &= ui->treeView->currentIndex().isValid();
00472   ui->actionDelete->setEnabled(state);
00473   ui->actionEdit->setEnabled(state);
00474   updateGitButtonVisibility();
00475   updateOtpButtonVisibility();
00476 }
00477
00478 void MainWindow::restoreWindow() {
00479   QByteArray geometry = QtPassSettings::getGeometry(saveGeometry());
00480   restoreGeometry(geometry);
00481   QByteArray savestate = QtPassSettings::getSavestate(saveState());
00482   restoreState(savestate);
00483   QPoint position = QtPassSettings::getPos(pos());
00484   move(position);
00485   QSize newSize = QtPassSettings::getSize(size());
00486   resize(newSize);
00487   if (QtPassSettings::isMaximized(isMaximized())) {
00488     showMaximized();
00489   }
00490
00491   if (QtPassSettings::isAlwaysOnTop()) {
00492     Qt::WindowFlags flags = windowFlags();
00493     setWindowFlags(flags | Qt::WindowStaysOnTopHint);
00494     show();
00495   }
00496
00497   if (QtPassSettings::isUseTrayIcon() && tray == nullptr) {
00498     initTrayIcon();
00499     if (QtPassSettings::isStartMinimized()) {
00500       // since we are still in constructor, can't directly hide
00501       QTimer::singleShot(10, this, SLOT(hide()));
00502     }
00503   } else if (!QtPassSettings::isUseTrayIcon() && tray != nullptr) {
00504     destroyTrayIcon();
00505   }
00506 }
00507
00511 void MainWindow::onConfig() { config(); }
00512
00518 void MainWindow::on_lineEdit_textChanged(const QString &arg1) {
00519   ui->statusBar->showMessage(tr("Looking for: %1").arg(arg1), 1000);
00520   ui->treeView->expandAll();
00521   clearPanel(false);
00522   ui->passwordName->setText("");
00523   ui->actionEdit->setEnabled(false);
00524   ui->actionDelete->setEnabled(false);
00525   searchTimer.start();
00526 }
00527
00532 void MainWindow::onTimeoutSearch() {
00533   QString query = ui->lineEdit->text();
00534
00535   if (query.isEmpty()) {
00536     ui->treeView->collapseAll();
00537     deselect();
00538   }
00539
00540   query.replace(QStringLiteral(" "), ".*");
00541   QRegularExpression regExp(query, QRegularExpression::CaseInsensitiveOption);
00542   proxyModel.setFilterRegularExpression(regExp);
00543   ui->treeView->setRootIndex(proxyModel.mapFromSource(
00544       model.setRootPath(QtPassSettings::getPassStore())));
00545
00546   if (proxyModel.rowCount() > 0 && !query.isEmpty()) {
00547     selectFirstFile();
00548   } else {
00549     ui->actionEdit->setEnabled(false);
00550     ui->actionDelete->setEnabled(false);
00551   }
00552 }
00553
00559 void MainWindow::on_lineEdit_returnPressed() {
00560 #ifdef QT_DEBUG
00561   dbg() « "on_lineEdit_returnPressed" « proxyModel.rowCount();
```

```
00562 #endif
00563
00564   if (proxyModel.rowCount() > 0) {
00565     selectFirstFile();
00566     on_treeView_clicked(ui->treeView->currentIndex());
00567   }
00568 }
00569
00574 void MainWindow::selectFirstFile() {
00575   QModelIndex index = proxyModel.mapFromSource(
00576       model.setRootPath(QtPassSettings::getPassStore()));
00577   index = firstFile(index);
00578   ui->treeView->setCurrentIndex(index);
00579 }
00580
00586 QModelIndex MainWindow::firstFile(QModelIndex parentIndex) {
00587   QModelIndex index = parentIndex;
00588   int numRows = proxyModel.rowCount(parentIndex);
00589   for (int row = 0; row < numRows; ++row) {
00590     index = proxyModel.index(row, 0, parentIndex);
00591     if (model.fileInfo(proxyModel.mapToSource(index)).isFile())
00592       return index;
00593     if (proxyModel.hasChildren(index))
00594       return firstFile(index);
00595   }
00596   return index;
00597 }
00598
00604 void MainWindow::setPassword(QString file, bool isNew) {
00605   PasswordDialog d(file, isNew, this);
00606
00607   if (!d.exec()) {
00608     ui->treeView->setFocus();
00609   }
00610 }
00611
00616 void MainWindow::addPassword() {
00617   bool ok;
00618   QString dir =
00619       Util::getDir(ui->treeView->currentIndex(), true, model, proxyModel);
00620   QString file =
00621       QInputDialog::getText(this, tr("New file"),
00622                             tr("New password file: \n(Will be placed in %1 )")
00623                                 .arg(QtPassSettings::getPassStore() +
00624                                      Util::getDir(ui->treeView->currentIndex(),
00625                                                   true, model, proxyModel)),
00626                             QLineEdit::Normal, "", &ok);
00627   if (!ok || file.isEmpty())
00628     return;
00629   file = dir + file;
00630   setPassword(file);
00631 }
00632
00637 void MainWindow::onDelete() {
00638   QModelIndex currentIndex = ui->treeView->currentIndex();
00639   if (!currentIndex.isValid()) {
00640     // This fixes https://github.com/IJHack/QtPass/issues/556
00641     // Otherwise the entire password directory would be deleted if
00642     // nothing is selected in the tree view.
00643     return;
00644   }
00645
00646   QFileInfo fileOrFolder =
00647       model.fileInfo(proxyModel.mapToSource(ui->treeView->currentIndex()));
00648   QString file = "";
00649   bool isDir = false;
00650
00651   if (fileOrFolder.isFile()) {
00652     file = getFile(ui->treeView->currentIndex(), true);
00653   } else {
00654     file = Util::getDir(ui->treeView->currentIndex(), true, model, proxyModel);
00655     isDir = true;
00656   }
00657
00658   QString dirMessage = tr(" and the whole content?");
00659   if (isDir) {
00660     QDirIterator it(model.rootPath() + QDir::separator() + file,
00661                     QDirIterator::Subdirectories);
00662     bool okDir = true;
00663     while (it.hasNext() && okDir) {
00664       it.next();
00665       if (QFileInfo(it.filePath()).isFile()) {
00666         if (QFileInfo(it.filePath()).suffix() != "gpg") {
00667           okDir = false;
00668           dirMessage = tr(" and the whole content? <br><strong>Attention: "
00669                           "there are unexpected files in the given folder, "
00670                           "check them before continue.</strong>");
```

```
00671         }
00672       }
00673     }
00674   }
00675
00676   if (QMessageBox::question(
00677           this, isDir ? tr("Delete folder?") : tr("Delete password?"),
00678           tr("Are you sure you want to delete %1%2?")
00679               .arg(QDir::separator() + file, isDir ? dirMessage : "?"),
00680           QMessageBox::Yes | QMessageBox::No) != QMessageBox::Yes)
00681     return;
00682
00683   QtPassSettings::getPass()->Remove(file, isDir);
00684 }
00685
00689 void MainWindow::onOtp() {
00690   QString file = getFile(ui->treeView->currentIndex(), true);
00691   if (!file.isEmpty()) {
00692     if (QtPassSettings::isUseOtp())
00693       QtPassSettings::getPass()->OtpGenerate(file);
00694   }
00695 }
00696
00700 void MainWindow::onEdit() {
00701   QString file = getFile(ui->treeView->currentIndex(), true);
00702   editPassword(file);
00703 }
00704
00709 void MainWindow::userDialog(QString dir) {
00710   if (!dir.isEmpty())
00711     currentDir = dir;
00712   onUsers();
00713 }
00714
00720 void MainWindow::onUsers() {
00721   QString dir =
00722       currentDir.isEmpty()
00723           ? Util::getDir(ui->treeView->currentIndex(), false, model, proxyModel)
00724           : currentDir;
00725
00726   UsersDialog d(dir, this);
00727   if (!d.exec()) {
00728     ui->treeView->setFocus();
00729   }
00730 }
00731
00736 void MainWindow::messageAvailable(QString message) {
00737   if (message.isEmpty()) {
00738     focusInput();
00739   } else {
00740     ui->treeView->expandAll();
00741     ui->lineEdit->setText(message);
00742     on_lineEdit_returnPressed();
00743   }
00744   show();
00745   raise();
00746 }
00747
00753 void MainWindow::generateKeyPair(QString batch, QDialog *keygenWindow) {
00754   keygen = keygenWindow;
00755   emit generateGPGKeyPair(batch);
00756 }
00757
00762 void MainWindow::updateProfileBox() {
00763   QHash<QString, QHash<QString, QString» profiles =
00764       QtPassSettings::getProfiles();
00765
00766   if (profiles.isEmpty()) {
00767     ui->profileWidget->hide();
00768   } else {
00769     ui->profileWidget->show();
00770     ui->profileBox->setEnabled(profiles.size() > 1);
00771     ui->profileBox->clear();
00772     QHashIterator<QString, QHash<QString, QString» i(profiles);
00773     while (i.hasNext()) {
00774       i.next();
00775       if (!i.key().isEmpty())
00776         ui->profileBox->addItem(i.key());
00777     }
00778     ui->profileBox->model()->sort(0);
00779   }
00780   int index = ui->profileBox->findText(QtPassSettings::getProfile());
00781   if (index != -1) // -1 for not found
00782     ui->profileBox->setCurrentIndex(index);
00783 }
00784
00790 void MainWindow::on_profileBox_currentIndexChanged(QString name) {
```

```
00791   if (m_qtPass->isFreshStart() || name == QtPassSettings::getProfile())
00792     return;
00793
00794   ui->lineEdit->clear();
00795
00796   QtPassSettings::setProfile(name);
00797
00798   QtPassSettings::setPassStore(
00799       QtPassSettings::getProfiles().value(name).value("path"));
00800   QtPassSettings::setPassSigningKey(
00801       QtPassSettings::getProfiles().value(name).value("signingKey"));
00802   ui->statusBar->showMessage(tr("Profile changed to %1").arg(name), 2000);
00803
00804   QtPassSettings::getPass()->updateEnv();
00805
00806   ui->treeView->selectionModel()->clear();
00807   ui->treeView->setRootIndex(proxyModel.mapFromSource(
00808       model.setRootPath(QtPassSettings::getPassStore())));
00809
00810   ui->actionEdit->setEnabled(false);
00811   ui->actionDelete->setEnabled(false);
00812 }
00813
00819 void MainWindow::initTrayIcon() {
00820   this->tray = new TrayIcon(this);
00821   // Setup tray icon
00822
00823   if (tray == nullptr) {
00824 #ifdef QT_DEBUG
00825     dbg() « "Allocating tray icon failed.";
00826 #endif
00827   }
00828
00829   if (!tray->getIsAllocated()) {
00830     destroyTrayIcon();
00831   }
00832 }
00833
00837 void MainWindow::destroyTrayIcon() {
00838   delete this->tray;
00839   tray = nullptr;
00840 }
00841
00846 void MainWindow::closeEvent(QCloseEvent *event) {
00847   if (QtPassSettings::isHideOnClose()) {
00848     this->hide();
00849     event->ignore();
00850   } else {
00851     m_qtPass->clearClipboard();
00852
00853     QtPassSettings::setGeometry(saveGeometry());
00854     QtPassSettings::setSavestate(saveState());
00855     QtPassSettings::setMaximized(isMaximized());
00856     if (!isMaximized()) {
00857       QtPassSettings::setPos(pos());
00858       QtPassSettings::setSize(size());
00859     }
00860     event->accept();
00861   }
00862 }
00863
00871 bool MainWindow::eventFilter(QObject *obj, QEvent *event) {
00872   if (obj == ui->lineEdit && event->type() == QEvent::KeyPress) {
00873     auto *key = dynamic_cast<QKeyEvent *>(event);
00874     if (key != NULL && key->key() == Qt::Key_Down) {
00875       ui->treeView->setFocus();
00876     }
00877   }
00878   return QObject::eventFilter(obj, event);
00879 }
00880
00885 void MainWindow::keyPressEvent(QKeyEvent *event) {
00886   switch (event->key()) {
00887   case Qt::Key_Delete:
00888     onDelete();
00889     break;
00890   case Qt::Key_Return:
00891   case Qt::Key_Enter:
00892     if (proxyModel.rowCount() > 0)
00893       on_treeView_clicked(ui->treeView->currentIndex());
00894     break;
00895   case Qt::Key_Escape:
00896     ui->lineEdit->clear();
00897     break;
00898   default:
00899     break;
00900   }
```

```
00901 }
00902
00908 void MainWindow::showContextMenu(const QPoint &pos) {
00909   QModelIndex index = ui->treeView->indexAt(pos);
00910   bool selected = true;
00911   if (!index.isValid()) {
00912     ui->treeView->clearSelection();
00913     ui->actionDelete->setEnabled(false);
00914     ui->actionEdit->setEnabled(false);
00915     currentDir = "";
00916     selected = false;
00917   }
00918
00919   ui->treeView->setCurrentIndex(index);
00920
00921   QPoint globalPos = ui->treeView->viewport()->mapToGlobal(pos);
00922
00923   QFileInfo fileOrFolder =
00924       model.fileInfo(proxyModel.mapToSource(ui->treeView->currentIndex()));
00925
00926   QMenu contextMenu;
00927   if (!selected || fileOrFolder.isDir()) {
00928     QAction *openFolder =
00929         contextMenu.addAction(tr("Open folder with file manager"));
00930     QAction *addFolder = contextMenu.addAction(tr("Add folder"));
00931     QAction *addPassword = contextMenu.addAction(tr("Add password"));
00932     QAction *users = contextMenu.addAction(tr("Users"));
00933     connect(openFolder, &QAction::triggered, this, &MainWindow::openFolder);
00934     connect(addFolder, &QAction::triggered, this, &MainWindow::addFolder);
00935     connect(addPassword, &QAction::triggered, this, &MainWindow::addPassword);
00936     connect(users, &QAction::triggered, this, &MainWindow::onUsers);
00937   } else if (fileOrFolder.isFile()) {
00938     QAction *edit = contextMenu.addAction(tr("Edit"));
00939     connect(edit, &QAction::triggered, this, &MainWindow::onEdit);
00940   }
00941   if (selected) {
00942     // if (useClipboard != CLIPBOARD_NEVER) {
00943     // contextMenu.addSeparator();
00944     // QAction* copyItem = contextMenu.addAction(tr("Copy Password"));
00945     // if (getClippedPassword().length() == 0) copyItem->setEnabled(false);
00946     // connect(copyItem, SIGNAL(triggered()), this,
00947     // SLOT(copyPasswordToClipboard()));
00948     // }
00949     contextMenu.addSeparator();
00950     if (fileOrFolder.isDir()) {
00951       QAction *renameFolder = contextMenu.addAction(tr("Rename folder"));
00952       connect(renameFolder, &QAction::triggered, this,
00953               &MainWindow::renameFolder);
00954     } else if (fileOrFolder.isFile()) {
00955       QAction *renamePassword = contextMenu.addAction(tr("Rename password"));
00956       connect(renamePassword, &QAction::triggered, this,
00957               &MainWindow::renamePassword);
00958     }
00959     QAction *deleteItem = contextMenu.addAction(tr("Delete"));
00960     connect(deleteItem, &QAction::triggered, this, &MainWindow::onDelete);
00961   }
00962   contextMenu.exec(globalPos);
00963 }
00964
00970 void MainWindow::showBrowserContextMenu(const QPoint &pos) {
00971   QMenu *contextMenu = ui->textBrowser->createStandardContextMenu(pos);
00972   QPoint globalPos = ui->textBrowser->viewport()->mapToGlobal(pos);
00973
00974   contextMenu->exec(globalPos);
00975   delete contextMenu;
00976 }
00977
00981 void MainWindow::openFolder() {
00982   QString dir =
00983       Util::getDir(ui->treeView->currentIndex(), false, model, proxyModel);
00984
00985   QString path = QDir::toNativeSeparators(dir);
00986   QDesktopServices::openUrl(QUrl::fromLocalFile(path));
00987 }
00988
00992 void MainWindow::addFolder() {
00993   bool ok;
00994   QString dir =
00995       Util::getDir(ui->treeView->currentIndex(), false, model, proxyModel);
00996   QString newdir =
00997       QInputDialog::getText(this, tr("New file"),
00998                             tr("New Folder: \n(Will be placed in %1 )")
00999                                 .arg(QtPassSettings::getPassStore() +
01000                                      Util::getDir(ui->treeView->currentIndex(),
01001                                                   true, model, proxyModel)),
01002                             QLineEdit::Normal, "", &ok);
01003   if (!ok || newdir.isEmpty())
```

```
01004     return;
01005   newdir.prepend(dir);
01006   // dbg()« newdir;
01007   QDir().mkdir(newdir);
01008 }
01009
01013 void MainWindow::renameFolder() {
01014   bool ok;
01015   QString srcDir = QDir::cleanPath(
01016       Util::getDir(ui->treeView->currentIndex(), false, model, proxyModel));
01017   QString srcDirName = QDir(srcDir).dirName();
01018   QString newName =
01019       QInputDialog::getText(this, tr("Rename file"), tr("Rename Folder To: "),
01020                             QLineEdit::Normal, srcDirName, &ok);
01021   if (!ok || newName.isEmpty())
01022     return;
01023   QString destDir = srcDir;
01024   destDir.replace(srcDir.lastIndexOf(srcDirName), srcDirName.length(), newName);
01025   QtPassSettings::getPass()->Move(srcDir, destDir);
01026 }
01027
01032 void MainWindow::editPassword(const QString &file) {
01033   if (!file.isEmpty()) {
01034     if (QtPassSettings::isUseGit() && QtPassSettings::isAutoPull())
01035       onUpdate(true);
01036     setPassword(file, false);
01037   }
01038 }
01039
01043 void MainWindow::renamePassword() {
01044   bool ok;
01045   QString file = getFile(ui->treeView->currentIndex(), false);
01046   QString filePath = QFileInfo(file).path();
01047   QString fileName = QFileInfo(file).fileName();
01048   if (fileName.endsWith(".gpg", Qt::CaseInsensitive))
01049     fileName.chop(4);
01050
01051   QString newName =
01052       QInputDialog::getText(this, tr("Rename file"), tr("Rename File To: "),
01053                             QLineEdit::Normal, fileName, &ok);
01054   if (!ok || newName.isEmpty())
01055     return;
01056   QString newFile = QDir(filePath).filePath(newName);
01057   QtPassSettings::getPass()->Move(file, newFile);
01058 }
01059
01064 void MainWindow::clearTemplateWidgets() {
01065   while (ui->gridLayout->count() > 0) {
01066     QLayoutItem *item = ui->gridLayout->takeAt(0);
01067     delete item->widget();
01068     delete item;
01069   }
01070   ui->verticalLayoutPassword->setSpacing(0);
01071 }
01072
01073 void MainWindow::copyPasswordFromTreeview() {
01074   QFileInfo fileOrFolder =
01075       model.fileInfo(proxyModel.mapToSource(ui->treeView->currentIndex()));
01076
01077   if (fileOrFolder.isFile()) {
01078     QString file = getFile(ui->treeView->currentIndex(), true);
01079     connect(QtPassSettings::getPass(), &Pass::finishedShow, this,
01080             &MainWindow::passwordFromFileToClipboard);
01081     QtPassSettings::getPass()->Show(file);
01082   }
01083 }
01084
01085 void MainWindow::passwordFromFileToClipboard(const QString &text) {
01086   QStringList tokens = text.split('\n');
01087   m_qtPass->copyTextToClipboard(tokens[0]);
01088 }
01089
01096 void MainWindow::addToGridLayout(int position, const QString &field,
01097                                  const QString &value) {
01098   QString trimmedField = field.trimmed();
01099   QString trimmedValue = value.trimmed();
01100
01101   // Combine the Copy button and the line edit in one widget
01102   QFrame *frame = new QFrame();
01103   QLayout *ly = new QHBoxLayout();
01104   ly->setContentsMargins(5, 2, 2, 2);
01105   ly->setSpacing(0);
01106   frame->setLayout(ly);
01107   if (QtPassSettings::getClipBoardType() != Enums::CLIPBOARD_NEVER) {
01108     auto *fieldLabel = new QPushButtonWithClipboard(trimmedValue, this);
01109     connect(fieldLabel, &QPushButtonWithClipboard::clicked, m_qtPass,
01110             &QtPass::copyTextToClipboard);
```

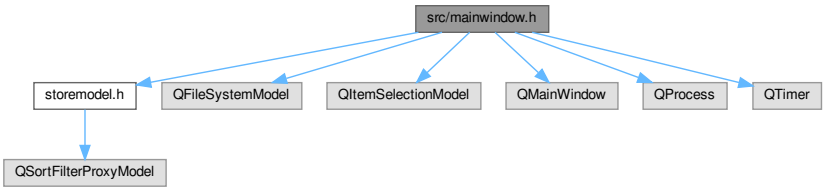
```
01111
01112     fieldLabel->setStyleSheet(
01113         "border-style: none ; background: transparent; padding: 0; margin: 0;");
01114     frame->layout()->addWidget(fieldLabel);
01115   }
01116
01117   if (QtPassSettings::isUseQrencode()) {
01118     QPushButtonAsQRCode *qrbutton = new QPushButtonAsQRCode(trimmedValue, this);
01119     connect(qrbutton, &QPushButtonAsQRCode::clicked, m_qtPass,
01120             &QtPass::showTextAsQRCode);
01121     qrbutton->setStyleSheet(
01122         "border-style: none ; background: transparent; padding: 0; margin: 0;");
01123     frame->layout()->addWidget(qrbutton);
01124   }
01125
01126   // set the echo mode to password, if the field is "password"
01127   const QString lineStyle =
01128       QtPassSettings::isUseMonospace()
01129           ? "border-style: none; background: transparent; font-family: "
01130             "monospace;"
01131           : "border-style: none; background: transparent;";
01132
01133   if (QtPassSettings::isHidePassword() && trimmedField == tr("Password")) {
01134
01135     auto *line = new QLineEdit();
01136     line->setObjectName(trimmedField);
01137     line->setText(trimmedValue);
01138     line->setReadOnly(true);
01139     line->setStyleSheet(lineStyle);
01140     line->setContentsMargins(0, 0, 0, 0);
01141     line->setEchoMode(QLineEdit::Password);
01142     QPushButtonShowPassword *showButton =
01143         new QPushButtonShowPassword(line, this);
01144     showButton->setStyleSheet(
01145         "border-style: none ; background: transparent; padding: 0; margin: 0;");
01146     showButton->setContentsMargins(0, 0, 0, 0);
01147     frame->layout()->addWidget(showButton);
01148     frame->layout()->addWidget(line);
01149   } else {
01150     auto *line = new QTextBrowser();
01151     line->setOpenExternalLinks(true);
01152     line->setOpenLinks(true);
01153     line->setMaximumHeight(26);
01154     line->setMinimumHeight(26);
01155     line->setSizePolicy(
01156         QSizePolicy(QSizePolicy::Expanding, QSizePolicy::Minimum));
01157     line->setObjectName(trimmedField);
01158     trimmedValue.replace(Util::protocolRegex(), R"(<a href="\1">\1</a>)");
01159     line->setText(trimmedValue);
01160     line->setReadOnly(true);
01161     line->setStyleSheet(lineStyle);
01162     line->setContentsMargins(0, 0, 0, 0);
01163     frame->layout()->addWidget(line);
01164   }
01165
01166   frame->setStyleSheet(
01167       ".QFrame{border: 1px solid lightgrey; border-radius: 5px;}");
01168
01169   // set into the layout
01170   ui->gridLayout->addWidget(new QLabel(trimmedField), position, 0);
01171   ui->gridLayout->addWidget(frame, position, 1);
01172 }
01173
01180 void MainWindow::showStatusMessage(QString msg, int timeout) {
01181   ui->statusBar->showMessage(msg, timeout);
01182 }
01183
01187 void MainWindow::startReencryptPath() {
01188   setUiElementsEnabled(false);
01189   ui->treeView->setDisabled(true);
01190 }
01191
01195 void MainWindow::endReencryptPath() { setUiElementsEnabled(true); }
01196
01197 void MainWindow::updateGitButtonVisibility() {
01198   if (!QtPassSettings::isUseGit() ||
01199       (QtPassSettings::getGitExecutable().isEmpty() &&
01200        QtPassSettings::getPassExecutable().isEmpty())) {
01201     enableGitButtons(false);
01202   } else {
01203     enableGitButtons(true);
01204   }
01205 }
01206
01207 void MainWindow::updateOtpButtonVisibility() {
01208 #if defined(Q_OS_WIN) || defined(__APPLE__)
01209   ui->actionOtp->setVisible(false);
```

```
01210 #endif
01211   if (!QtPassSettings::isUseOtp())
01212     ui->actionOtp->setEnabled(false);
01213   else
01214     ui->actionOtp->setEnabled(true);
01215 }
01216
01217 void MainWindow::enableGitButtons(const bool &state) {
01218   // Following GNOME guidelines is preferable disable buttons instead of hide
01219   ui->actionPush->setEnabled(state);
01220   ui->actionUpdate->setEnabled(state);
01221 }
01222
01228 void MainWindow::critical(QString title, QString msg) {
01229   QMessageBox::critical(this, title, msg);
01230 }
```

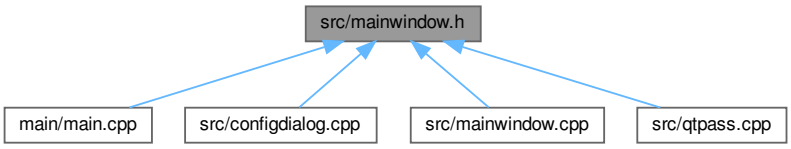
14.36 src/mainwindow.h File Reference

```
#include "storemodel.h"
#include <QFileSystemModel>
#include <QItemSelectionModel>
#include <QMainWindow>
#include <QProcess>
#include <QTimer>
```

Include dependency graph for mainwindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class MainWindow

  The MainWindow class does way too much, not only is it a switchboard, configuration handler and more, it's also the process-manager.

Namespaces

- namespace Ui

Macros

- #define SingleApplication QApplication

14.36.1 Macro Definition Documentation

14.36.1.1 SingleApplication

```
#define SingleApplication QApplication
```

Definition at line 15 of file mainwindow.h.

14.37 mainwindow.h

Go to the documentation of this file.

```
00001 #ifndef MAINWINDOW_H_
00002 #define MAINWINDOW_H_
00003
00004 #include "storemodel.h"
00005
00006 #include <QFileSystemModel>
00007 #include <QItemSelectionModel>
00008 #include <QMainWindow>
00009 #include <QProcess>
00010 #include <QTimer>
00011
00012 #if SINGLE_APP
00013 class SingleApplication;
00014 #else
00015 #define SingleApplication QApplication
00016 #endif
00017
00018 #ifdef __APPLE__
00019 // http://doc.qt.io/qt-5/qkeysequence.html#qt_set_sequence_auto_mnemonic
00020 void qt_set_sequence_auto_mnemonic(bool b);
00021 #endif
00022
00023 namespace Ui {
00024 class MainWindow;
00025 }
00026
00034 class QDialog;
00035 class QtPass;
00036 class TrayIcon;
00037 class MainWindow : public QMainWindow {
00038   Q_OBJECT
00039
00040 public:
00041   explicit MainWindow(const QString &searchText = QString(),
00042                       QWidget *parent = nullptr);
00043   ~MainWindow();
00044
00045   void restoreWindow();
00046   void generateKeyPair(QString, QDialog *);
00047   void userDialog(QString = "");
00048   void config();
00049
00050   void setUiElementsEnabled(bool state);
00051   void flashText(const QString &text, const bool isError,
00052                  const bool isHtml = false);
```

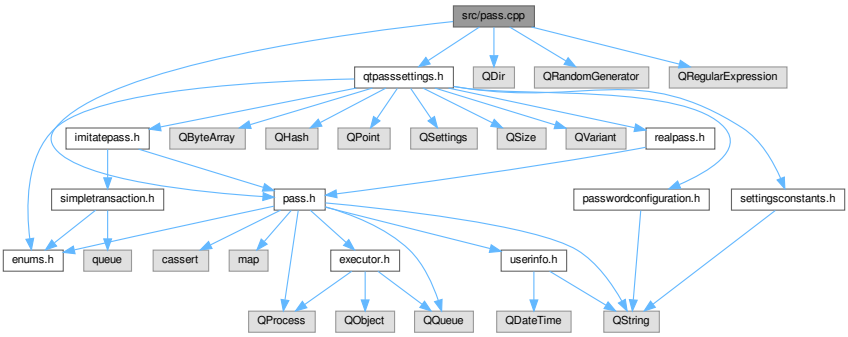
```
00053
00054   const QModelIndex getCurrentTreeViewIndex();
00055
00056   QDialog *getKeygenDialog() { return this->keygen; }
00057   void cleanKeygenDialog();
00058
00059 protected:
00060   void closeEvent(QCloseEvent *event);
00061   void keyPressEvent(QKeyEvent *event);
00062   void changeEvent(QEvent *event);
00063   bool eventFilter(QObject *obj, QEvent *event);
00064
00065 signals:
00066   void passShowHandlerFinished(QString output);
00067   void passGitInitNeeded();
00068   void generateGPGKeyPair(QString batch);
00069
00070 public slots:
00071   void deselect();
00072
00073   void messageAvailable(QString message);
00074   void critical(QString, QString);
00075
00076   void executeWrapperStarted();
00077   void showStatusMessage(QString msg, int timeout = 2000);
00078   void passShowHandler(const QString &);
00079   void passOtpHandler(const QString &);
00080
00081   void onPush();
00082   void on_treeView_clicked(const QModelIndex &index);
00083
00084   void startReencryptPath();
00085   void endReencryptPath();
00086
00087 private slots:
00088   void addPassword();
00089   void addFolder();
00090   void onEdit();
00091   void onDelete();
00092   void onOtp();
00093   void onUpdate(bool block = false);
00094   void onUsers();
00095   void onConfig();
00096   void on_treeView_doubleClicked(const QModelIndex &index);
00097   void clearPanel(bool notify = true);
00098   void on_lineEdit_textChanged(const QString &arg1);
00099   void on_lineEdit_returnPressed();
00100   void on_profileBox_currentIndexChanged(QString);
00101   void showContextMenu(const QPoint &pos);
00102   void showBrowserContextMenu(const QPoint &pos);
00103   void openFolder();
00104   void renameFolder();
00105   void editPassword(const QString &);
00106   void renamePassword();
00107   void focusInput();
00108   void copyPasswordFromTreeview();
00109   void passwordFromFileToClipboard(const QString &text);
00110   void onTimeoutSearch();
00111
00112 private:
00113   QtPass *m_qtPass;
00114   QScopedPointer<Ui::MainWindow> ui;
00115   QFileSystemModel model;
00116   StoreModel proxyModel;
00117   QScopedPointer<QItemSelectionModel> selectionModel;
00118   QTimer clearPanelTimer, searchTimer;
00119   QDialog *keygen;
00120   QString currentDir;
00121   TrayIcon *tray;
00122
00123   void initToolBarButtons();
00124   void initStatusBar();
00125
00126   void updateText();
00127   void selectFirstFile();
00128   QModelIndex firstFile(QModelIndex parentIndex);
00129   QString getFile(const QModelIndex &, bool);
00130   void setPassword(QString, bool isNew = true);
00131
00132   void updateProfileBox();
00133   void initTrayIcon();
00134   void destroyTrayIcon();
00135   void clearTemplateWidgets();
00136   void reencryptPath(QString dir);
00137   void addToGridLayout(int position, const QString &field,
00138                        const QString &value);
00139
```

```
00140   void updateGitButtonVisibility();
00141   void updateOtpButtonVisibility();
00142   void enableGitButtons(const bool &);
00143 };
00144
00145 #endif // MAINWINDOW_H_
```

14.38 src/pass.cpp File Reference

```
#include "pass.h"
#include "qtpasssettings.h"
#include <QDir>
#include <QRandomGenerator>
#include <QRegularExpression>
```

Include dependency graph for pass.cpp:



14.39 pass.cpp

Go to the documentation of this file.

```
00001 #include "pass.h"
00002 #include "qtpasssettings.h"
00003 #include <QDir>
00004 #include <QRandomGenerator>
00005 #include <QRegularExpression>
00006
00007 #ifdef QT_DEBUG
00008 #include "debughelper.h"
00009 #endif
00010
00011 using namespace std;
00012 using namespace Enums;
00013
00017 Pass::Pass() : wrapperRunning(false), env(QProcess::systemEnvironment()) {
00018   connect(&exec,
00019           static_cast<void (Executor::*)(int, int, const QString &,
00020                                          const QString &)>(&Executor::finished),
00021           this, &Pass::finished);
00022
00023   // TODO(bezet): stop using process
00024   // connect(&process, SIGNAL(error(QProcess::ProcessError)), this,
00025   //         SIGNAL(error(QProcess::ProcessError)));
00026
00027   connect(&exec, &Executor::starting, this, &Pass::startingExecuteWrapper);
00028   env.append("WSLENV=PASSWORD_STORE_DIR/p");
00029 }
00030
00031 void Pass::executeWrapper(PROCESS id, const QString &app,
00032                           const QStringList &args, bool readStdout,
00033                           bool readStderr) {
00034   executeWrapper(id, app, args, QString(), readStdout, readStderr);
```

```
00035 }
00036
00037 void Pass::executeWrapper(PROCESS id, const QString &app,
00038                           const QStringList &args, QString input,
00039                           bool readStdout, bool readStderr) {
00040 #ifdef QT_DEBUG
00041   dbg() « app « args;
00042 #endif
00043   exec.execute(id, QtPassSettings::getPassStore(), app, args, input, readStdout,
00044                readStderr);
00045 }
00046
00047 void Pass::init() {
00048 #ifdef __APPLE__
00049   // If it exists, add the gpgtools to PATH
00050   if (QFile("/usr/local/MacGPG2/bin").exists())
00051     env.replaceInStrings("PATH=", "PATH=/usr/local/MacGPG2/bin:");
00052   // Add missing /usr/local/bin
00053   if (env.filter("/usr/local/bin").isEmpty())
00054     env.replaceInStrings("PATH=", "PATH=/usr/local/bin:");
00055 #endif
00056
00057   if (!QtPassSettings::getGpgHome().isEmpty()) {
00058     QDir absHome(QtPassSettings::getGpgHome());
00059     absHome.makeAbsolute();
00060     env « "GNUPGHOME=" + absHome.path();
00061   }
00062 }
00063
00071 QString Pass::Generate_b(unsigned int length, const QString &charset) {
00072   QString passwd;
00073   if (QtPassSettings::isUsePwgen()) {
00074     // --secure goes first as it overrides --no-* otherwise
00075     QStringList args;
00076     args.append("-1");
00077     if (!QtPassSettings::isLessRandom())
00078       args.append("--secure");
00079     args.append(QtPassSettings::isAvoidCapitals() ? "--no-capitalize"
00080                                                   : "--capitalize");
00081     args.append(QtPassSettings::isAvoidNumbers() ? "--no-numerals"
00082                                                  : "--numerals");
00083     if (QtPassSettings::isUseSymbols())
00084       args.append("--symbols");
00085     args.append(QString::number(length));
00086     //  TODO(bezet): try-catch here(2 statuses to merge o_O)
00087     if (exec.executeBlocking(QtPassSettings::getPwgenExecutable(), args,
00088                              &passwd) == 0) {
00089       static const QRegularExpression literalNewLines{"[\\n\\r]"};
00090       passwd.remove(literalNewLines);
00091     } else {
00092       passwd.clear();
00093 #ifdef QT_DEBUG
00094       qDebug() « __FILE__ « ":" « __LINE__ « "\t"
00095                « "pwgen fail";
00096 #endif
00097       //    TODO(bezet): emit critical ?
00098     }
00099   } else {
00100     if (charset.length() > 0) {
00101       passwd = generateRandomPassword(charset, length);
00102     } else {
00103       emit critical(
00104           tr("No characters chosen"),
00105           tr("Can't generate password, there are no characters to choose from "
00106              "set in the configuration!"));
00107     }
00108   }
00109   return passwd;
00110 }
00111
00116 void Pass::GenerateGPGKeys(QString batch) {
00117   executeWrapper(GPG_GENKEYS, QtPassSettings::getGpgExecutable(),
00118                  {"--gen-key", "--no-tty", "--batch"}, batch);
00119   // TODO check status / error messages - probably not here, it's just started
00120   // here, see finished for details
00121   // https://github.com/IJHack/QtPass/issues/202#issuecomment-251081688
00122 }
00123
00130 QList<UserInfo> Pass::listKeys(QStringList keystrings, bool secret) {
00131   QList<UserInfo> users;
00132   QStringList args = {"--no-tty", "--with-colons", "--with-fingerprint"};
00133   args.append(secret ? "--list-secret-keys" : "--list-keys");
00134
00135   for (const QString &keystring : qAsConst(keystrings)) {
00136     if (!keystring.isEmpty()) {
00137       args.append(keystring);
00138     }
```

```
00139   }
00140   QString p_out;
00141   if (exec.executeBlocking(QtPassSettings::getGpgExecutable(), args, &p_out) !=
00142       0)
00143     return users;
00144   static const QRegularExpression newLines{"[\r\n]"};
00145 #if QT_VERSION >= QT_VERSION_CHECK(5, 15, 0)
00146   const QStringList keys = p_out.split(newLines, Qt::SkipEmptyParts);
00147 #else
00148   const QStringList keys = p_out.split(newLines, QString::SkipEmptyParts);
00149 #endif
00150   UserInfo current_user;
00151   for (const QString &key : keys) {
00152     QStringList props = key.split(':');
00153     if (props.size() < 10)
00154       continue;
00155     if (props[0] == (secret ? "sec" : "pub")) {
00156       if (!current_user.key_id.isEmpty())
00157         users.append(current_user);
00158       current_user = UserInfo();
00159       current_user.key_id = props[4];
00160       current_user.name = props[9].toUtf8();
00161       current_user.validity = props[1][0].toLatin1();
00162       current_user.created.setSecsSinceEpoch(props[5].toUInt());
00163       current_user.expiry.setSecsSinceEpoch(props[6].toUInt());
00164     } else if (current_user.name.isEmpty() && props[0] == "uid") {
00165       current_user.name = props[9];
00166     } else if ((props[0] == "fpr") && props[9].endsWith(current_user.key_id)) {
00167       current_user.key_id = props[9];
00168     }
00169   }
00170   if (!current_user.key_id.isEmpty())
00171     users.append(current_user);
00172   return users;
00173 }
00174
00181 QList<UserInfo> Pass::listKeys(QString keystring, bool secret) {
00182   return listKeys(QStringList(keystring), secret);
00183 }
00184
00195 void Pass::finished(int id, int exitCode, const QString &out,
00196                     const QString &err) {
00197   auto pid = static_cast<PROCESS>(id);
00198   if (exitCode != 0) {
00199     emit processErrorExit(exitCode, err);
00200     return;
00201   }
00202   switch (pid) {
00203   case GIT_INIT:
00204     emit finishedGitInit(out, err);
00205     break;
00206   case GIT_PULL:
00207     emit finishedGitPull(out, err);
00208     break;
00209   case GIT_PUSH:
00210     emit finishedGitPush(out, err);
00211     break;
00212   case PASS_SHOW:
00213     emit finishedShow(out);
00214     break;
00215   case PASS_OTP_GENERATE:
00216     emit finishedOtpGenerate(out);
00217     break;
00218   case PASS_INSERT:
00219     emit finishedInsert(out, err);
00220     break;
00221   case PASS_REMOVE:
00222     emit finishedRemove(out, err);
00223     break;
00224   case PASS_INIT:
00225     emit finishedInit(out, err);
00226     break;
00227   case PASS_MOVE:
00228     emit finishedMove(out, err);
00229     break;
00230   case PASS_COPY:
00231     emit finishedCopy(out, err);
00232     break;
00233   case GPG_GENKEYS:
00234     emit finishedGenerateGPGKeys(out, err);
00235     break;
00236   default:
00237 #ifdef QT_DEBUG
00238     dbg() « "Unhandled process type" « pid;
00239 #endif
00240     break;
00241   }
```

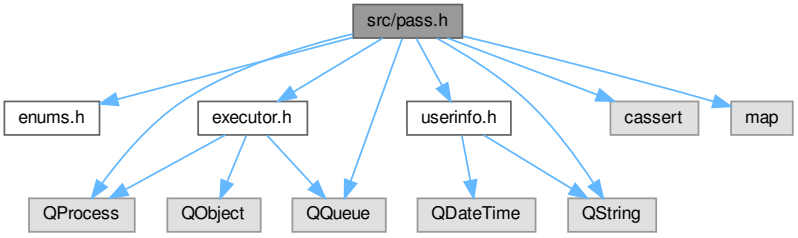
```
00242 }
00243
00248 void Pass::updateEnv() {
00249   // put PASSWORD_STORE_SIGNING_KEY in env
00250   QStringList envSigningKey = env.filter("PASSWORD_STORE_SIGNING_KEY=");
00251   QString currentSigningKey = QtPassSettings::getPassSigningKey();
00252   if (envSigningKey.isEmpty()) {
00253     if (!currentSigningKey.isEmpty()) {
00254       // dbg()« "Added
00255       // PASSWORD_STORE_SIGNING_KEY with" + currentSigningKey;
00256       env.append("PASSWORD_STORE_SIGNING_KEY=" + currentSigningKey);
00257     }
00258   } else {
00259     if (currentSigningKey.isEmpty()) {
00260       // dbg() « "Removed
00261       // PASSWORD_STORE_SIGNING_KEY";
00262       env.removeAll(envSigningKey.first());
00263     } else {
00264       // dbg()« "Update
00265       // PASSWORD_STORE_SIGNING_KEY with " + currentSigningKey;
00266       env.replaceInStrings(envSigningKey.first(),
00267                            "PASSWORD_STORE_SIGNING_KEY=" + currentSigningKey);
00268     }
00269   }
00270   // put PASSWORD_STORE_DIR in env
00271   QStringList store = env.filter("PASSWORD_STORE_DIR=");
00272   if (store.isEmpty()) {
00273     // dbg()« "Added
00274     // PASSWORD_STORE_DIR";
00275     env.append("PASSWORD_STORE_DIR=" + QtPassSettings::getPassStore());
00276   } else {
00277     // dbg()« "Update
00278     // PASSWORD_STORE_DIR with " + passStore;
00279     env.replaceInStrings(store.first(), "PASSWORD_STORE_DIR=" +
00280                                           QtPassSettings::getPassStore());
00281   }
00282   exec.setEnvironment(env);
00283 }
00284
00290 QString Pass::getGpgIdPath(QString for_file) {
00291   QString passStore =
00292       QDir::fromNativeSeparators(QtPassSettings::getPassStore());
00293   QDir gpgIdDir(
00294       QFileInfo(QDir::fromNativeSeparators(for_file).startsWith(passStore)
00295                     ? for_file
00296                     : QtPassSettings::getPassStore() + for_file)
00297           .absoluteDir());
00298   bool found = false;
00299   while (gpgIdDir.exists() && gpgIdDir.absolutePath().startsWith(passStore)) {
00300     if (QFile(gpgIdDir.absoluteFilePath(".gpg-id")).exists()) {
00301       found = true;
00302       break;
00303     }
00304     if (!gpgIdDir.cdUp())
00305       break;
00306   }
00307   QString gpgIdPath(found ? gpgIdDir.absoluteFilePath(".gpg-id")
00308                           : QtPassSettings::getPassStore() + ".gpg-id");
00309
00310   return gpgIdPath;
00311 }
00312
00318 QStringList Pass::getRecipientList(QString for_file) {
00319   QFile gpgId(getGpgIdPath(for_file));
00320   if (!gpgId.open(QIODevice::ReadOnly | QIODevice::Text))
00321     return QStringList();
00322   QStringList recipients;
00323   while (!gpgId.atEnd()) {
00324     QString recipient(gpgId.readLine());
00325     recipient = recipient.trimmed();
00326     if (!recipient.isEmpty())
00327       recipients += recipient;
00328   }
00329   return recipients;
00330 }
00331
00339 QStringList Pass::getRecipientString(QString for_file, QString separator,
00340                                      int *count) {
00341   Q_UNUSED(separator)
00342   Q_UNUSED(count)
00343   return Pass::getRecipientList(for_file);
00344 }
00345
00346 /* Copyright (C) 2017 Jason A. Donenfeld <Jason@zx2c4.com>. All Rights Reserved.
00347  */
00348
00349 quint32 Pass::boundedRandom(quint32 bound) {
```

```
00350   if (bound < 2) {
00351     return 0;
00352   }
00353
00354   quint32 randval;
00355   const quint32 max_mod_bound = (1 + ~bound) % bound;
00356
00357   do {
00358     randval = QRandomGenerator::system()->generate();
00359   } while (randval < max_mod_bound);
00360
00361   return randval % bound;
00362 }
00363
00364 QString Pass::generateRandomPassword(const QString &charset,
00365                                      unsigned int length) {
00366   QString out;
00367   for (unsigned int i = 0; i < length; ++i) {
00368     out.append(charset.at(static_cast<int>(
00369         boundedRandom(static_cast<quint32>(charset.length())))));
00370   }
00371   return out;
00372 }
```

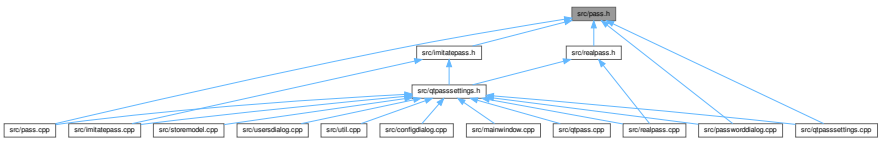
14.40 src/pass.h File Reference

```
#include "enums.h"
#include "executor.h"
#include "userinfo.h"
#include <QProcess>
#include <QQueue>
#include <QString>
#include <cassert>
#include <map>
```

Include dependency graph for pass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Pass

  Acts as an abstraction for pass or pass imitation.

14.41 pass.h

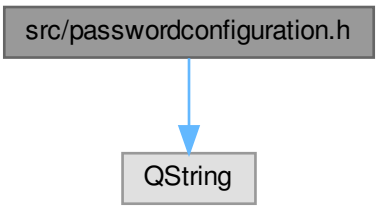
Go to the documentation of this file.

```
00001 #ifndef PASS_H
00002 #define PASS_H
00003
00004 #include "enums.h"
00005 #include "executor.h"
00006 #include "userinfo.h"
00007
00008 #include <QProcess>
00009 #include <QQueue>
00010 #include <QString>
00011 #include <cassert>
00012 #include <map>
00013
00018 class Pass : public QObject {
00019   Q_OBJECT
00020
00021   bool wrapperRunning;
00022   QStringList env;
00023
00024 protected:
00025   Executor exec;
00026
00027   typedef Enums::PROCESS PROCESS;
00028
00029 public:
00030   Pass();
00031   void init();
00032
00033   virtual ~Pass() {}
00034   virtual void GitInit() = 0;
00035   virtual void GitPull() = 0;
00036   virtual void GitPull_b() = 0;
00037   virtual void GitPush() = 0;
00038   virtual void Show(QString file) = 0;
00039   virtual void OtpGenerate(QString file) = 0;
00040   virtual void Insert(QString file, QString value, bool force) = 0;
00041   virtual void Remove(QString file, bool isDir) = 0;
00042   virtual void Move(const QString srcDir, const QString dest,
00043                     const bool force = false) = 0;
00044   virtual void Copy(const QString srcDir, const QString dest,
00045                     const bool force = false) = 0;
00046   virtual void Init(QString path, const QList<UserInfo> &users) = 0;
00047   virtual QString Generate_b(unsigned int length, const QString &charset);
00048
00049   void GenerateGPGKeys(QString batch);
00050   QList<UserInfo> listKeys(QStringList keystrings, bool secret = false);
00051   QList<UserInfo> listKeys(QString keystring = "", bool secret = false);
00052   void updateEnv();
00053   static QString getGpgIdPath(QString for_file);
00054   static QStringList getRecipientList(QString for_file);
00055   //  TODO(bezet): getRecipientString is useless, refactor
00056   static QStringList getRecipientString(QString for_file,
00057                                         QString separator = " ",
00058                                         int *count = NULL);
00059
00060 protected:
00061   void executeWrapper(PROCESS id, const QString &app, const QStringList &args,
00062                       bool readStdout = true, bool readStderr = true);
00063   QString generateRandomPassword(const QString &charset, unsigned int length);
00064   quint32 boundedRandom(quint32 bound);
00065
00066   virtual void executeWrapper(PROCESS id, const QString &app,
00067                               const QStringList &args, QString input,
00068                               bool readStdout = true, bool readStderr = true);
00069
00070 protected slots:
00071   virtual void finished(int id, int exitCode, const QString &out,
00072                         const QString &err);
00073
00074 signals:
00075   void error(QProcess::ProcessError);
00076   void startingExecuteWrapper();
00077   void statusMsg(QString, int);
00078   void critical(QString, QString);
00079
00080   void processErrorExit(int exitCode, const QString &err);
00081
00082   void finishedAny(const QString &, const QString &);
00083   void finishedGitInit(const QString &, const QString &);
00084   void finishedGitPull(const QString &, const QString &);
00085   void finishedGitPush(const QString &, const QString &);
00086   void finishedShow(const QString &);
```

```
00087   void finishedOtpGenerate(const QString &);
00088   void finishedInsert(const QString &, const QString &);
00089   void finishedRemove(const QString &, const QString &);
00090   void finishedInit(const QString &, const QString &);
00091   void finishedMove(const QString &, const QString &);
00092   void finishedCopy(const QString &, const QString &);
00093   void finishedGenerate(const QString &, const QString &);
00094   void finishedGenerateGPGKeys(const QString &, const QString &);
00095 };
00096
00097 #endif // PASS_H
```

14.42 src/passwordconfiguration.h File Reference

`#include <QString>`
Include dependency graph for passwordconfiguration.h:



This graph shows which files directly or indirectly include this file:

Classes

- struct PasswordConfiguration

  Holds the Password configuration settings.

14.43 passwordconfiguration.h

Go to the documentation of this file.

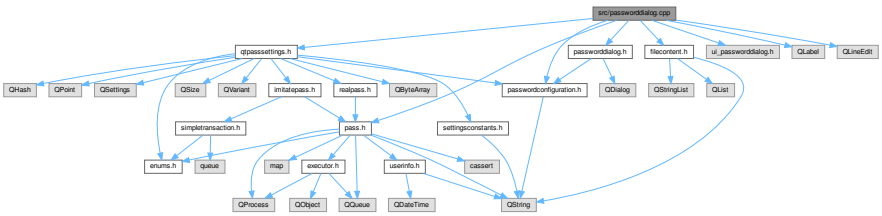
```
00001 #ifndef PASSWORDCONFIGURATION_H
00002 #define PASSWORDCONFIGURATION_H
00003
00004 #include <QString>
00005
00010 struct PasswordConfiguration {
00014   enum characterSet {
00015     ALLCHARS = 0,
00016     ALPHABETICAL,
```

```
00017     ALPHANUMERIC,
00018     CUSTOM,
00019     CHARSETS_COUNT //  have to be last, for easier initialization of arrays
00020   } selected;
00024   int length;
00028   QString Characters[CHARSETS_COUNT];
00029   PasswordConfiguration() : selected(ALLCHARS), length(16) {
00030     Characters[ALLCHARS] =
00031         "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890~!@#$%^&"
00032         "*()_-+={}[]|:;<>,.?"; /*AllChars*/
00033     Characters[ALPHABETICAL] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstu"
00034                                "vwxyz"; /*Only Alphabetical*/
00035     Characters[ALPHANUMERIC] = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstu"
00036                                "vwxyz1234567890"; /*Alphabetical and Numerical*/
00037     Characters[CUSTOM] = Characters[ALLCHARS]; //  this may be redefined by user
00038   }
00039 };
00040
00041 #endif // PASSWORDCONFIGURATION_H
```

14.44 src/passworddialog.cpp File Reference

```
#include "passworddialog.h"
#include "filecontent.h"
#include "pass.h"
#include "passwordconfiguration.h"
#include "qtpasssettings.h"
#include "ui_passworddialog.h"
#include <QLabel>
#include <QLineEdit>
```

Include dependency graph for passworddialog.cpp:



14.45 passworddialog.cpp

Go to the documentation of this file.

```
00001 #include "passworddialog.h"
00002 #include "filecontent.h"
00003 #include "pass.h"
00004 #include "passwordconfiguration.h"
00005 #include "qtpasssettings.h"
00006 #include "ui_passworddialog.h"
00007
00008 #include <QLabel>
00009 #include <QLineEdit>
00010
00011 #ifdef QT_DEBUG
00012 #include "debughelper.h"
00013 #endif
00014
00020 PasswordDialog::PasswordDialog(const PasswordConfiguration &passConfig,
00021                                QWidget *parent)
00022     : QDialog(parent), ui(new Ui::PasswordDialog), m_passConfig(passConfig) {
00023   m_templating = false;
00024   m_allFields = false;
00025   m_isNew = false;
00026
```

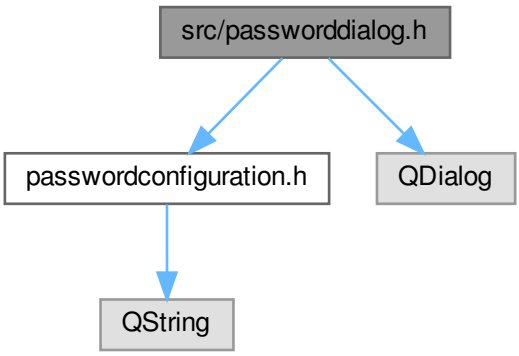
```
00027   ui->setupUi(this);
00028   setLength(m_passConfig.length);
00029   setPasswordCharTemplate(m_passConfig.selected);
00030
00031   connect(QtPassSettings::getPass(), &Pass::finishedShow, this,
00032           &PasswordDialog::setPass);
00033 }
00034
00041 PasswordDialog::PasswordDialog(const QString &file, const bool &isNew,
00042                                QWidget *parent)
00043     : QDialog(parent), ui(new Ui::PasswordDialog), m_file(file),
00044       m_isNew(isNew) {
00045
00046   if (!isNew)
00047     QtPassSettings::getPass()->Show(m_file);
00048
00049   ui->setupUi(this);
00050
00051   setWindowTitle(this->windowTitle() + " " + m_file);
00052   m_passConfig = QtPassSettings::getPasswordConfiguration();
00053   usePwgen(QtPassSettings::isUsePwgen());
00054   setTemplate(QtPassSettings::getPassTemplate(),
00055               QtPassSettings::isUseTemplate());
00056   templateAll(QtPassSettings::isTemplateAllFields());
00057
00058   setLength(m_passConfig.length);
00059   setPasswordCharTemplate(m_passConfig.selected);
00060
00061   connect(QtPassSettings::getPass(), &Pass::finishedShow, this,
00062           &PasswordDialog::setPass);
00063   connect(QtPassSettings::getPass(), &Pass::processErrorExit, this,
00064           &PasswordDialog::close);
00065   connect(this, &PasswordDialog::accepted, this, &PasswordDialog::on_accepted);
00066   connect(this, &PasswordDialog::rejected, this, &PasswordDialog::on_rejected);
00067 }
00068
00072 PasswordDialog::~PasswordDialog() { delete ui; }
00073
00078 void PasswordDialog::on_checkBoxShow_stateChanged(int arg1) {
00079   if (arg1)
00080     ui->lineEditPassword->setEchoMode(QLineEdit::Normal);
00081   else
00082     ui->lineEditPassword->setEchoMode(QLineEdit::Password);
00083 }
00084
00090 void PasswordDialog::on_createPasswordButton_clicked() {
00091   ui->widget->setEnabled(false);
00092   QString newPass = QtPassSettings::getPass()->Generate_b(
00093       static_cast<unsigned int>(ui->spinBox_pwdLength->value()),
00094       m_passConfig.Characters[static_cast<PasswordConfiguration::characterSet>(
00095           ui->passwordTemplateSwitch->currentIndex())]);
00096   if (newPass.length() > 0)
00097     ui->lineEditPassword->setText(newPass);
00098   ui->widget->setEnabled(true);
00099 }
00100
00104 void PasswordDialog::on_accepted() {
00105   QString newValue = getPassword();
00106   if (newValue.isEmpty())
00107     return;
00108
00109   if (newValue.right(1) != "\n")
00110     newValue += "\n";
00111
00112   QtPassSettings::getPass()->Insert(m_file, newValue, !m_isNew);
00113 }
00114
00118 void PasswordDialog::on_rejected() { setPassword(QString()); }
00119
00124 void PasswordDialog::setPassword(QString password) {
00125   FileContent fileContent = FileContent::parse(
00126       password, m_templating ? m_fields : QStringList(), m_allFields);
00127   ui->lineEditPassword->setText(fileContent.getPassword());
00128
00129   QWidget *previous = ui->checkBoxShow;
00130   // first set templated values
00131   NamedValues namedValues = fileContent.getNamedValues();
00132   for (QLineEdit *line : qAsConst(templateLines)) {
00133     line->setText(namedValues.takeValue(line->objectName()));
00134     previous = line;
00135   }
00136   // show remaining values (if there are)
00137   otherLines.clear();
00138   for (const NamedValue &nv : qAsConst(namedValues)) {
00139     auto *line = new QLineEdit();
00140     line->setObjectName(nv.name);
00141     line->setText(nv.value);
```

```
00142     ui->formLayout->addRow(new QLabel(nv.name), line);
00143     setTabOrder(previous, line);
00144     otherLines.append(line);
00145     previous = line;
00146   }
00147
00148   ui->plainTextEdit->insertPlainText(fileContent.getRemainingData());
00149 }
00150
00156 QString PasswordDialog::getPassword() {
00157   QString passFile = ui->lineEditPassword->text() + "\n";
00158   QList<QLineEdit *> allLines(templateLines);
00159   allLines.append(otherLines);
00160   for (QLineEdit *line : allLines) {
00161     QString text = line->text();
00162     if (text.isEmpty())
00163       continue;
00164     passFile += line->objectName() + ": " + text + "\n";
00165   }
00166   passFile += ui->plainTextEdit->toPlainText();
00167   return passFile;
00168 }
00169
00174 void PasswordDialog::setTemplate(QString rawFields, bool useTemplate) {
00175   m_fields = rawFields.split('\n');
00176   m_templating = useTemplate;
00177   templateLines.clear();
00178
00179   if (m_templating) {
00180     QWidget *previous = ui->checkBoxShow;
00181     foreach (QString field, m_fields) {
00182       if (field.isEmpty())
00183         continue;
00184       auto *line = new QLineEdit();
00185       line->setObjectName(field);
00186       ui->formLayout->addRow(new QLabel(field), line);
00187       setTabOrder(previous, line);
00188       templateLines.append(line);
00189       previous = line;
00190     }
00191   }
00192 }
00193
00199 void PasswordDialog::templateAll(bool templateAll) {
00200   m_allFields = templateAll;
00201 }
00202
00208 void PasswordDialog::setLength(int l) { ui->spinBox_pwdLength->setValue(l); }
00209
00215 void PasswordDialog::setPasswordCharTemplate(int t) {
00216   ui->passwordTemplateSwitch->setCurrentIndex(t);
00217 }
00218
00224 void PasswordDialog::usePwgen(bool usePwgen) {
00225   ui->passwordTemplateSwitch->setDisabled(usePwgen);
00226   ui->label_characterset->setDisabled(usePwgen);
00227 }
00228
00229 void PasswordDialog::setPass(const QString &output) {
00230   setPassword(output);
00231   //    TODO(bezet): enable ui
00232 }
```

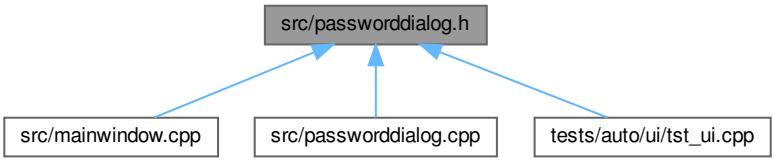
14.46 src/passworddialog.h File Reference

```
#include "passwordconfiguration.h"
#include <QDialog>
```

Include dependency graph for passworddialog.h:



This graph shows which files directly or indirectly include this file:



Classes

- class PasswordDialog

  PasswordDialog *Handles the inserting and editing of passwords.*

Namespaces

- namespace Ui

14.47 passworddialog.h

Go to the documentation of this file.

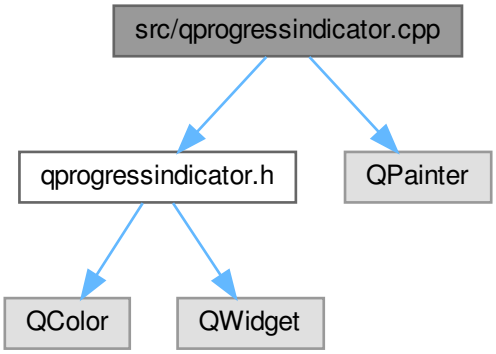
```
00001 #ifndef PASSWORDDIALOG_H_
00002 #define PASSWORDDIALOG_H_
00003
00004 #include "passwordconfiguration.h"
00005 #include <QDialog>
00006
00007 namespace Ui {
```

```
00008 class PasswordDialog;
00009 }
00010
00011 class QLineEdit;
00012 class QWidget;
00013
00020 class PasswordDialog : public QDialog {
00021   Q_OBJECT
00022
00023 public:
00024   explicit PasswordDialog(const PasswordConfiguration &passConfig,
00025                           QWidget *parent = nullptr);
00026   PasswordDialog(const QString &file, const bool &isNew,
00027                  QWidget *parent = nullptr);
00028   ~PasswordDialog();
00029
00034   void setPassword(QString password);
00035
00040   QString getPassword();
00041
00046   void setTemplate(QString rawFields, bool useTemplate);
00047
00048   void templateAll(bool templateAll);
00049   void setLength(int l);
00050   void setPasswordCharTemplate(int t);
00051   void usePwgen(bool usePwgen);
00052
00053 public slots:
00054   void setPass(const QString &output);
00055
00056 private slots:
00057   void on_checkBoxShow_stateChanged(int arg1);
00058   void on_createPasswordButton_clicked();
00059   void on_accepted();
00060   void on_rejected();
00061
00062 private:
00063   Ui::PasswordDialog *ui;
00064   PasswordConfiguration m_passConfig;
00065   QStringList m_fields;
00066   QString m_file;
00067   bool m_templating;
00068   bool m_allFields;
00069   bool m_isNew;
00070   QList<QLineEdit *> templateLines;
00071   QList<QLineEdit *> otherLines;
00072 };
00073
00074 #endif // PASSWORDDIALOG_H_
```

14.48 src/qprogressindicator.cpp File Reference

```
#include "qprogressindicator.h"
#include <QPainter>
```

Include dependency graph for qprogressindicator.cpp:



14.49 qprogressindicator.cpp

Go to the documentation of this file.

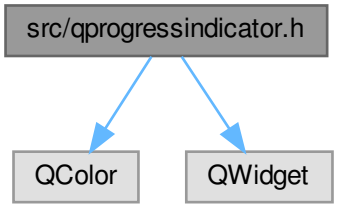
```
00001 /*
00002  * This code is based on https://github.com/mojocorp/QProgressIndicator
00003  * and published under
00004  *
00005  * The MIT License (MIT)
00006  *
00007  * Copyright (c) 2011 Morgan Leborgne
00008  *
00009  * Permission is hereby granted, free of charge, to any person obtaining a copy
00010  * of this software and associated documentation files (the "Software"), to deal
00011  * in the Software without restriction, including without limitation the rights
00012  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00013  * copies of the Software, and to permit persons to whom the Software is
00014  * furnished to do so, subject to the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be included in
00017  * all copies or substantial portions of the Software.
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00020  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00021  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00022  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00023  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00024  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00025  * SOFTWARE.
00026  */
00027 #include "qprogressindicator.h"
00028 #include <QPainter>
00029
00034 QProgressIndicator::QProgressIndicator(QWidget *parent)
00035     : QWidget(parent), m_angle(0), m_timerId(-1), m_delay(40),
00036       m_displayedWhenStopped(false), m_color(Qt::black) {
00037   setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);
00038   setFocusPolicy(Qt::NoFocus);
00039 }
00040
00041 bool QProgressIndicator::isAnimated() const { return m_timerId != -1; }
00042
00043 void QProgressIndicator::setDisplayedWhenStopped(bool state) {
00044   m_displayedWhenStopped = state;
00045
00046   update();
00047 }
00048
00049 bool QProgressIndicator::isDisplayedWhenStopped() const {
00050   return m_displayedWhenStopped;
00051 }
```

```
00052
00053 void QProgressIndicator::startAnimation() {
00054   m_angle = 0;
00055
00056   if (m_timerId == -1)
00057     m_timerId = startTimer(m_delay);
00058 }
00059
00060 void QProgressIndicator::stopAnimation() {
00061   if (m_timerId != -1)
00062     killTimer(m_timerId);
00063
00064   m_timerId = -1;
00065
00066   update();
00067 }
00068
00069 void QProgressIndicator::setAnimationDelay(int delay) {
00070   if (m_timerId != -1)
00071     killTimer(m_timerId);
00072
00073   m_delay = delay;
00074
00075   if (m_timerId != -1)
00076     m_timerId = startTimer(m_delay);
00077 }
00078
00079 void QProgressIndicator::setColor(const QColor &color) {
00080   m_color = color;
00081
00082   update();
00083 }
00084
00089 QSize QProgressIndicator::sizeHint() const { return {20, 20}; }
00090
00096 int QProgressIndicator::heightForWidth(int w) const { return w; }
00097
00101 void QProgressIndicator::timerEvent(QTimerEvent * /*event*/) {
00102   m_angle = (m_angle + 30) % 360;
00103
00104   update();
00105 }
00106
00110 void QProgressIndicator::paintEvent(QPaintEvent * /*event*/) {
00111   if (!m_displayedWhenStopped && !isAnimated())
00112     return;
00113
00114   int width = qMin(this->width(), this->height());
00115
00116   QPainter p(this);
00117   p.setRenderHint(QPainter::Antialiasing);
00118
00119   auto outerRadius = int((width - 1) * 0.5);
00120   auto innerRadius = int((width - 1) * 0.5 * 0.38);
00121
00122   int capsuleHeight = outerRadius - innerRadius;
00123   int capsuleWidth =
00124       (width > 32) ? int(capsuleHeight * 0.23) : int(capsuleHeight * 0.35);
00125   int capsuleRadius = capsuleWidth / 2;
00126
00127   for (int i = 0; i < 12; ++i) {
00128     QColor color = m_color;
00129     color.setAlphaF(int(1.0f - (i / 12.0f)));
00130     p.setPen(Qt::NoPen);
00131     p.setBrush(color);
00132     p.save();
00133     p.translate(rect().center());
00134     p.rotate(int(m_angle - i * 30.0f));
00135     p.drawRoundedRect(int(-capsuleWidth * 0.5), -(innerRadius + capsuleHeight),
00136                       capsuleWidth, capsuleHeight, capsuleRadius,
00137                       capsuleRadius);
00138     p.restore();
00139   }
00140 }
```

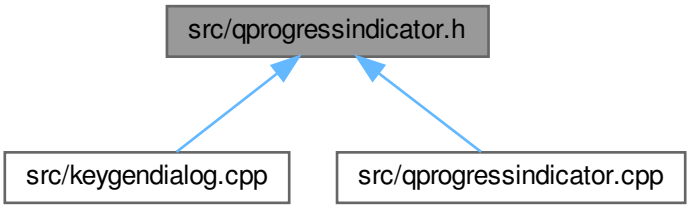
14.50 src/qprogressindicator.h File Reference

```
#include <QColor>
#include <QWidget>
```

Include dependency graph for qprogressindicator.h:



This graph shows which files directly or indirectly include this file:



Classes

- class QProgressIndicator

  The QProgressIndicator class lets an application display a progress indicator to show that a lengthy task is under way.

14.51 qprogressindicator.h

Go to the documentation of this file.

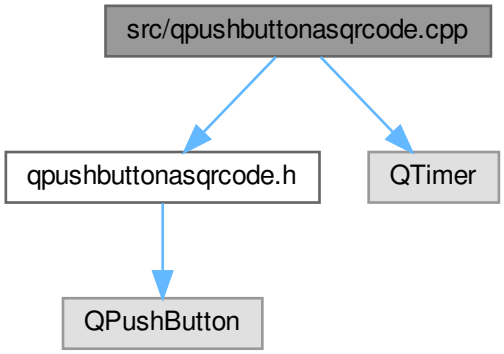
```
00001 /*
00002  * This code is based on https://github.com/mojocorp/QProgressIndicator
00003  * and published under
00004  *
00005  * The MIT License (MIT)
00006  *
00007  * Copyright (c) 2011 Morgan Leborgne
00008  *
00009  * Permission is hereby granted, free of charge, to any person obtaining a copy
00010  * of this software and associated documentation files (the "Software"), to deal
00011  * in the Software without restriction, including without limitation the rights
00012  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
00013  * copies of the Software, and to permit persons to whom the Software is
00014  * furnished to do so, subject to the following conditions:
00015  *
00016  * The above copyright notice and this permission notice shall be included in
00017  * all copies or substantial portions of the Software.
```

```
00018  *
00019  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
00020  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
00021  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
00022  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
00023  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
00024  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
00025  * SOFTWARE.
00026  */
00027
00028 #ifndef QPROGRESSINDICATOR_H_
00029 #define QPROGRESSINDICATOR_H_
00030
00031 #include <QColor>
00032 #include <QWidget>
00033
00043 class QProgressIndicator : public QWidget {
00044   Q_OBJECT
00045 public:
00046   explicit QProgressIndicator(QWidget *parent = 0);
00047
00053   int animationDelay() const { return m_delay; }
00054
00060   bool isAnimated() const;
00061
00068   bool isDisplayedWhenStopped() const;
00069
00073   const QColor &color() const { return m_color; }
00074
00075   virtual QSize sizeHint() const;
00076   int heightForWidth(int w) const;
00077
00078 public slots:
00082   void startAnimation();
00083
00087   void stopAnimation();
00088
00095   void setAnimationDelay(int delay);
00096
00102   void setDisplayedWhenStopped(bool state);
00103
00107   void setColor(const QColor &color);
00108
00109 protected:
00110   virtual void timerEvent(QTimerEvent *event);
00111   virtual void paintEvent(QPaintEvent *event);
00112
00113 private:
00114   int m_angle;
00115   int m_timerId;
00116   int m_delay;
00117   bool m_displayedWhenStopped;
00118   QColor m_color;
00119 };
00120
00121 #endif // QPROGRESSINDICATOR_H_
```

14.52 src/qpushbuttonasqrcode.cpp File Reference

```
#include "qpushbuttonasqrcode.h"
#include <QTimer>
```

Include dependency graph for qpushbuttonasqrcode.cpp:



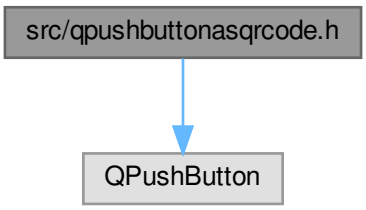
14.53 qpushbuttonasqrcode.cpp

Go to the documentation of this file.

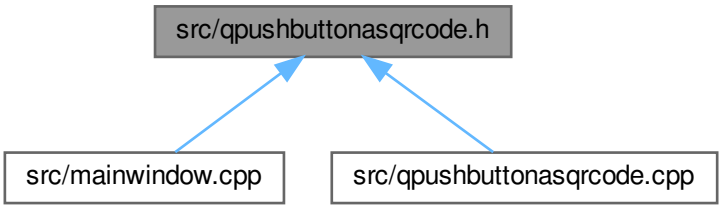
```
00001 #include "qpushbuttonasqrcode.h"
00002 #include <QTimer>
00003
00012 QPushButtonAsQRCode::QPushButtonAsQRCode(const QString &textToCopy,
00013                                          QWidget *parent)
00014     : QPushButton(parent), textToCopy(textToCopy),
00015       iconEdit(QIcon::fromTheme("qrcode", QIcon(":/icons/qrcode.svg"))) {
00016   setIcon(iconEdit);
00017   connect(this, SIGNAL(clicked(bool)), this, SLOT(buttonClicked(bool)));
00018 }
00019
00025 QString QPushButtonAsQRCode::getTextToCopy() const { return textToCopy; }
00026
00032 void QPushButtonAsQRCode::setTextToCopy(const QString &value) {
00033   textToCopy = value;
00034 }
00035
00040 void QPushButtonAsQRCode::buttonClicked(bool) { emit clicked(textToCopy); }
00041
00046 void QPushButtonAsQRCode::changeIconDefault() { this->setIcon(iconEdit); }
```

14.54 src/qpushbuttonasqrcode.h File Reference

`#include <QPushButton>`
Include dependency graph for qpushbuttonasqrcode.h:



This graph shows which files directly or indirectly include this file:



Classes

- class QPushButtonAsQRCode

  Stylish widget to display the field as QR Code.

14.55 qpushbuttonasqrcode.h

Go to the documentation of this file.

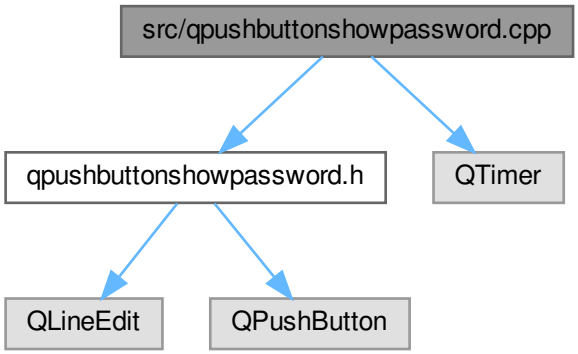
```
00001 #ifndef QPUSHBUTTONASQRCODE_H_
00002 #define QPUSHBUTTONASQRCODE_H_
00003
00004 #include <QPushButton>
00005
00010 class QWidget;
00011 class QPushButtonAsQRCode : public QPushButton {
00012   Q_OBJECT
00013
00014 public:
00015   explicit QPushButtonAsQRCode(const QString &textToCopy = "",
```

```
00016                               QWidget *parent = nullptr);
00017
00018   QString getTextToCopy() const;
00019   void setTextToCopy(const QString &value);
00020
00021 signals:
00022   void clicked(QString);
00023
00024 private slots:
00025   void changeIconDefault();
00026   void buttonClicked(bool);
00027
00028 private:
00029   QString textToCopy;
00030   QIcon iconEdit;
00031 };
00032
00033 #endif // QPUSHBUTTONASQRCODE_H_
```

14.56 src/qpushbuttonshowpassword.cpp File Reference

```
#include "qpushbuttonshowpassword.h"
#include <QTimer>
```

Include dependency graph for qpushbuttonshowpassword.cpp:



14.57 qpushbuttonshowpassword.cpp

Go to the documentation of this file.

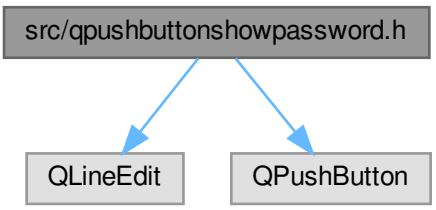
```
00001 #include "qpushbuttonshowpassword.h"
00002 #include <QTimer>
00003
00012 QPushButtonShowPassword::QPushButtonShowPassword(QLineEdit *line,
00013                                                 QWidget *parent)
00014     : QPushButton(parent),
00015       iconEdit(QIcon::fromTheme("show", QIcon(":/icons/view.svg"))),
00016       iconEditPushed(QIcon::fromTheme("hide-new", QIcon(":/icons/hide.svg"))) {
00017   setIcon(iconEdit);
00018   connect(this, SIGNAL(clicked(bool)), this, SLOT(buttonClicked(bool)));
00019   this->line = line;
00020 }
00021
00026 void QPushButtonShowPassword::buttonClicked(bool) {
00027   if (this->line->echoMode() == QLineEdit::Password) {
00028     this->line->setEchoMode(QLineEdit::Normal);
00029     setIcon(iconEditPushed);
```

```
00030   } else {
00031     this->line->setEchoMode(QLineEdit::Password);
00032     setIcon(iconEdit);
00033   }
00034 }
```

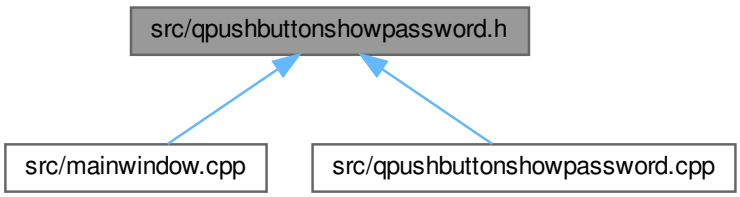
14.58 src/qpushbuttonshowpassword.h File Reference

```
#include <QLineEdit>
#include <QPushButton>
```

Include dependency graph for qpushbuttonshowpassword.h:



This graph shows which files directly or indirectly include this file:



Classes

- class QPushButtonShowPassword

14.59 qpushbuttonshowpassword.h

Go to the documentation of this file.

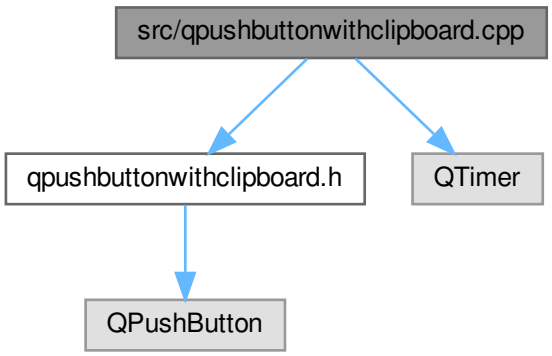
```
00001 #ifndef QPUSHBUTTONSHOWPASSWORD_H_
00002 #define QPUSHBUTTONSHOWPASSWORD_H_
00003
```

```
00004 #include <QLineEdit>
00005 #include <QPushButton>
00006
00011 class QWidget;
00012 class QPushButtonShowPassword : public QPushButton {
00013   Q_OBJECT
00014
00015 public:
00016   explicit QPushButtonShowPassword(QLineEdit *line, QWidget *parent = nullptr);
00017
00018 signals:
00019   void clicked(QString);
00020
00021 private slots:
00022   void buttonClicked(bool);
00023
00024 private:
00025   QIcon iconEdit;
00026   QIcon iconEditPushed;
00027   QLineEdit *line;
00028 };
00029
00030 #endif // QPUSHBUTTONSHOWPASSWORD_H_
```

14.60 src/qpushbuttonwithclipboard.cpp File Reference

```
#include "qpushbuttonwithclipboard.h"
#include <QTimer>
```

Include dependency graph for qpushbuttonwithclipboard.cpp:



14.61 qpushbuttonwithclipboard.cpp

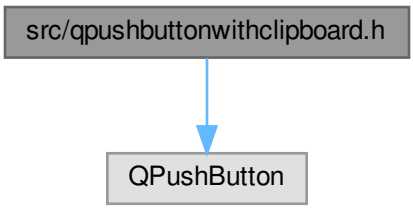
Go to the documentation of this file.

```
00001 #include "qpushbuttonwithclipboard.h"
00002 #include <QTimer>
00003
00012 QPushButtonWithClipboard::QPushButtonWithClipboard(const QString &textToCopy,
00013                                                    QWidget *parent)
00014     : QPushButton(parent), textToCopy(textToCopy),
00015       iconEdit(QIcon::fromTheme("edit-copy", QIcon(":/icons/edit-copy.svg"))),
00016       iconEditPushed(
00017           QIcon::fromTheme("document-new", QIcon(":/icons/document-new.svg"))) {
00018   setIcon(iconEdit);
00019   connect(this, SIGNAL(clicked(bool)), this, SLOT(buttonClicked(bool)));
00020 }
```

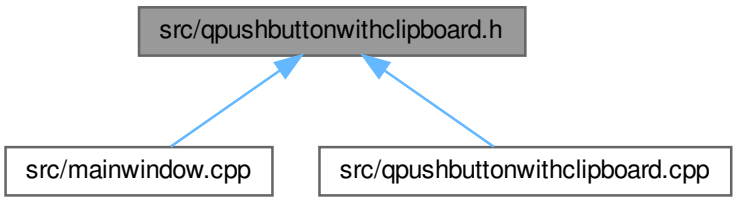
```
00021
00027 QString QPushButtonWithClipboard::getTextToCopy() const { return textToCopy; }
00028
00034 void QPushButtonWithClipboard::setTextToCopy(const QString &value) {
00035   textToCopy = value;
00036 }
00037
00042 void QPushButtonWithClipboard::buttonClicked(bool) {
00043   setIcon(iconEditPushed);
00044   QTimer::singleShot(500, this, SLOT(changeIconDefault()));
00045   emit clicked(textToCopy);
00046 }
00047
00052 void QPushButtonWithClipboard::changeIconDefault() { this->setIcon(iconEdit); }
```

14.62 src/qpushbuttonwithclipboard.h File Reference

`#include <QPushButton>`
Include dependency graph for qpushbuttonwithclipboard.h:



This graph shows which files directly or indirectly include this file:



Classes

- class QPushButtonWithClipboard

  Stylish widget to allow copying of password and account details.

14.63 qpushbuttonwithclipboard.h

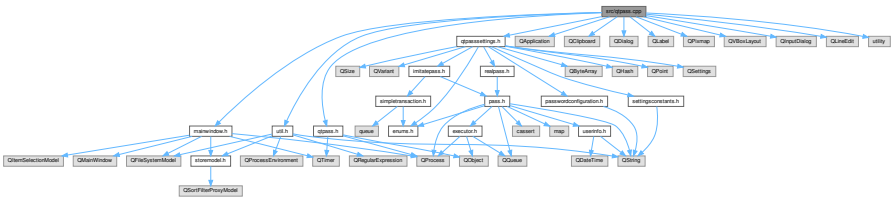
Go to the documentation of this file.

```
00001 #ifndef QPUSHBUTTONWITHCLIPBOARD_H_
00002 #define QPUSHBUTTONWITHCLIPBOARD_H_
00003
00004 #include <QPushButton>
00005
00010 class QWidget;
00011 class QPushButtonWithClipboard : public QPushButton {
00012   Q_OBJECT
00013
00014 public:
00015   explicit QPushButtonWithClipboard(const QString &textToCopy = "",
00016                                     QWidget *parent = nullptr);
00017
00018   QString getTextToCopy() const;
00019   void setTextToCopy(const QString &value);
00020
00021 signals:
00022   void clicked(QString);
00023
00024 private slots:
00025   void changeIconDefault();
00026   void buttonClicked(bool);
00027
00028 private:
00029   QString textToCopy;
00030   QIcon iconEdit;
00031   QIcon iconEditPushed;
00032 };
00033
00034 #endif // QPUSHBUTTONWITHCLIPBOARD_H_
```

14.64 src/qtpass.cpp File Reference

```
#include "qtpass.h"
#include "mainwindow.h"
#include "qtpasssettings.h"
#include "util.h"
#include <QApplication>
#include <QClipboard>
#include <QDialog>
#include <QLabel>
#include <QPixmap>
#include <QVBoxLayout>
#include <QInputDialog>
#include <QLineEdit>
#include <utility>
```

Include dependency graph for qtpass.cpp:



14.65 qtpass.cpp

Go to the documentation of this file.

```
00001 #include "qtpass.h"
00002 #include "mainwindow.h"
00003 #include "qtpasssettings.h"
00004 #include "util.h"
00005 #include <QApplication>
00006 #include <QClipboard>
00007 #include <QDialog>
00008 #include <QLabel>
00009 #include <QPixmap>
00010 #include <QVBoxLayout>
00011
00012 #ifndef Q_OS_WIN
00013 #include <QInputDialog>
00014 #include <QLineEdit>
00015 #include <utility>
00016 #else
00017 #define WIN32_LEAN_AND_MEAN /*_KILLING_MACHINE*/
00018 #define WIN32_EXTRA_LEAN
00019 #include <windows.h>
00020 #include <winnetwk.h>
00021 #undef DELETE
00022 #endif
00023
00024 #ifdef QT_DEBUG
00025 #include "debughelper.h"
00026 #endif
00027
00028 QtPass::QtPass(MainWindow *mainWindow)
00029     : m_mainWindow(mainWindow), clippedText(QString()), freshStart(true) {
00030   setClipboardTimer();
00031   clearClipboardTimer.setSingleShot(true);
00032   connect(&clearClipboardTimer, SIGNAL(timeout()), this,
00033           SLOT(clearClipboard()));
00034
00035   QObject::connect(qApp, &QApplication::aboutToQuit, this,
00036                    &QtPass::clearClipboard);
00037
00038   setMainWindow();
00039 }
00040
00044 QtPass::~QtPass() {
00045 #ifdef Q_OS_WIN
00046   if (QtPassSettings::isUseWebDav())
00047     WNetCancelConnection2A(QtPassSettings::getPassStore().toUtf8().constData(),
00048                            0, 1);
00049 #else
00050   if (fusedav.state() == QProcess::Running) {
00051     fusedav.terminate();
00052     fusedav.waitForFinished(2000);
00053   }
00054 #endif
00055 }
00056
00061 bool QtPass::init() {
00062   QString passStore = QtPassSettings::getPassStore(Util::findPasswordStore());
00063   QtPassSettings::setPassStore(passStore);
00064
00065   QtPassSettings::initExecutables();
00066
00067   QString version = QtPassSettings::getVersion();
00068   // dbg()« version;
00069
00070   // Config updates
00071   if (version.isEmpty()) {
00072 #ifdef QT_DEBUG
00073     dbg() « "assuming fresh install";
00074 #endif
00075
00076     if (QtPassSettings::getAutoclearSeconds() < 5)
00077       QtPassSettings::setAutoclearSeconds(10);
00078     if (QtPassSettings::getAutoclearPanelSeconds() < 5)
00079       QtPassSettings::setAutoclearPanelSeconds(10);
00080     if (!QtPassSettings::getPwgenExecutable().isEmpty())
00081       QtPassSettings::setUsePwgen(true);
00082     else
00083       QtPassSettings::setUsePwgen(false);
00084     QtPassSettings::setPassTemplate("login\nurl");
00085   } else {
00086     // QStringList ver = version.split(".");
00087     // dbg()« ver;
00088     // if (ver[0] == "0" && ver[1] == "8") {
00090     // }
00091     if (QtPassSettings::getPassTemplate().isEmpty())
00092       QtPassSettings::setPassTemplate("login\nurl");
00093   }
00094
00095   QtPassSettings::setVersion(VERSION);
```

```
00096
00097   if (Util::checkConfig()) {
00098     m_mainWindow->config();
00099     if (freshStart && Util::checkConfig())
00100       return false;
00101   }
00102
00103   // TODO(annejan): this needs to be before we try to access the store,
00104   // but it would be better to do it after the Window is shown,
00105   // as the long delay it can cause is irritating otherwise.
00106   if (QtPassSettings::isUseWebDav())
00107     mountWebDav();
00108
00109   freshStart = false;
00110   // startupPhase = false;
00111   return true;
00112 }
00113
00114 void QtPass::setMainWindow(void) {
00115   m_mainWindow->restoreWindow();
00116
00117   fusedav.setParent(m_mainWindow);
00118
00119   //  TODO(bezet): this should be reconnected dynamically when pass changes
00120   connectPassSignalHandlers(QtPassSettings::getRealPass());
00121   connectPassSignalHandlers(QtPassSettings::getImitatePass());
00122
00123   connect(m_mainWindow, &MainWindow::passShowHandlerFinished, this,
00124           &QtPass::passShowHandlerFinished);
00125
00126   // only for ipass
00127   connect(QtPassSettings::getImitatePass(), &ImitatePass::startReencryptPath,
00128           m_mainWindow, &MainWindow::startReencryptPath);
00129   connect(QtPassSettings::getImitatePass(), &ImitatePass::endReencryptPath,
00130           m_mainWindow, &MainWindow::endReencryptPath);
00131
00132   connect(m_mainWindow, &MainWindow::passGitInitNeeded, [=]() {
00133 #ifdef QT_DEBUG
00134     dbg() « "Pass git init called";
00135 #endif
00136     QtPassSettings::getPass()->GitInit();
00137   });
00138
00139   connect(m_mainWindow, &MainWindow::generateGPGKeyPair, m_mainWindow,
00140           [=](const QString &batch) {
00141             QtPassSettings::getPass()->GenerateGPGKeys(batch);
00142             m_mainWindow->showStatusMessage(tr("Generating GPG key pair"),
00143                                             60000);
00144           });
00145 }
00146
00147 void QtPass::connectPassSignalHandlers(Pass *pass) {
00148   connect(pass, &Pass::error, this, &QtPass::processError);
00149   connect(pass, &Pass::processErrorExit, this, &QtPass::processErrorExit);
00150   connect(pass, &Pass::critical, m_mainWindow, &MainWindow::critical);
00151   connect(pass, &Pass::startingExecuteWrapper, m_mainWindow,
00152           &MainWindow::executeWrapperStarted);
00153   connect(pass, &Pass::statusMsg, m_mainWindow, &MainWindow::showStatusMessage);
00154   connect(pass, &Pass::finishedShow, m_mainWindow,
00155           &MainWindow::passShowHandler);
00156   connect(pass, &Pass::finishedOtpGenerate, m_mainWindow,
00157           &MainWindow::passOtpHandler);
00158
00159   connect(pass, &Pass::finishedGitInit, this, &QtPass::passStoreChanged);
00160   connect(pass, &Pass::finishedGitPull, this, &QtPass::processFinished);
00161   connect(pass, &Pass::finishedGitPush, this, &QtPass::processFinished);
00162   connect(pass, &Pass::finishedInsert, this, &QtPass::finishedInsert);
00163   connect(pass, &Pass::finishedRemove, this, &QtPass::passStoreChanged);
00164   connect(pass, &Pass::finishedInit, this, &QtPass::passStoreChanged);
00165   connect(pass, &Pass::finishedMove, this, &QtPass::passStoreChanged);
00166   connect(pass, &Pass::finishedCopy, this, &QtPass::passStoreChanged);
00167   connect(pass, &Pass::finishedGenerateGPGKeys, this,
00168           &QtPass::onKeyGenerationComplete);
00169 }
00170
00174 void QtPass::mountWebDav() {
00175 #ifdef Q_OS_WIN
00176   char dst[20] = {0};
00177   NETRESOURCEA netres;
00178   memset(&netres, 0, sizeof(netres));
00179   netres.dwType = RESOURCETYPE_DISK;
00180   netres.lpLocalName = 0;
00181   netres.lpRemoteName = QtPassSettings::getWebDavUrl().toUtf8().data();
00182   DWORD size = sizeof(dst);
00183   DWORD r = WNetUseConnectionA(
00184       reinterpret_cast<HWND>(m_mainWindow->effectiveWinId()), &netres,
00185       QtPassSettings::getWebDavPassword().toUtf8().constData(),
```

```
00186       QtPassSettings::getWebDavUser().toUtf8().constData(),
00187       CONNECT_TEMPORARY | CONNECT_INTERACTIVE | CONNECT_REDIRECT, dst, &size,
00188       0);
00189   if (r == NO_ERROR) {
00190     QtPassSettings::setPassStore(dst);
00191   } else {
00192     char message[256] = {0};
00193     FormatMessageA(FORMAT_MESSAGE_FROM_SYSTEM, 0, r, 0, message,
00194                    sizeof(message), 0);
00195     m_mainWindow->flashText(tr("Failed to connect WebDAV:\n") + message +
00196                             " (0x" + QString::number(r, 16) + ")",
00197                             true);
00198   }
00199 #else
00200   fusedav.start("fusedav", QStringList()
00201                              << "-o"
00202                              << "nonempty"
00203                              << "-u"
00204                              << "\"" + QtPassSettings::getWebDavUser() + "\""
00205                              << QtPassSettings::getWebDavUrl()
00206                              << "\"" + QtPassSettings::getPassStore() + "\"");
00207   fusedav.waitForStarted();
00208   if (fusedav.state() == QProcess::Running) {
00209     QString pwd = QtPassSettings::getWebDavPassword();
00210     bool ok = true;
00211     if (pwd.isEmpty()) {
00212       pwd = QInputDialog::getText(m_mainWindow, tr("QtPass WebDAV password"),
00213                                   tr("Enter password to connect to WebDAV:"),
00214                                   QLineEdit::Password, "", &ok);
00215     }
00216     if (ok && !pwd.isEmpty()) {
00217       fusedav.write(pwd.toUtf8() + '\n');
00218       fusedav.closeWriteChannel();
00219       fusedav.waitForFinished(2000);
00220     } else {
00221       fusedav.terminate();
00222     }
00223   }
00224   QString error = fusedav.readAllStandardError();
00225   int prompt = error.indexOf("Password:");
00226   if (prompt >= 0)
00227     error.remove(0, prompt + 10);
00228   if (fusedav.state() != QProcess::Running)
00229     error = tr("fusedav exited unexpectedly\n") + error;
00230   if (error.size() > 0) {
00231     m_mainWindow->flashText(
00232         tr("Failed to start fusedav to connect WebDAV:\n") + error, true);
00233   }
00234 #endif
00235 }
00236
00241 void QtPass::processError(QProcess::ProcessError error) {
00242   QString errorString;
00243   switch (error) {
00244   case QProcess::FailedToStart:
00245     errorString = tr("QProcess::FailedToStart");
00246     break;
00247   case QProcess::Crashed:
00248     errorString = tr("QProcess::Crashed");
00249     break;
00250   case QProcess::Timedout:
00251     errorString = tr("QProcess::Timedout");
00252     break;
00253   case QProcess::ReadError:
00254     errorString = tr("QProcess::ReadError");
00255     break;
00256   case QProcess::WriteError:
00257     errorString = tr("QProcess::WriteError");
00258     break;
00259   case QProcess::UnknownError:
00260     errorString = tr("QProcess::UnknownError");
00261     break;
00262   }
00263
00264   m_mainWindow->flashText(errorString, true);
00265   m_mainWindow->setUiElementsEnabled(true);
00266 }
00267
00268 void QtPass::processErrorExit(int exitCode, const QString &p_error) {
00269   if (!p_error.isEmpty()) {
00270     QString output;
00271     QString error = p_error.toHtmlEscaped();
00272     if (exitCode == 0) {
00273       //  https://github.com/IJHack/qtpass/issues/111
00274       output = "<span style=\"color: darkgray;\">" + error + "</span><br />";
00275     } else {
00276       output = "<span style=\"color: red;\">" + error + "</span><br />";
```

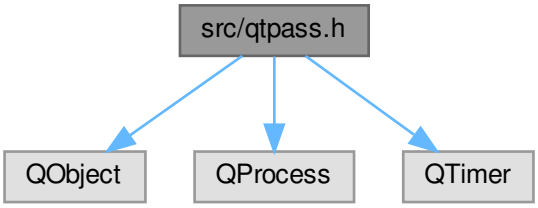
```
00277     }
00278
00279     output.replace(Util::protocolRegex(), R"(<a href="\1">\1</a>)");
00280     output.replace(QStringLiteral("\n"), "<br />");
00281
00282     m_mainWindow->flashText(output, false, true);
00283   }
00284
00285   m_mainWindow->setUiElementsEnabled(true);
00286 }
00287
00295 void QtPass::processFinished(const QString &p_output, const QString &p_errout) {
00296   showInTextBrowser(p_output);
00297   //    Sometimes there is error output even with 0 exit code, which is
00298   //    assumed in this function
00299   processErrorExit(0, p_errout);
00300
00301   m_mainWindow->setUiElementsEnabled(true);
00302 }
00303
00304 void QtPass::passStoreChanged(const QString &p_out, const QString &p_err) {
00305   processFinished(p_out, p_err);
00306   doGitPush();
00307 }
00308
00309 void QtPass::finishedInsert(const QString &p_output, const QString &p_errout) {
00310   processFinished(p_output, p_errout);
00311   doGitPush();
00312   m_mainWindow->on_treeView_clicked(m_mainWindow->getCurrentTreeViewIndex());
00313 }
00314
00315 void QtPass::onKeyGenerationComplete(const QString &p_output,
00316                                      const QString &p_errout) {
00317   if (nullptr != m_mainWindow->getKeygenDialog()) {
00318 #ifdef QT_DEBUG
00319     qDebug() << "Keygen Done";
00320 #endif
00321
00322     m_mainWindow->cleanKeygenDialog();
00323     // TODO(annejan) some sanity checking ?
00324   }
00325
00326   processFinished(p_output, p_errout);
00327 }
00328
00329 void QtPass::passShowHandlerFinished(QString output) {
00330   showInTextBrowser(std::move(output));
00331 }
00332
00333 void QtPass::showInTextBrowser(QString output, QString prefix,
00334                                QString postfix) {
00335   output = output.toHtmlEscaped();
00336
00337   output.replace(Util::protocolRegex(), R"(<a href="\1">\1</a>)");
00338   output.replace(QStringLiteral("\n"), "<br />");
00339   output = prefix + output + postfix;
00340
00341   m_mainWindow->flashText(output, false, true);
00342 }
00343
00344 void QtPass::doGitPush() {
00345   if (QtPassSettings::isAutoPush())
00346     m_mainWindow->onPush();
00347 }
00348
00349 void QtPass::setClippedText(const QString &password, const QString &p_output) {
00350   if (QtPassSettings::getClipBoardType() != Enums::CLIPBOARD_NEVER &&
00351       !p_output.isEmpty()) {
00352     clippedText = password;
00353     if (QtPassSettings::getClipBoardType() == Enums::CLIPBOARD_ALWAYS)
00354       copyTextToClipboard(password);
00355   }
00356 }
00357 void QtPass::clearClippedText() { clippedText = ""; }
00358
00359 void QtPass::setClipboardTimer() {
00360   clearClipboardTimer.setInterval(1000 * QtPassSettings::getAutoclearSeconds());
00361 }
00362
00366 void QtPass::clearClipboard() {
00367   QClipboard *clipboard = QApplication::clipboard();
00368   bool cleared = false;
00369   if (this->clippedText == clipboard->text(QClipboard::Selection)) {
00370     clipboard->clear(QClipboard::Selection);
00371     clipboard->setText(QString(""), QClipboard::Selection);
00372     cleared = true;
00373   }
```

```
00374   if (this->clippedText == clipboard->text(QClipboard::Clipboard)) {
00375     clipboard->clear(QClipboard::Clipboard);
00376     cleared = true;
00377   }
00378   if (cleared) {
00379     m_mainWindow->showStatusMessage(tr("Clipboard cleared"));
00380   } else {
00381     m_mainWindow->showStatusMessage(tr("Clipboard not cleared"));
00382   }
00383
00384   clippedText.clear();
00385 }
00386
00391 void QtPass::copyTextToClipboard(const QString &text) {
00392   QClipboard *clip = QApplication::clipboard();
00393   if (!QtPassSettings::isUseSelection()) {
00394     clip->setText(text, QClipboard::Clipboard);
00395   } else {
00396     clip->setText(text, QClipboard::Selection);
00397   }
00398
00399   clippedText = text;
00400   m_mainWindow->showStatusMessage(tr("Copied to clipboard"));
00401   if (QtPassSettings::isUseAutoclear()) {
00402     clearClipboardTimer.start();
00403   }
00404 }
00405
00410 void QtPass::showTextAsQRCode(const QString &text) {
00411   QProcess qrencode;
00412   qrencode.start(QtPassSettings::getQrencodeExecutable("/usr/bin/qrencode"),
00413                  QStringList() << "-o-"
00414                                << "-tPNG");
00415   qrencode.write(text.toUtf8());
00416   qrencode.closeWriteChannel();
00417   qrencode.waitForFinished();
00418   QByteArray output(qrencode.readAllStandardOutput());
00419
00420   if (qrencode.exitStatus() || qrencode.exitCode()) {
00421     QString error(qrencode.readAllStandardError());
00422     m_mainWindow->showStatusMessage(error);
00423   } else {
00424     QPixmap image;
00425     image.loadFromData(output, "PNG");
00426
00427     QDialog *popup = new QDialog(0, Qt::Popup | Qt::FramelessWindowHint);
00428     QVBoxLayout *layout = new QVBoxLayout;
00429     QLabel *popupLabel = new QLabel();
00430     layout->addWidget(popupLabel);
00431     popupLabel->setPixmap(image);
00432     popupLabel->setScaledContents(true);
00433     popupLabel->show();
00434     popup->setLayout(layout);
00435     popup->move(QCursor::pos());
00436     popup->exec();
00437   }
00438 }
```

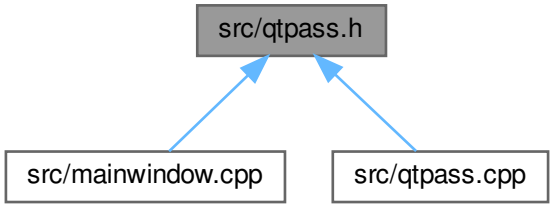
14.66 src/qtpass.h File Reference

```
#include <QObject>
#include <QProcess>
#include <QTimer>
```

Include dependency graph for qtpass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class QtPass

14.67 qtpass.h

Go to the documentation of this file.

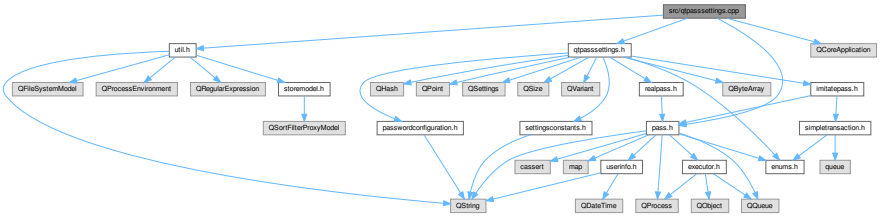
```
00001 #ifndef QTPASS_H
00002 #define QTPASS_H
00003
00004 #include <QObject>
00005 #include <QProcess>
00006 #include <QTimer>
00007
00008 class MainWindow;
00009 class Pass;
00010 class QtPass : public QObject {
00011   Q_OBJECT
00012
00013 public:
00014   QtPass(MainWindow *mainWindow);
00015   ~QtPass();
00016
00017   bool init();
00018   void setClippedText(const QString &, const QString &p_output = QString());
00019   void clearClippedText();
```

```
00020   void setClipboardTimer();
00021   bool isFreshStart() { return this->freshStart; }
00022   void setFreshStart(const bool &fs) { this->freshStart = fs; }
00023
00024 private:
00025   MainWindow *m_mainWindow;
00026
00027   QProcess fusedav;
00028
00029   QTimer clearClipboardTimer;
00030   QString clippedText;
00031   bool freshStart;
00032
00033   void setMainWindow();
00034   void connectPassSignalHandlers(Pass *pass);
00035   void mountWebDav();
00036
00037 signals:
00038
00039 public slots:
00040   void clearClipboard();
00041   void copyTextToClipboard(const QString &text);
00042   void showTextAsQRCode(const QString &text);
00043
00044 private slots:
00045   void processError(QProcess::ProcessError);
00046   void processErrorExit(int exitCode, const QString &);
00047   void processFinished(const QString &, const QString &);
00048
00049   void passStoreChanged(const QString &, const QString &);
00050   void passShowHandlerFinished(QString output);
00051
00052   void doGitPush();
00053   void finishedInsert(const QString &, const QString &);
00054   void onKeyGenerationComplete(const QString &p_output,
00055                                const QString &p_errout);
00056
00057   void showInTextBrowser(QString output, QString prefix = QString(),
00058                          QString postfix = QString());
00059 };
00060
00061 #endif // QTPASS_H
```

14.68 src/qtpasssettings.cpp File Reference

```
#include "qtpasssettings.h"
#include "pass.h"
#include "util.h"
#include <QCoreApplication>
```

Include dependency graph for qtpasssettings.cpp:



14.69 qtpasssettings.cpp

Go to the documentation of this file.

```
00001 #include "qtpasssettings.h"
00002 #include "pass.h"
00003
00004 #include "util.h"
```

```
00005
00006 #include <QCoreApplication>
00007
00008 bool QtPassSettings::initialized = false;
00009
00010 Pass *QtPassSettings::pass;
00011 // Go via pointer to avoid dynamic initialization,
00012 // due to "random" initialization order realtive to other
00013 // globals, especially around QObject emtadata dynamic initialization
00014 // can lead to crashes depending on compiler, linker etc.
00015 QScopedPointer<RealPass> QtPassSettings::realPass;
00016 QScopedPointer<ImitatePass> QtPassSettings::imitatePass;
00017
00018 QtPassSettings *QtPassSettings::m_instance = nullptr;
00019 QtPassSettings *QtPassSettings::getInstance() {
00020   if (!QtPassSettings::initialized) {
00021     QString portable_ini = QCoreApplication::applicationDirPath() +
00022                            QDir::separator() + "qtpass.ini";
00023     if (QFile(portable_ini).exists()) {
00024       m_instance = new QtPassSettings(portable_ini, QSettings::IniFormat);
00025     } else {
00026       m_instance = new QtPassSettings("IJHack", "QtPass");
00027     }
00028
00029     initialized = true;
00030   }
00031
00032   return m_instance;
00033 }
00034
00035 PasswordConfiguration QtPassSettings::getPasswordConfiguration() {
00036   PasswordConfiguration config;
00037
00038   config.length =
00039       getInstance()->value(SettingsConstants::passwordLength, 0).toInt();
00040   config.selected = static_cast<PasswordConfiguration::characterSet>(
00041       getInstance()
00042           ->value(SettingsConstants::passwordCharsselection, 0)
00043           .toInt());
00044   config.Characters[PasswordConfiguration::CUSTOM] =
00045       getInstance()
00046           ->value(SettingsConstants::passwordChars, QString())
00047           .toString();
00048
00049   return config;
00050 }
00051
00052 void QtPassSettings::setPasswordConfiguration(
00053     const PasswordConfiguration &config) {
00054   getInstance()->setValue(SettingsConstants::passwordLength, config.length);
00055   getInstance()->setValue(SettingsConstants::passwordCharsselection,
00056                           config.selected);
00057   getInstance()->setValue(SettingsConstants::passwordChars,
00058                           config.Characters[PasswordConfiguration::CUSTOM]);
00059 }
00060
00061 QHash<QString, QHash<QString, QString>> QtPassSettings::getProfiles() {
00062   getInstance()->beginGroup(SettingsConstants::profile);
00063   QHash<QString, QHash<QString, QString>> profiles;
00064
00065   // migration from version <= v1.3.2: profiles datastructure
00066   QStringList childKeys = getInstance()->childKeys();
00067   if (!childKeys.empty()) {
00068     foreach (QString key, childKeys) {
00069       QHash<QString, QString> profile;
00070       profile.insert("path", getInstance()->value(key).toString());
00071       profile.insert("signingKey", "");
00072       profiles.insert(key, profile);
00073     }
00074   }
00075   // /migration from version <= v1.3.2
00076
00077   QStringList childGroups = getInstance()->childGroups();
00078   foreach (QString group, childGroups) {
00079     QHash<QString, QString> profile;
00080     profile.insert("path", getInstance()->value(group + "/path").toString());
00081     profile.insert("signingKey",
00082                    getInstance()->value(group + "/signingKey").toString());
00083     // profiles.insert(group, getInstance()->value(group).toString());
00084     profiles.insert(group, profile);
00085   }
00086
00087   getInstance()->endGroup();
00088
00089   return profiles;
00090 }
00091
```

```
00092 void QtPassSettings::setProfiles(
00093     const QHash<QString, QHash<QString, QString» &profiles) {
00094   getInstance()->remove(SettingsConstants::profile);
00095   getInstance()->beginGroup(SettingsConstants::profile);
00096
00097   QHash<QString, QHash<QString, QString»::const_iterator i = profiles.begin();
00098   for (; i != profiles.end(); ++i) {
00099     getInstance()->setValue(i.key() + "/path", i.value().value("path"));
00100     getInstance()->setValue(i.key() + "/signingKey",
00101                             i.value().value("signingKey"));
00102   }
00103
00104   getInstance()->endGroup();
00105 }
00106
00107 Pass *QtPassSettings::getPass() {
00108   if (!pass) {
00109     if (isUsePass()) {
00110       QtPassSettings::pass = getRealPass();
00111     } else {
00112       QtPassSettings::pass = getImitatePass();
00113     }
00114     pass->init();
00115   }
00116   return pass;
00117 }
00118
00119 QString QtPassSettings::getVersion(const QString &defaultValue) {
00120   return getInstance()
00121       ->value(SettingsConstants::version, defaultValue)
00122       .toString();
00123 }
00124 void QtPassSettings::setVersion(const QString &version) {
00125   getInstance()->setValue(SettingsConstants::version, version);
00126 }
00127
00128 QByteArray QtPassSettings::getGeometry(const QByteArray &defaultValue) {
00129   return getInstance()
00130       ->value(SettingsConstants::geometry, defaultValue)
00131       .toByteArray();
00132 }
00133 void QtPassSettings::setGeometry(const QByteArray &geometry) {
00134   getInstance()->setValue(SettingsConstants::geometry, geometry);
00135 }
00136
00137 QByteArray QtPassSettings::getSavestate(const QByteArray &defaultValue) {
00138   return getInstance()
00139       ->value(SettingsConstants::savestate, defaultValue)
00140       .toByteArray();
00141 }
00142 void QtPassSettings::setSavestate(const QByteArray &saveState) {
00143   getInstance()->setValue(SettingsConstants::savestate, saveState);
00144 }
00145
00146 QPoint QtPassSettings::getPos(const QPoint &defaultValue) {
00147   return getInstance()->value(SettingsConstants::pos, defaultValue).toPoint();
00148 }
00149 void QtPassSettings::setPos(const QPoint &pos) {
00150   getInstance()->setValue(SettingsConstants::pos, pos);
00151 }
00152
00153 QSize QtPassSettings::getSize(const QSize &defaultValue) {
00154   return getInstance()->value(SettingsConstants::size, defaultValue).toSize();
00155 }
00156 void QtPassSettings::setSize(const QSize &size) {
00157   getInstance()->setValue(SettingsConstants::size, size);
00158 }
00159
00160 bool QtPassSettings::isMaximized(const bool &defaultValue) {
00161   return getInstance()
00162       ->value(SettingsConstants::maximized, defaultValue)
00163       .toBool();
00164 }
00165 void QtPassSettings::setMaximized(const bool &maximized) {
00166   getInstance()->setValue(SettingsConstants::maximized, maximized);
00167 }
00168
00169 bool QtPassSettings::isUsePass(const bool &defaultValue) {
00170   return getInstance()
00171       ->value(SettingsConstants::usePass, defaultValue)
00172       .toBool();
00173 }
00174 void QtPassSettings::setUsePass(const bool &usePass) {
00175   if (usePass) {
00176     QtPassSettings::pass = getRealPass();
00177   } else {
00178     QtPassSettings::pass = getImitatePass();
```

```
00179   }
00180   getInstance()->setValue(SettingsConstants::usePass, usePass);
00181 }
00182
00183 int QtPassSettings::getClipBoardTypeRaw(
00184     const Enums::clipBoardType &defaultvalue) {
00185   return getInstance()
00186       ->value(SettingsConstants::clipBoardType, static_cast<int>(defaultvalue))
00187       .toInt();
00188 }
00189
00190 Enums::clipBoardType
00191 QtPassSettings::getClipBoardType(const Enums::clipBoardType &defaultvalue) {
00192   return static_cast<Enums::clipBoardType>(getClipBoardTypeRaw(defaultvalue));
00193 }
00194 void QtPassSettings::setClipBoardType(const int &clipBoardType) {
00195   getInstance()->setValue(SettingsConstants::clipBoardType, clipBoardType);
00196 }
00197
00198 bool QtPassSettings::isUseSelection(const bool &defaultValue) {
00199   return getInstance()
00200       ->value(SettingsConstants::useSelection, defaultValue)
00201       .toBool();
00202 }
00203 void QtPassSettings::setUseSelection(const bool &useSelection) {
00204   getInstance()->setValue(SettingsConstants::useSelection, useSelection);
00205 }
00206
00207 bool QtPassSettings::isUseAutoclear(const bool &defaultValue) {
00208   return getInstance()
00209       ->value(SettingsConstants::useAutoclear, defaultValue)
00210       .toBool();
00211 }
00212 void QtPassSettings::setUseAutoclear(const bool &useAutoclear) {
00213   getInstance()->setValue(SettingsConstants::useAutoclear, useAutoclear);
00214 }
00215
00216 int QtPassSettings::getAutoclearSeconds(const int &defaultValue) {
00217   return getInstance()
00218       ->value(SettingsConstants::autoclearSeconds, defaultValue)
00219       .toInt();
00220 }
00221 void QtPassSettings::setAutoclearSeconds(const int &autoClearSeconds) {
00222   getInstance()->setValue(SettingsConstants::autoclearSeconds,
00223                           autoClearSeconds);
00224 }
00225
00226 bool QtPassSettings::isUseAutoclearPanel(const bool &defaultValue) {
00227   return getInstance()
00228       ->value(SettingsConstants::useAutoclearPanel, defaultValue)
00229       .toBool();
00230 }
00231 void QtPassSettings::setUseAutoclearPanel(const bool &useAutoclearPanel) {
00232   getInstance()->setValue(SettingsConstants::useAutoclearPanel,
00233                           useAutoclearPanel);
00234 }
00235
00236 int QtPassSettings::getAutoclearPanelSeconds(const int &defaultValue) {
00237   return getInstance()
00238       ->value(SettingsConstants::autoclearPanelSeconds, defaultValue)
00239       .toInt();
00240 }
00241 void QtPassSettings::setAutoclearPanelSeconds(
00242     const int &autoClearPanelSeconds) {
00243   getInstance()->setValue(SettingsConstants::autoclearPanelSeconds,
00244                           autoClearPanelSeconds);
00245 }
00246
00247 bool QtPassSettings::isHidePassword(const bool &defaultValue) {
00248   return getInstance()
00249       ->value(SettingsConstants::hidePassword, defaultValue)
00250       .toBool();
00251 }
00252 void QtPassSettings::setHidePassword(const bool &hidePassword) {
00253   getInstance()->setValue(SettingsConstants::hidePassword, hidePassword);
00254 }
00255
00256 bool QtPassSettings::isHideContent(const bool &defaultValue) {
00257   return getInstance()
00258       ->value(SettingsConstants::hideContent, defaultValue)
00259       .toBool();
00260 }
00261 void QtPassSettings::setHideContent(const bool &hideContent) {
00262   getInstance()->setValue(SettingsConstants::hideContent, hideContent);
00263 }
00264
00265 bool QtPassSettings::isUseMonospace(const bool &defaultValue) {
```

```
00266   return getInstance()
00267       ->value(SettingsConstants::useMonospace, defaultValue)
00268       .toBool();
00269 }
00270 void QtPassSettings::setUseMonospace(const bool &useMonospace) {
00271   getInstance()->setValue(SettingsConstants::useMonospace, useMonospace);
00272 }
00273
00274 bool QtPassSettings::isDisplayAsIs(const bool &defaultValue) {
00275   return getInstance()
00276       ->value(SettingsConstants::displayAsIs, defaultValue)
00277       .toBool();
00278 }
00279 void QtPassSettings::setDisplayAsIs(const bool &displayAsIs) {
00280   getInstance()->setValue(SettingsConstants::displayAsIs, displayAsIs);
00281 }
00282
00283 bool QtPassSettings::isNoLineWrapping(const bool &defaultValue) {
00284   return getInstance()
00285       ->value(SettingsConstants::noLineWrapping, defaultValue)
00286       .toBool();
00287 }
00288 void QtPassSettings::setNoLineWrapping(const bool &noLineWrapping) {
00289   getInstance()->setValue(SettingsConstants::noLineWrapping, noLineWrapping);
00290 }
00291
00292 bool QtPassSettings::isAddGPGId(const bool &defaultValue) {
00293   return getInstance()
00294       ->value(SettingsConstants::addGPGId, defaultValue)
00295       .toBool();
00296 }
00297 void QtPassSettings::setAddGPGId(const bool &addGPGId) {
00298   getInstance()->setValue(SettingsConstants::addGPGId, addGPGId);
00299 }
00300
00301 QString QtPassSettings::getPassStore(const QString &defaultValue) {
00302   QString returnValue = getInstance()
00303                             ->value(SettingsConstants::passStore, defaultValue)
00304                             .toString();
00305
00306   // Normalize the path string
00307   returnValue = QDir(returnValue).absolutePath();
00308
00309   // ensure directory exists if never used pass or misconfigured.
00310   // otherwise process->setWorkingDirectory(passStore); will fail on execution.
00311   if (!QDir(returnValue).exists()) {
00312     QDir().mkdir(returnValue);
00313   }
00314
00315   // ensure path ends in /
00316   if (!returnValue.endsWith("/") && !returnValue.endsWith(QDir::separator())) {
00317     returnValue += QDir::separator();
00318   }
00319
00320   return returnValue;
00321 }
00322 void QtPassSettings::setPassStore(const QString &passStore) {
00323   getInstance()->setValue(SettingsConstants::passStore, passStore);
00324 }
00325
00326 QString QtPassSettings::getPassSigningKey(const QString &defaultValue) {
00327   return getInstance()
00328       ->value(SettingsConstants::passSigningKey, defaultValue)
00329       .toString();
00330 }
00331 void QtPassSettings::setPassSigningKey(const QString &passSigningKey) {
00332   getInstance()->setValue(SettingsConstants::passSigningKey, passSigningKey);
00333 }
00334
00335 void QtPassSettings::initExecutables() {
00336   QString passExecutable =
00337       QtPassSettings::getPassExecutable(Util::findBinaryInPath("pass"));
00338   QtPassSettings::setPassExecutable(passExecutable);
00339
00340   QString gitExecutable =
00341       QtPassSettings::getGitExecutable(Util::findBinaryInPath("git"));
00342   QtPassSettings::setGitExecutable(gitExecutable);
00343
00344   QString gpgExecutable =
00345       QtPassSettings::getGpgExecutable(Util::findBinaryInPath("gpg2"));
00346   QtPassSettings::setGpgExecutable(gpgExecutable);
00347
00348   QString pwgenExecutable =
00349       QtPassSettings::getPwgenExecutable(Util::findBinaryInPath("pwgen"));
00350   QtPassSettings::setPwgenExecutable(pwgenExecutable);
00351 }
00352 QString QtPassSettings::getPassExecutable(const QString &defaultValue) {
```

```
00353   return getInstance()
00354       ->value(SettingsConstants::passExecutable, defaultValue)
00355       .toString();
00356 }
00357 void QtPassSettings::setPassExecutable(const QString &passExecutable) {
00358   getInstance()->setValue(SettingsConstants::passExecutable, passExecutable);
00359 }
00360
00361 QString QtPassSettings::getGitExecutable(const QString &defaultValue) {
00362   return getInstance()
00363       ->value(SettingsConstants::gitExecutable, defaultValue)
00364       .toString();
00365 }
00366 void QtPassSettings::setGitExecutable(const QString &gitExecutable) {
00367   getInstance()->setValue(SettingsConstants::gitExecutable, gitExecutable);
00368 }
00369
00370 QString QtPassSettings::getGpgExecutable(const QString &defaultValue) {
00371   return getInstance()
00372       ->value(SettingsConstants::gpgExecutable, defaultValue)
00373       .toString();
00374 }
00375 void QtPassSettings::setGpgExecutable(const QString &gpgExecutable) {
00376   getInstance()->setValue(SettingsConstants::gpgExecutable, gpgExecutable);
00377 }
00378
00379 QString QtPassSettings::getPwgenExecutable(const QString &defaultValue) {
00380   return getInstance()
00381       ->value(SettingsConstants::pwgenExecutable, defaultValue)
00382       .toString();
00383 }
00384 void QtPassSettings::setPwgenExecutable(const QString &pwgenExecutable) {
00385   getInstance()->setValue(SettingsConstants::pwgenExecutable, pwgenExecutable);
00386 }
00387
00388 QString QtPassSettings::getGpgHome(const QString &defaultValue) {
00389   return getInstance()
00390       ->value(SettingsConstants::gpgHome, defaultValue)
00391       .toString();
00392 }
00393
00394 bool QtPassSettings::isUseWebDav(const bool &defaultValue) {
00395   return getInstance()
00396       ->value(SettingsConstants::useWebDav, defaultValue)
00397       .toBool();
00398 }
00399 void QtPassSettings::setUseWebDav(const bool &useWebDav) {
00400   getInstance()->setValue(SettingsConstants::useWebDav, useWebDav);
00401 }
00402
00403 QString QtPassSettings::getWebDavUrl(const QString &defaultValue) {
00404   return getInstance()
00405       ->value(SettingsConstants::webDavUrl, defaultValue)
00406       .toString();
00407 }
00408 void QtPassSettings::setWebDavUrl(const QString &webDavUrl) {
00409   getInstance()->setValue(SettingsConstants::webDavUrl, webDavUrl);
00410 }
00411
00412 QString QtPassSettings::getWebDavUser(const QString &defaultValue) {
00413   return getInstance()
00414       ->value(SettingsConstants::webDavUser, defaultValue)
00415       .toString();
00416 }
00417 void QtPassSettings::setWebDavUser(const QString &webDavUser) {
00418   getInstance()->setValue(SettingsConstants::webDavUser, webDavUser);
00419 }
00420
00421 QString QtPassSettings::getWebDavPassword(const QString &defaultValue) {
00422   return getInstance()
00423       ->value(SettingsConstants::webDavPassword, defaultValue)
00424       .toString();
00425 }
00426 void QtPassSettings::setWebDavPassword(const QString &webDavPassword) {
00427   getInstance()->setValue(SettingsConstants::webDavPassword, webDavPassword);
00428 }
00429
00430 QString QtPassSettings::getProfile(const QString &defaultValue) {
00431   return getInstance()
00432       ->value(SettingsConstants::profile, defaultValue)
00433       .toString();
00434 }
00435 void QtPassSettings::setProfile(const QString &profile) {
00436   getInstance()->setValue(SettingsConstants::profile, profile);
00437 }
00438
00439 bool QtPassSettings::isUseGit(const bool &defaultValue) {
```

```
00440   return getInstance()->value(SettingsConstants::useGit, defaultValue).toBool();
00441 }
00442 void QtPassSettings::setUseGit(const bool &useGit) {
00443   getInstance()->setValue(SettingsConstants::useGit, useGit);
00444 }
00445
00446 bool QtPassSettings::isUseOtp(const bool &defaultValue) {
00447   return getInstance()->value(SettingsConstants::useOtp, defaultValue).toBool();
00448 }
00449
00450 void QtPassSettings::setUseOtp(const bool &useOtp) {
00451   getInstance()->setValue(SettingsConstants::useOtp, useOtp);
00452 }
00453
00454 bool QtPassSettings::isUseQrencode(const bool &defaultValue) {
00455   return getInstance()
00456       ->value(SettingsConstants::useQrencode, defaultValue)
00457       .toBool();
00458 }
00459
00460 void QtPassSettings::setUseQrencode(const bool &useQrencode) {
00461   getInstance()->setValue(SettingsConstants::useQrencode, useQrencode);
00462 }
00463
00464 QString QtPassSettings::getQrencodeExecutable(const QString &defaultValue) {
00465   return getInstance()
00466       ->value(SettingsConstants::qrencodeExecutable, defaultValue)
00467       .toString();
00468 }
00469 void QtPassSettings::setQrencodeExecutable(const QString &qrencodeExecutable) {
00470   getInstance()->setValue(SettingsConstants::qrencodeExecutable,
00471                           qrencodeExecutable);
00472 }
00473
00474 bool QtPassSettings::isUsePwgen(const bool &defaultValue) {
00475   return getInstance()
00476       ->value(SettingsConstants::usePwgen, defaultValue)
00477       .toBool();
00478 }
00479 void QtPassSettings::setUsePwgen(const bool &usePwgen) {
00480   getInstance()->setValue(SettingsConstants::usePwgen, usePwgen);
00481 }
00482
00483 bool QtPassSettings::isAvoidCapitals(const bool &defaultValue) {
00484   return getInstance()
00485       ->value(SettingsConstants::avoidCapitals, defaultValue)
00486       .toBool();
00487 }
00488 void QtPassSettings::setAvoidCapitals(const bool &avoidCapitals) {
00489   getInstance()->setValue(SettingsConstants::avoidCapitals, avoidCapitals);
00490 }
00491
00492 bool QtPassSettings::isAvoidNumbers(const bool &defaultValue) {
00493   return getInstance()
00494       ->value(SettingsConstants::avoidNumbers, defaultValue)
00495       .toBool();
00496 }
00497 void QtPassSettings::setAvoidNumbers(const bool &avoidNumbers) {
00498   getInstance()->setValue(SettingsConstants::avoidNumbers, avoidNumbers);
00499 }
00500
00501 bool QtPassSettings::isLessRandom(const bool &defaultValue) {
00502   return getInstance()
00503       ->value(SettingsConstants::lessRandom, defaultValue)
00504       .toBool();
00505 }
00506 void QtPassSettings::setLessRandom(const bool &lessRandom) {
00507   getInstance()->setValue(SettingsConstants::lessRandom, lessRandom);
00508 }
00509
00510 bool QtPassSettings::isUseSymbols(const bool &defaultValue) {
00511   return getInstance()
00512       ->value(SettingsConstants::useSymbols, defaultValue)
00513       .toBool();
00514 }
00515 void QtPassSettings::setUseSymbols(const bool &useSymbols) {
00516   getInstance()->setValue(SettingsConstants::useSymbols, useSymbols);
00517 }
00518
00519 void QtPassSettings::setPasswordLength(const int &passwordLength) {
00520   getInstance()->setValue(SettingsConstants::passwordLength, passwordLength);
00521 }
00522 void QtPassSettings::setPasswordCharsselection(
00523     const int &passwordCharsselection) {
00524   getInstance()->setValue(SettingsConstants::passwordCharsselection,
00525                           passwordCharsselection);
00526 }
```

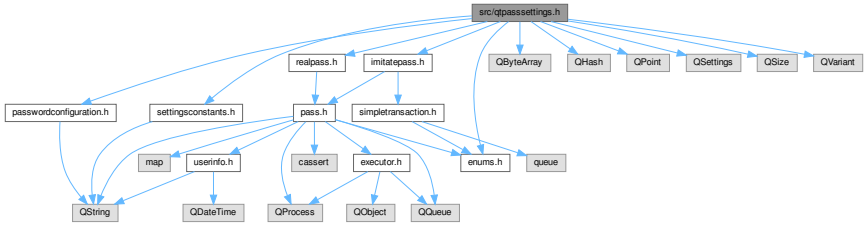
```
00527 void QtPassSettings::setPasswordChars(const QString &passwordChars) {
00528   getInstance()->setValue(SettingsConstants::passwordChars, passwordChars);
00529 }
00530
00531 bool QtPassSettings::isUseTrayIcon(const bool &defaultValue) {
00532   return getInstance()
00533       ->value(SettingsConstants::useTrayIcon, defaultValue)
00534       .toBool();
00535 }
00536 void QtPassSettings::setUseTrayIcon(const bool &useTrayIcon) {
00537   getInstance()->setValue(SettingsConstants::useTrayIcon, useTrayIcon);
00538 }
00539
00540 bool QtPassSettings::isHideOnClose(const bool &defaultValue) {
00541   return getInstance()
00542       ->value(SettingsConstants::hideOnClose, defaultValue)
00543       .toBool();
00544 }
00545 void QtPassSettings::setHideOnClose(const bool &hideOnClose) {
00546   getInstance()->setValue(SettingsConstants::hideOnClose, hideOnClose);
00547 }
00548
00549 bool QtPassSettings::isStartMinimized(const bool &defaultValue) {
00550   return getInstance()
00551       ->value(SettingsConstants::startMinimized, defaultValue)
00552       .toBool();
00553 }
00554 void QtPassSettings::setStartMinimized(const bool &startMinimized) {
00555   getInstance()->setValue(SettingsConstants::startMinimized, startMinimized);
00556 }
00557
00558 bool QtPassSettings::isAlwaysOnTop(const bool &defaultValue) {
00559   return getInstance()
00560       ->value(SettingsConstants::alwaysOnTop, defaultValue)
00561       .toBool();
00562 }
00563 void QtPassSettings::setAlwaysOnTop(const bool &alwaysOnTop) {
00564   getInstance()->setValue(SettingsConstants::alwaysOnTop, alwaysOnTop);
00565 }
00566
00567 bool QtPassSettings::isAutoPull(const bool &defaultValue) {
00568   return getInstance()
00569       ->value(SettingsConstants::autoPull, defaultValue)
00570       .toBool();
00571 }
00572 void QtPassSettings::setAutoPull(const bool &autoPull) {
00573   getInstance()->setValue(SettingsConstants::autoPull, autoPull);
00574 }
00575
00576 bool QtPassSettings::isAutoPush(const bool &defaultValue) {
00577   return getInstance()
00578       ->value(SettingsConstants::autoPush, defaultValue)
00579       .toBool();
00580 }
00581 void QtPassSettings::setAutoPush(const bool &autoPush) {
00582   getInstance()->setValue(SettingsConstants::autoPush, autoPush);
00583 }
00584
00585 QString QtPassSettings::getPassTemplate(const QString &defaultValue) {
00586   return getInstance()
00587       ->value(SettingsConstants::passTemplate, defaultValue)
00588       .toString();
00589 }
00590 void QtPassSettings::setPassTemplate(const QString &passTemplate) {
00591   getInstance()->setValue(SettingsConstants::passTemplate, passTemplate);
00592 }
00593
00594 bool QtPassSettings::isUseTemplate(const bool &defaultValue) {
00595   return getInstance()
00596       ->value(SettingsConstants::useTemplate, defaultValue)
00597       .toBool();
00598 }
00599 void QtPassSettings::setUseTemplate(const bool &useTemplate) {
00600   getInstance()->setValue(SettingsConstants::useTemplate, useTemplate);
00601 }
00602
00603 bool QtPassSettings::isTemplateAllFields(const bool &defaultValue) {
00604   return getInstance()
00605       ->value(SettingsConstants::templateAllFields, defaultValue)
00606       .toBool();
00607 }
00608 void QtPassSettings::setTemplateAllFields(const bool &templateAllFields) {
00609   getInstance()->setValue(SettingsConstants::templateAllFields,
00610                           templateAllFields);
00611 }
00612
00613 RealPass *QtPassSettings::getRealPass() {
```

```
00614   if (realPass.isNull())
00615     realPass.reset(new RealPass());
00616   return realPass.data();
00617 }
00618 ImitatePass *QtPassSettings::getImitatePass() {
00619   if (imitatePass.isNull())
00620     imitatePass.reset(new ImitatePass());
00621   return imitatePass.data();
00622 }
```

14.70 src/qtpasssettings.h File Reference

```
#include "enums.h"
#include "imitatepass.h"
#include "passwordconfiguration.h"
#include "realpass.h"
#include "settingsconstants.h"
#include <QByteArray>
#include <QHash>
#include <QPoint>
#include <QSettings>
#include <QSize>
#include <QVariant>
```

Include dependency graph for qtpasssettings.h:



This graph shows which files directly or indirectly include this file:

Classes

- class QtPassSettings

  Singleton that stores qtpass' settings, saves and loads config.

14.71 qtpasssettings.h

Go to the documentation of this file.

```
00001 #ifndef QTPASSSETTINGS_H
00002 #define QTPASSSETTINGS_H
```

```
00003
00004 #include "enums.h"
00005 #include "imitatepass.h"
00006 #include "passwordconfiguration.h"
00007 #include "realpass.h"
00008 #include "settingsconstants.h"
00009
00010 #include <QByteArray>
00011 #include <QHash>
00012 #include <QPoint>
00013 #include <QSettings>
00014 #include <QSize>
00015 #include <QVariant>
00016
00021 class QtPassSettings : public QSettings {
00022 private:
00023   explicit QtPassSettings();
00024
00025   QtPassSettings(const QString &organization, const QSettings::Format format)
00026       : QSettings(organization, format) {}
00027   QtPassSettings(const QString &organization, const QString &application)
00028       : QSettings(organization, application) {}
00029
00030   virtual ~QtPassSettings() {}
00031
00032   static bool initialized;
00033   static QtPassSettings *m_instance;
00034
00035   static Pass *pass;
00036   static QScopedPointer<RealPass> realPass;
00037   static QScopedPointer<ImitatePass> imitatePass;
00038
00039 public:
00040   static QtPassSettings *getInstance();
00041
00042   static QString
00043   getVersion(const QString &defaultValue = QVariant().toString());
00044   static void setVersion(const QString &version);
00045
00046   static QByteArray
00047   getGeometry(const QByteArray &defaultValue = QVariant().toByteArray());
00048   static void setGeometry(const QByteArray &geometry);
00049
00050   static QByteArray
00051   getSavestate(const QByteArray &defaultValue = QVariant().toByteArray());
00052   static void setSavestate(const QByteArray &saveState);
00053
00054   static QPoint getPos(const QPoint &defaultValue = QVariant().toPoint());
00055   static void setPos(const QPoint &pos);
00056
00057   static QSize getSize(const QSize &defaultValue = QVariant().toSize());
00058   static void setSize(const QSize &size);
00059
00060   static bool isMaximized(const bool &defaultValue = QVariant().toBool());
00061   static void setMaximized(const bool &maximized);
00062
00063   static bool isUsePass(const bool &defaultValue = QVariant().toBool());
00064   static void setUsePass(const bool &usePass);
00065
00066   static int getClipBoardTypeRaw(
00067       const Enums::clipBoardType &defaultvalue = Enums::CLIPBOARD_NEVER);
00068   static Enums::clipBoardType getClipBoardType(
00069       const Enums::clipBoardType &defaultvalue = Enums::CLIPBOARD_NEVER);
00070   static void setClipBoardType(const int &clipBoardType);
00071
00072   static bool isUseSelection(const bool &defaultValue = QVariant().toBool());
00073   static void setUseSelection(const bool &useSelection);
00074
00075   static bool isUseAutoclear(const bool &defaultValue = QVariant().toBool());
00076   static void setUseAutoclear(const bool &useAutoclear);
00077
00078   static int getAutoclearSeconds(const int &defaultValue = QVariant().toInt());
00079   static void setAutoclearSeconds(const int &autoClearSeconds);
00080
00081   static bool
00082   isUseAutoclearPanel(const bool &defaultValue = QVariant().toBool());
00083   static void setUseAutoclearPanel(const bool &useAutoclearPanel);
00084
00085   static int
00086   getAutoclearPanelSeconds(const int &defaultValue = QVariant().toInt());
00087   static void setAutoclearPanelSeconds(const int &autoClearPanelSeconds);
00088
00089   static bool isHidePassword(const bool &defaultValue = QVariant().toBool());
00090   static void setHidePassword(const bool &hidePassword);
00091
00092   static bool isHideContent(const bool &defaultValue = QVariant().toBool());
00093   static void setHideContent(const bool &hideContent);
```

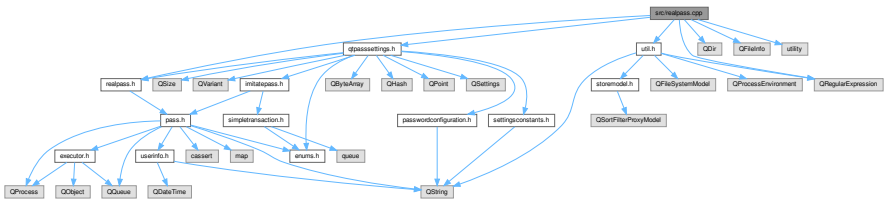
```
00094
00095   static bool isUseMonospace(const bool &defaultValue = QVariant().toBool());
00096   static void setUseMonospace(const bool &useMonospace);
00097
00098   static bool isDisplayAsIs(const bool &defaultValue = QVariant().toBool());
00099   static void setDisplayAsIs(const bool &displayAsIs);
00100
00101   static bool isNoLineWrapping(const bool &defaultValue = QVariant().toBool());
00102   static void setNoLineWrapping(const bool &noLineWrapping);
00103
00104   static bool isAddGPGId(const bool &defaultValue = QVariant().toBool());
00105   static void setAddGPGId(const bool &addGPGId);
00106
00107   static QString
00108   getPassStore(const QString &defaultValue = QVariant().toString());
00109   static void setPassStore(const QString &passStore);
00110
00111   static QString
00112   getPassSigningKey(const QString &defaultValue = QVariant().toString());
00113   static void setPassSigningKey(const QString &passSigningKey);
00114
00115   static void initExecutables();
00116   static QString
00117   getPassExecutable(const QString &defaultValue = QVariant().toString());
00118   static void setPassExecutable(const QString &passExecutable);
00119
00120   static QString
00121   getGitExecutable(const QString &defaultValue = QVariant().toString());
00122   static void setGitExecutable(const QString &gitExecutable);
00123
00124   static QString
00125   getGpgExecutable(const QString &defaultValue = QVariant().toString());
00126   static void setGpgExecutable(const QString &gpgExecutable);
00127
00128   static QString
00129   getPwgenExecutable(const QString &defaultValue = QVariant().toString());
00130   static void setPwgenExecutable(const QString &pwgenExecutable);
00131
00132   static QString
00133   getGpgHome(const QString &defaultValue = QVariant().toString());
00134
00135   static bool isUseWebDav(const bool &defaultValue = QVariant().toBool());
00136   static void setUseWebDav(const bool &useWebDav);
00137
00138   static QString
00139   getWebDavUrl(const QString &defaultValue = QVariant().toString());
00140   static void setWebDavUrl(const QString &webDavUrl);
00141
00142   static QString
00143   getWebDavUser(const QString &defaultValue = QVariant().toString());
00144   static void setWebDavUser(const QString &webDavUser);
00145
00146   static QString
00147   getWebDavPassword(const QString &defaultValue = QVariant().toString());
00148   static void setWebDavPassword(const QString &webDavPassword);
00149
00150   static QString
00151   getProfile(const QString &defaultValue = QVariant().toString());
00152   static void setProfile(const QString &profile);
00153
00154   static bool isUseGit(const bool &defaultValue = QVariant().toBool());
00155   static void setUseGit(const bool &useGit);
00156
00157   static bool isUseOtp(const bool &defaultValue = QVariant().toBool());
00158   static void setUseOtp(const bool &useOtp);
00159
00160   static bool isUseQrencode(const bool &defaultValue = QVariant().toBool());
00161   static void setUseQrencode(const bool &useQrencode);
00162
00163   static QString
00164   getQrencodeExecutable(const QString &defaultValue = QVariant().toString());
00165   static void setQrencodeExecutable(const QString &qrencodeExecutable);
00166
00167   static bool isUsePwgen(const bool &defaultValue = QVariant().toBool());
00168   static void setUsePwgen(const bool &usePwgen);
00169
00170   static bool isAvoidCapitals(const bool &defaultValue = QVariant().toBool());
00171   static void setAvoidCapitals(const bool &avoidCapitals);
00172
00173   static bool isAvoidNumbers(const bool &defaultValue = QVariant().toBool());
00174   static void setAvoidNumbers(const bool &avoidNumbers);
00175
00176   static bool isLessRandom(const bool &defaultValue = QVariant().toBool());
00177   static void setLessRandom(const bool &lessRandom);
00178
00179   static bool isUseSymbols(const bool &defaultValue = QVariant().toBool());
00180   static void setUseSymbols(const bool &useSymbols);
```

```
00181
00182   static PasswordConfiguration getPasswordConfiguration();
00183   static void setPasswordConfiguration(const PasswordConfiguration &config);
00184   static void setPasswordLength(const int &passwordLength);
00185   static void setPasswordCharsselection(const int &passwordCharsselection);
00186   static void setPasswordChars(const QString &passwordChars);
00187
00188   static bool isUseTrayIcon(const bool &defaultValue = QVariant().toBool());
00189   static void setUseTrayIcon(const bool &useTrayIcon);
00190
00191   static bool isHideOnClose(const bool &defaultValue = QVariant().toBool());
00192   static void setHideOnClose(const bool &hideOnClose);
00193
00194   static bool isStartMinimized(const bool &defaultValue = QVariant().toBool());
00195   static void setStartMinimized(const bool &startMinimized);
00196
00197   static bool isAlwaysOnTop(const bool &defaultValue = QVariant().toBool());
00198   static void setAlwaysOnTop(const bool &alwaysOnTop);
00199
00200   static bool isAutoPull(const bool &defaultValue = QVariant().toBool());
00201   static void setAutoPull(const bool &autoPull);
00202
00203   static bool isAutoPush(const bool &defaultValue = QVariant().toBool());
00204   static void setAutoPush(const bool &autoPush);
00205
00206   static QString
00207   getPassTemplate(const QString &defaultValue = QVariant().toString());
00208   static void setPassTemplate(const QString &passTemplate);
00209
00210   static bool isUseTemplate(const bool &defaultValue = QVariant().toBool());
00211   static void setUseTemplate(const bool &useTemplate);
00212
00213   static bool
00214   isTemplateAllFields(const bool &defaultValue = QVariant().toBool());
00215   static void setTemplateAllFields(const bool &templateAllFields);
00216
00217   static QHash<QString, QHash<QString, QString» getProfiles();
00218   static void
00219   setProfiles(const QHash<QString, QHash<QString, QString» &profiles);
00220
00221   static Pass *getPass();
00222   static RealPass *getRealPass();
00223   static ImitatePass *getImitatePass();
00224 };
00225
00226 #endif // QTPASSSETTINGS_H
```

14.72 src/realpass.cpp File Reference

```
#include "realpass.h"
#include "qtpasssettings.h"
#include "util.h"
#include <QDir>
#include <QFileInfo>
#include <QRegularExpression>
#include <utility>
```

Include dependency graph for realpass.cpp:



14.73 realpass.cpp

Go to the documentation of this file.

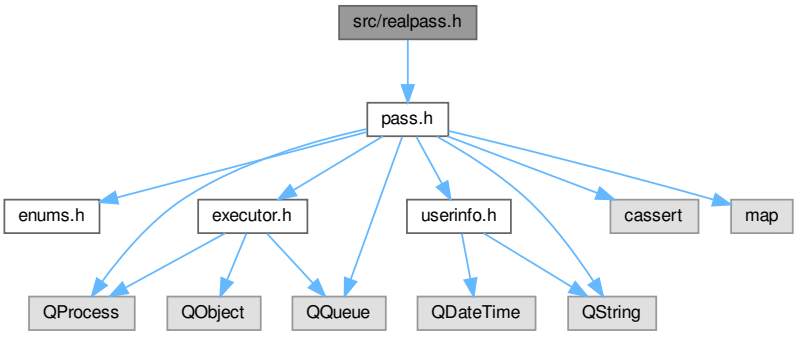
```
00001 #include "realpass.h"
00002 #include "qtpasssettings.h"
00003 #include "util.h"
00004
00005 #include <QDir>
00006 #include <QFileInfo>
00007 #include <QRegularExpression>
00008 #include <utility>
00009
00010 using namespace Enums;
00011
00012 RealPass::RealPass() = default;
00013
00017 void RealPass::GitInit() { executePass(GIT_INIT, {"git", "init"}); }
00018
00023 void RealPass::GitPull_b() {
00024   exec.executeBlocking(QtPassSettings::getPassExecutable(), {"git", "pull"});
00025 }
00026
00030 void RealPass::GitPull() { executePass(GIT_PULL, {"git", "pull"}); }
00031
00035 void RealPass::GitPush() { executePass(GIT_PUSH, {"git", "push"}); }
00036
00046 void RealPass::Show(QString file) {
00047   executePass(PASS_SHOW, {"show", file}, "", true);
00048 }
00049
00054 void RealPass::OtpGenerate(QString file) {
00055   executePass(PASS_OTP_GENERATE, {"otp", file}, "", true);
00056 }
00057
00061 void RealPass::Insert(QString file, QString newValue, bool overwrite) {
00062   QStringList args = {"insert", "-m"};
00063   if (overwrite)
00064     args.append("-f");
00065   args.append(file);
00066   executePass(PASS_INSERT, args, newValue);
00067 }
00068
00072 void RealPass::Remove(QString file, bool isDir) {
00073   executePass(PASS_REMOVE, {"rm", (isDir ? "-rf" : "-f"), file});
00074 }
00075
00082 void RealPass::Init(QString path, const QList<UserInfo> &users) {
00083   // remove the passStore directory otherwise,
00084   // pass would create a passStore/passStore/dir
00085   // but you want passStore/dir
00086   QString dirWithoutPassdir =
00087       path.remove(0, QtPassSettings::getPassStore().size());
00088   QStringList args = {"init", "--path=" + dirWithoutPassdir};
00089   foreach (const UserInfo &user, users) {
00090     if (user.enabled)
00091       args.append(user.key_id);
00092   }
00093   executePass(PASS_INIT, args);
00094 }
00095
00102 void RealPass::Move(const QString src, const QString dest, const bool force) {
00103   QFileInfo srcFileInfo = QFileInfo(src);
00104   QFileInfo destFileInfo = QFileInfo(dest);
00105
00106   // force mode?
00107   // pass uses always the force mode, when call from eg. QT. so we have to check
00108   // if this are to files
00109   // and the user didnt want to move force
00110   if (!force && srcFileInfo.isFile() && destFileInfo.isFile()) {
00111     return;
00112   }
00113
00114   QString passSrc = QDir(QtPassSettings::getPassStore())
00115                         .relativeFilePath(QDir(src).absolutePath());
00116   QString passDest = QDir(QtPassSettings::getPassStore())
00117                          .relativeFilePath(QDir(dest).absolutePath());
00118
00119   // remove the .gpg because pass will not work
00120   if (srcFileInfo.isFile() && srcFileInfo.suffix() == "gpg") {
00121     passSrc.replace(Util::endsWithGpg(), "");
00122   }
00123   if (destFileInfo.isFile() && destFileInfo.suffix() == "gpg") {
00124     passDest.replace(Util::endsWithGpg(), "");
00125   }
00126
00127   QStringList args;
00128   args << "mv";
00129   if (force) {
00130     args << "-f";
00131   }
```

```
00132   args « passSrc;
00133   args « passDest;
00134   executePass(PASS_MOVE, args);
00135 }
00136
00143 void RealPass::Copy(const QString src, const QString dest, const bool force) {
00144   QFileInfo srcFileInfo = QFileInfo(src);
00145   QFileInfo destFileInfo = QFileInfo(dest);
00146   // force mode?
00147   // pass uses always the force mode, when call from eg. QT. so we have to check
00148   // if this are to files
00149   // and the user didnt want to move force
00150   if (!force && srcFileInfo.isFile() && destFileInfo.isFile()) {
00151     return;
00152   }
00153
00154   QString passSrc = QDir(QtPassSettings::getPassStore())
00155                         .relativeFilePath(QDir(src).absolutePath());
00156   QString passDest = QDir(QtPassSettings::getPassStore())
00157                          .relativeFilePath(QDir(dest).absolutePath());
00158
00159   // remove the .gpg because pass will not work
00160   if (srcFileInfo.isFile() && srcFileInfo.suffix() == "gpg") {
00161     passSrc.replace(Util::endsWithGpg(), "");
00162   }
00163   if (destFileInfo.isFile() && destFileInfo.suffix() == "gpg") {
00164     passDest.replace(Util::endsWithGpg(), "");
00165   }
00166   QStringList args;
00167   args « "cp";
00168   if (force) {
00169     args « "-f";
00170   }
00171   args « passSrc;
00172   args « passDest;
00173   executePass(PASS_COPY, args);
00174 }
00175
00180 void RealPass::executePass(PROCESS id, const QStringList &args, QString input,
00181                            bool readStdout, bool readStderr) {
00182   executeWrapper(id, QtPassSettings::getPassExecutable(), args,
00183                  std::move(input), readStdout, readStderr);
00184 }
```

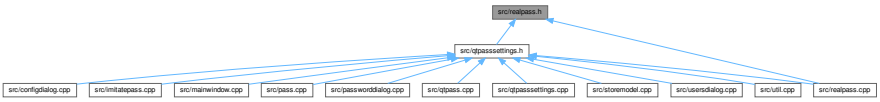
14.74 src/realpass.h File Reference

`#include "pass.h"`

Include dependency graph for realpass.h:



This graph shows which files directly or indirectly include this file:



Classes

- class RealPass

  Wrapper for executing pass to handle the password-store.

14.75 realpass.h

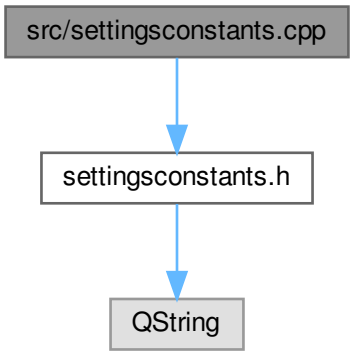
Go to the documentation of this file.

```
00001 #ifndef REALPASS_H
00002 #define REALPASS_H
00003
00004 #include "pass.h"
00005
00010 class RealPass : public Pass {
00011   void executePass(PROCESS id, const QStringList &args,
00012                    QString input = QString(), bool readStdout = true,
00013                    bool readStderr = true);
00014
00015 public:
00016   RealPass();
00017   virtual ~RealPass() {}
00018   virtual void GitInit() Q_DECL_OVERRIDE;
00019   virtual void GitPull() Q_DECL_OVERRIDE;
00020   virtual void GitPull_b() Q_DECL_OVERRIDE;
00021   virtual void GitPush() Q_DECL_OVERRIDE;
00022   virtual void Show(QString file) Q_DECL_OVERRIDE;
00023   virtual void OtpGenerate(QString file) Q_DECL_OVERRIDE;
00024   virtual void Insert(QString file, QString newValue,
00025                       bool overwrite = false) Q_DECL_OVERRIDE;
00026   virtual void Remove(QString file, bool isDir = false) Q_DECL_OVERRIDE;
00027   virtual void Init(QString path, const QList<UserInfo> &users) Q_DECL_OVERRIDE;
00028
00029   // Pass interface
00030 public:
00031   void Move(const QString src, const QString dest,
00032             const bool force = false) Q_DECL_OVERRIDE;
00033   void Copy(const QString src, const QString dest,
00034             const bool force = false) Q_DECL_OVERRIDE;
00035 };
00036
00037 #endif // REALPASS_H
```

14.76 src/settingsconstants.cpp File Reference

```
#include "settingsconstants.h"
```

Include dependency graph for settingsconstants.cpp:



14.77 settingsconstants.cpp

Go to the documentation of this file.

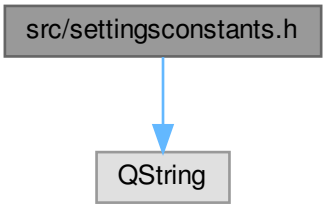
```
00001 #include "settingsconstants.h"
00002
00003 SettingsConstants::SettingsConstants() = default;
00004
00005 const QString SettingsConstants::version = "version";
00006 const QString SettingsConstants::groupMainwindow = "mainwindow";
00007 const QString SettingsConstants::geometry =
00008     SettingsConstants::groupMainwindow + "/geometry";
00009 const QString SettingsConstants::savestate =
00010     SettingsConstants::groupMainwindow + "/savestate";
00011 const QString SettingsConstants::pos =
00012     SettingsConstants::groupMainwindow + "/pos";
00013 const QString SettingsConstants::size =
00014     SettingsConstants::groupMainwindow + "/size";
00015 const QString SettingsConstants::splitterLeft =
00016     SettingsConstants::groupMainwindow + "/splitterLeft";
00017 const QString SettingsConstants::splitterRight =
00018     SettingsConstants::groupMainwindow + "/splitterRight";
00019 const QString SettingsConstants::maximized =
00020     SettingsConstants::groupMainwindow + "/maximized";
00021 const QString SettingsConstants::usePass = "usePass";
00022 const QString SettingsConstants::useSelection = "useSelection";
00023 const QString SettingsConstants::useAutoclear = "useAutoclear";
00024 const QString SettingsConstants::autoclearSeconds = "autoclearSeconds";
00025 const QString SettingsConstants::useAutoclearPanel = "useAutoclearPanel";
00026 const QString SettingsConstants::autoclearPanelSeconds =
00027     "autoclearPanelSeconds";
00028 const QString SettingsConstants::hidePassword = "hidePassword";
00029 const QString SettingsConstants::hideContent = "hideContent";
00030 const QString SettingsConstants::useMonospace = "useMonospace";
00031 const QString SettingsConstants::displayAsIs = "displayAsIs";
00032 const QString SettingsConstants::noLineWrapping = "noLineWrapping";
00033 const QString SettingsConstants::addGPGId = "addGPGId";
00034 const QString SettingsConstants::passStore = "passStore";
00035 const QString SettingsConstants::passSigningKey = "passSigningKey";
00036 const QString SettingsConstants::passExecutable = "passExecutable";
00037 const QString SettingsConstants::gitExecutable = "gitExecutable";
00038 const QString SettingsConstants::gpgExecutable = "gpgExecutable";
00039 const QString SettingsConstants::pwgenExecutable = "pwgenExecutable";
00040 const QString SettingsConstants::gpgHome = "gpgHome";
00041 const QString SettingsConstants::useWebDav = "useWebDav";
```

```
00042 const QString SettingsConstants::webDavUrl = "webDavUrl";
00043 const QString SettingsConstants::webDavUser = "webDavUser";
00044 const QString SettingsConstants::webDavPassword = "webDavPassword";
00045 const QString SettingsConstants::profile = "profile";
00046 const QString SettingsConstants::groupProfiles = "profiles";
00047 const QString SettingsConstants::useGit = "useGit";
00048 const QString SettingsConstants::useOtp = "useOtp";
00049 const QString SettingsConstants::useQrencode = "useQrencode";
00050 const QString SettingsConstants::qrencodeExecutable = "qrencodeExecutable";
00051 const QString SettingsConstants::useClipboard = "useClipboard";
00052 const QString SettingsConstants::usePwgen = "usePwgen";
00053 const QString SettingsConstants::avoidCapitals = "avoidCapitals";
00054 const QString SettingsConstants::avoidNumbers = "avoidNumbers";
00055 const QString SettingsConstants::lessRandom = "lessRandom";
00056 const QString SettingsConstants::useSymbols = "useSymbols";
00057 const QString SettingsConstants::passwordLength = "passwordLength";
00058 const QString SettingsConstants::passwordCharsselection =
00059     "passwordCharsselection";
00060 const QString SettingsConstants::passwordChars = "passwordChars";
00061 const QString SettingsConstants::useTrayIcon = "useTrayIcon";
00062 const QString SettingsConstants::hideOnClose = "hideOnClose";
00063 const QString SettingsConstants::startMinimized = "startMinimized";
00064 const QString SettingsConstants::alwaysOnTop = "alwaysOnTop";
00065 const QString SettingsConstants::autoPull = "autoPull";
00066 const QString SettingsConstants::autoPush = "autoPush";
00067 const QString SettingsConstants::passTemplate = "passTemplate";
00068 const QString SettingsConstants::useTemplate = "useTemplate";
00069 const QString SettingsConstants::templateAllFields = "templateAllFields";
00070 const QString SettingsConstants::clipBoardType = "clipBoardType";
```

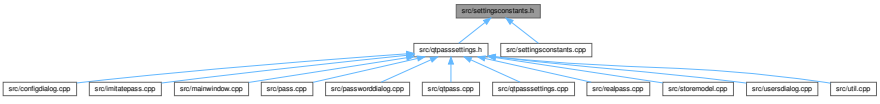
14.78 src/settingsconstants.h File Reference

`#include <QString>`

Include dependency graph for settingsconstants.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SettingsConstants

  Table for the naming of configuration items.

14.79 settingsconstants.h

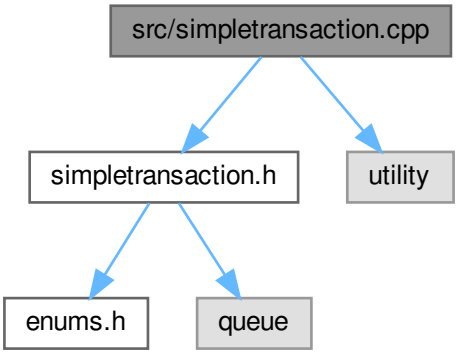
Go to the documentation of this file.

```
00001 #ifndef CONSTANTS_H
00002 #define CONSTANTS_H
00003
00004 #include <QString>
00005
00010 class SettingsConstants {
00011 public:
00012   const static QString version;
00013   const static QString groupMainwindow;
00014   const static QString geometry;
00015   const static QString savestate;
00016   const static QString pos;
00017   const static QString size;
00018   const static QString splitterLeft;
00019   const static QString splitterRight;
00020   const static QString maximized;
00021   const static QString usePass;
00022   const static QString useAutoclear;
00023   const static QString useSelection;
00024   const static QString autoclearSeconds;
00025   const static QString useAutoclearPanel;
00026   const static QString autoclearPanelSeconds;
00027   const static QString hidePassword;
00028   const static QString hideContent;
00029   const static QString useMonospace;
00030   const static QString displayAsIs;
00031   const static QString noLineWrapping;
00032   const static QString addGPGId;
00033   const static QString passStore;
00034   const static QString passSigningKey;
00035   const static QString passExecutable;
00036   const static QString gitExecutable;
00037   const static QString gpgExecutable;
00038   const static QString pwgenExecutable;
00039   const static QString gpgHome;
00040   const static QString useWebDav;
00041   const static QString webDavUrl;
00042   const static QString webDavUser;
00043   const static QString webDavPassword;
00044   const static QString profile;
00045   const static QString groupProfiles;
00046   const static QString useGit;
00047   const static QString useOtp;
00048   const static QString useQrencode;
00049   const static QString qrencodeExecutable;
00050   const static QString useClipboard;
00051   const static QString usePwgen;
00052   const static QString avoidCapitals;
00053   const static QString avoidNumbers;
00054   const static QString lessRandom;
00055   const static QString useSymbols;
00056   const static QString passwordLength;
00057   const static QString passwordCharsselection;
00058   const static QString passwordChars;
00059   const static QString useTrayIcon;
00060   const static QString hideOnClose;
00061   const static QString startMinimized;
00062   const static QString alwaysOnTop;
00063   const static QString autoPull;
00064   const static QString autoPush;
00065   const static QString passTemplate;
00066   const static QString useTemplate;
00067   const static QString templateAllFields;
00068   const static QString clipBoardType;
00069
00070 private:
00071   explicit SettingsConstants();
00072 };
00073
00074 #endif // CONSTANTS_H
```

14.80 src/simpletransaction.cpp File Reference

```
#include "simpletransaction.h"
#include <utility>
```

Include dependency graph for simpletransaction.cpp:



14.81 simpletransaction.cpp

Go to the documentation of this file.

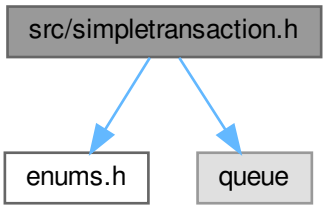
```
00001 #include "simpletransaction.h"
00002 #include <utility>
00003
00004 #ifdef QT_DEBUG
00005 #include "debughelper.h"
00006 #endif
00007
00008 using std::pair;
00009 using namespace Enums;
00010
00014 void simpleTransaction::transactionStart() {
00015 #ifdef QT_DEBUG
00016   dbg() « "START" « transactionDepth;
00017 #endif
00018   transactionDepth++;
00019 }
00020
00025 void simpleTransaction::transactionAdd(PROCESS id) {
00026 #ifdef QT_DEBUG
00027   dbg() « "ADD" « transactionDepth « id;
00028 #endif
00029   if (transactionDepth > 0) {
00030     lastInTransaction = id;
00031   } else {
00032     transactionQueue.push(pair<PROCESS, PROCESS>(id, id));
00033   }
00034 }
00035
00040 void simpleTransaction::transactionEnd(PROCESS pid) {
00041 #ifdef QT_DEBUG
00042   dbg() « "END" « transactionDepth;
00043 #endif
00044   if (transactionDepth > 0) {
00045     transactionDepth--;
00046     if (transactionDepth == 0 && lastInTransaction != INVALID) {
00047       transactionQueue.push(pair<PROCESS, PROCESS>(lastInTransaction, pid));
00048       lastInTransaction = INVALID;
00049     }
00050   }
00051 }
00052
00058 PROCESS simpleTransaction::transactionIsOver(PROCESS id) {
00059 #ifdef QT_DEBUG
00060   dbg() « "OVER" « transactionDepth « id;
00061 #endif
00062   if (!transactionQueue.empty() && id == transactionQueue.front().first) {
00063     PROCESS ret = transactionQueue.front().second;
```

```
00064     transactionQueue.pop();
00065     return ret;
00066   }
00067   return INVALID;
00068 }
```

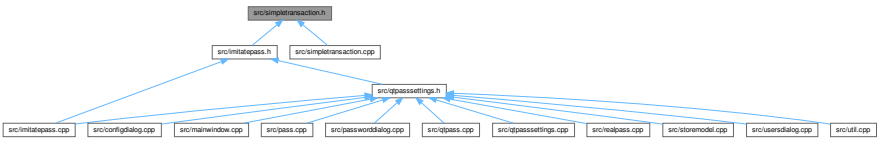
14.82 src/simpletransaction.h File Reference

```
#include "enums.h"
#include <queue>
```

Include dependency graph for simpletransaction.h:



This graph shows which files directly or indirectly include this file:



Classes

- class simpleTransaction

14.83 simpletransaction.h

Go to the documentation of this file.

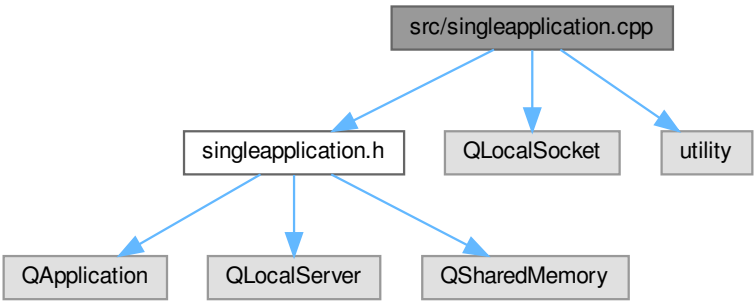
```
00001 #ifndef SIMPLETRANSACTION_H
00002 #define SIMPLETRANSACTION_H
00003
00004 #include "enums.h"
00005 #include <queue>
00006
00007 class simpleTransaction {
00008   int transactionDepth;
00009   Enums::PROCESS lastInTransaction;
00010   std::queue<std::pair<Enums::PROCESS, Enums::PROCESS» transactionQueue;
00011
00012 public:
00013   simpleTransaction()
```

```
00014       : transactionDepth(0), lastInTransaction(Enums::INVALID) {}
00020   void transactionStart();
00030   void transactionAdd(Enums::PROCESS);
00036   void transactionEnd(Enums::PROCESS);
00044   Enums::PROCESS transactionIsOver(Enums::PROCESS);
00045 };
00046
00047 #endif // SIMPLETRANSACTION_H
```

14.84 src/singleapplication.cpp File Reference

```
#include "singleapplication.h"
#include <QLocalSocket>
#include <utility>
```

Include dependency graph for singleapplication.cpp:



14.85 singleapplication.cpp

Go to the documentation of this file.

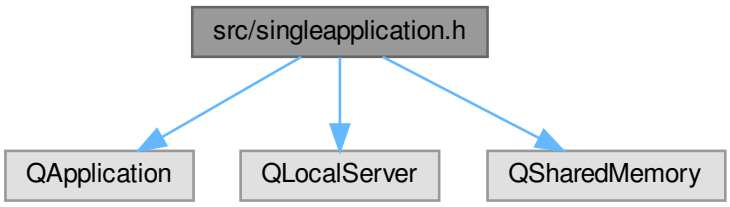
```
00001 #include "singleapplication.h"
00002 #include <QLocalSocket>
00003 #include <utility>
00004 #ifdef QT_DEBUG
00005 #include "debughelper.h"
00006 #endif
00007
00015 SingleApplication::SingleApplication(int &argc, char *argv[], QString uniqueKey)
00016     : QApplication(argc, argv), _uniqueKey(std::move(uniqueKey)) {
00017   sharedMemory.setKey(_uniqueKey);
00018   if (sharedMemory.attach()) {
00019     _isRunning = true;
00020   } else {
00021     _isRunning = false;
00022     // create shared memory.
00023     if (!sharedMemory.create(1)) {
00024 #ifdef QT_DEBUG
00025       dbg() << "Unable to create single instance.";
00026 #endif
00027       return;
00028     }
00029     // create local server and listen to incomming messages from other
00030     // instances.
00031     localServer.reset(new QLocalServer(this));
00032     connect(localServer.data(), &QLocalServer::newConnection, this,
00033             &SingleApplication::receiveMessage);
00034     localServer->listen(_uniqueKey);
00035   }
00036 }
```

```
00037
00038 // public slots.
00039
00044 void SingleApplication::receiveMessage() {
00045   QLocalSocket *localSocket = localServer->nextPendingConnection();
00046   if (!localSocket->waitForReadyRead(timeout)) {
00047 #ifdef QT_DEBUG
00048     dbg() « localSocket->errorString().toLatin1();
00049 #endif
00050     return;
00051   }
00052   QByteArray byteArray = localSocket->readAll();
00053   QString message = QString::fromUtf8(byteArray.constData());
00054   emit messageAvailable(message);
00055   localSocket->disconnectFromServer();
00056 }
00057
00058 // public functions.
00064 bool SingleApplication::isRunning() { return _isRunning; }
00065
00072 bool SingleApplication::sendMessage(const QString &message) {
00073   if (!_isRunning)
00074     return false;
00075   QLocalSocket localSocket(this);
00076   localSocket.connectToServer(_uniqueKey, QIODevice::WriteOnly);
00077   if (!localSocket.waitForConnected(timeout)) {
00078 #ifdef QT_DEBUG
00079     dbg() « localSocket.errorString().toLatin1();
00080 #endif
00081     return false;
00082   }
00083   localSocket.write(message.toUtf8());
00084   if (!localSocket.waitForBytesWritten(timeout)) {
00085 #ifdef QT_DEBUG
00086     dbg() « localSocket.errorString().toLatin1();
00087 #endif
00088     return false;
00089   }
00090   localSocket.disconnectFromServer();
00091   return true;
00092 }
```

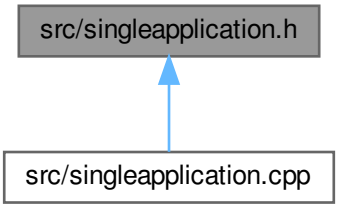
14.86 src/singleapplication.h File Reference

```
#include <QApplication>
#include <QLocalServer>
#include <QSharedMemory>
```

Include dependency graph for singleapplication.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SingleApplication

  The SingleApplication class is used for commandline intergration.

14.87 singleapplication.h

Go to the documentation of this file.

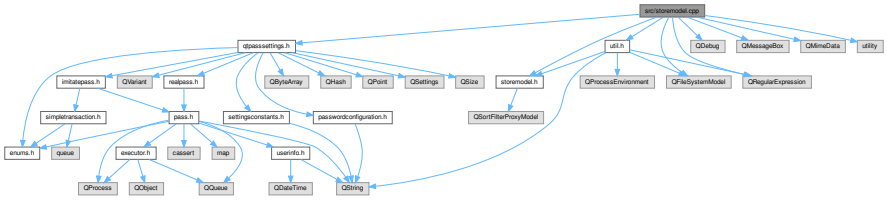
```
00001 #ifndef SINGLEAPPLICATION_H_
00002 #define SINGLEAPPLICATION_H_
00003
00004 #include <QApplication>
00005 #include <QLocalServer>
00006 #include <QSharedMemory>
00007
00014 class SingleApplication : public QApplication {
00015   Q_OBJECT
00016 public:
00017   SingleApplication(int &argc, char *argv[], QString uniqueKey);
00018   bool isRunning();
00019   bool sendMessage(const QString &message);
00020
00021 public slots:
00022   void receiveMessage();
00023
00024 signals:
00029   void messageAvailable(QString message);
00030
00031 private:
00032   bool _isRunning;
00033   QString _uniqueKey;
00034   QSharedMemory sharedMemory;
00035   QScopedPointer<QLocalServer> localServer;
00036
00037   static const int timeout = 1000;
00038 };
00039
00040 #endif // SINGLEAPPLICATION_H_
```

14.88 src/storemodel.cpp File Reference

```
#include "storemodel.h"
#include "qtpasssettings.h"
#include "util.h"
```

```
#include <QDebug>
#include <QFileSystemModel>
#include <QMessageBox>
#include <QMimeData>
#include <QRegularExpression>
#include <utility>
```

Include dependency graph for storemodel.cpp:



Functions

- QDataStream & operator<< (QDataStream &out, const dragAndDropInfoPasswordStore &dragAndDropInfoPasswordStore)
- QDataStream & operator>> (QDataStream &in, dragAndDropInfoPasswordStore &dragAndDropInfoPasswordStore)

14.88.1 Function Documentation

14.88.1.1 operator<<()

```
QDataStream & operator<< (
            QDataStream & out,
            const dragAndDropInfoPasswordStore & dragAndDropInfoPasswordStore )
```

Definition at line 12 of file storemodel.cpp.

14.88.1.2 operator>>()

```
QDataStream & operator>> (
            QDataStream & in,
            dragAndDropInfoPasswordStore & dragAndDropInfoPasswordStore )
```

Definition at line 22 of file storemodel.cpp.

14.89 storemodel.cpp

Go to the documentation of this file.

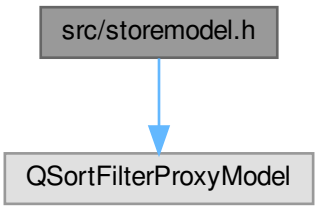
```
00001 #include "storemodel.h"
00002 #include "qtpasssettings.h"
00003
00004 #include "util.h"
00005 #include <QDebug>
00006 #include <QFileSystemModel>
00007 #include <QMessageBox>
00008 #include <QMimeData>
00009 #include <QRegularExpression>
00010 #include <utility>
00011
00012 QDataStream &
00013 operator«(QDataStream &out,
00014           const dragAndDropInfoPasswordStore &dragAndDropInfoPasswordStore) {
00015   out « dragAndDropInfoPasswordStore.isDir
00016       « dragAndDropInfoPasswordStore.isFile
00017       « dragAndDropInfoPasswordStore.path;
00018   return out;
00019 }
00020
00021 QDataStream &
00022 operator»(QDataStream &in,
00023           dragAndDropInfoPasswordStore &dragAndDropInfoPasswordStore) {
00024   in » dragAndDropInfoPasswordStore.isDir »
00025       dragAndDropInfoPasswordStore.isFile » dragAndDropInfoPasswordStore.path;
00026   return in;
00027 }
00028
00034 StoreModel::StoreModel() { fs = nullptr; }
00035
00043 bool StoreModel::filterAcceptsRow(int sourceRow,
00044                                   const QModelIndex &sourceParent) const {
00045   QModelIndex index = sourceModel()->index(sourceRow, 0, sourceParent);
00046   return ShowThis(index);
00047 }
00048
00055 bool StoreModel::ShowThis(const QModelIndex index) const {
00056   bool retVal = false;
00057   if (fs == nullptr)
00058     return retVal;
00059   // Gives you the info for number of childs with a parent
00060   if (sourceModel()->rowCount(index) > 0) {
00061     for (int nChild = 0; nChild < sourceModel()->rowCount(index); ++nChild) {
00062       QModelIndex childIndex = sourceModel()->index(nChild, 0, index);
00063       if (!childIndex.isValid())
00064         break;
00065       retVal = ShowThis(childIndex);
00066       if (retVal)
00067         break;
00068     }
00069   } else {
00070     QModelIndex useIndex = sourceModel()->index(index.row(), 0, index.parent());
00071     QString path = fs->filePath(useIndex);
00072     path = QDir(store).relativeFilePath(path);
00073     path.replace(Util::endsWithGpg(), "");
00074     retVal = path.contains(filterRegularExpression());
00075   }
00076   return retVal;
00077 }
00078
00084 void StoreModel::setModelAndStore(QFileSystemModel *sourceModel,
00085                                   QString passStore) {
00086   setSourceModel(sourceModel);
00087   fs = sourceModel;
00088   store = std::move(passStore);
00089 }
00090
00097 QVariant StoreModel::data(const QModelIndex &index, int role) const {
00098   if (!index.isValid())
00099     return QVariant();
00100
00101   QVariant initial_value;
00102   initial_value = QSortFilterProxyModel::data(index, role);
00103
00104   if (role == Qt::DisplayRole) {
00105     QString name = initial_value.toString();
00106     name.replace(Util::endsWithGpg(), "");
00107     initial_value.setValue(name);
00108   }
00109
00110   return initial_value;
00111 }
```

```
00112
00117 Qt::DropActions StoreModel::supportedDropActions() const {
00118   return Qt::CopyAction | Qt::MoveAction;
00119 }
00120
00125 Qt::DropActions StoreModel::supportedDragActions() const {
00126   return Qt::CopyAction | Qt::MoveAction;
00127 }
00128
00134 Qt::ItemFlags StoreModel::flags(const QModelIndex &index) const {
00135   Qt::ItemFlags defaultFlags = QSortFilterProxyModel::flags(index);
00136
00137   if (index.isValid()) {
00138     return Qt::ItemIsDragEnabled | Qt::ItemIsDropEnabled | defaultFlags;
00139   }
00140   return Qt::ItemIsDropEnabled | defaultFlags;
00141 }
00142
00147 QStringList StoreModel::mimeTypes() const {
00148   QStringList types;
00149   types « "application/vnd+qtpass.dragAndDropInfoPasswordStore";
00150   return types;
00151 }
00152
00158 QMimeData *StoreModel::mimeData(const QModelIndexList &indexes) const {
00159   dragAndDropInfoPasswordStore info;
00160
00161   QByteArray encodedData;
00162   // only use the first, otherwise we should enable multiselection
00163   QModelIndex index = indexes.at(0);
00164   if (index.isValid()) {
00165     QModelIndex useIndex = mapToSource(index);
00166
00167     info.isDir = fs->fileInfo(useIndex).isDir();
00168     info.isFile = fs->fileInfo(useIndex).isFile();
00169     info.path = fs->fileInfo(useIndex).absoluteFilePath();
00170     QDataStream stream(&encodedData, QIODevice::WriteOnly);
00171     stream « info;
00172   }
00173
00174   auto *mimeData = new QMimeData();
00175   mimeData->setData("application/vnd+qtpass.dragAndDropInfoPasswordStore",
00176                     encodedData);
00177   return mimeData;
00178 }
00179
00189 bool StoreModel::canDropMimeData(const QMimeData *data, Qt::DropAction action,
00190                                  int row, int column,
00191                                  const QModelIndex &parent) const {
00192 #ifdef QT_DEBUG
00193   qDebug() « action « row;
00194 #else
00195   Q_UNUSED(action)
00196   Q_UNUSED(row)
00197 #endif
00198
00199   QModelIndex useIndex =
00200       this->index(parent.row(), parent.column(), parent.parent());
00201   QByteArray encodedData =
00202       data->data("application/vnd+qtpass.dragAndDropInfoPasswordStore");
00203   QDataStream stream(&encodedData, QIODevice::ReadOnly);
00204   dragAndDropInfoPasswordStore info;
00205   stream » info;
00206   if (!data->hasFormat("application/vnd+qtpass.dragAndDropInfoPasswordStore"))
00207     return false;
00208
00209   if (column > 0) {
00210     return false;
00211   }
00212
00213   // you can drop a folder on a folder
00214   if (fs->fileInfo(mapToSource(useIndex)).isDir() && info.isDir) {
00215     return true;
00216   }
00217   // you can drop a file on a folder
00218   if (fs->fileInfo(mapToSource(useIndex)).isDir() && info.isFile) {
00219     return true;
00220   }
00221   // you can drop a file on a file
00222   if (fs->fileInfo(mapToSource(useIndex)).isFile() && info.isFile) {
00223     return true;
00224   }
00225
00226   return false;
00227 }
00228
00238 bool StoreModel::dropMimeData(const QMimeData *data, Qt::DropAction action,
```

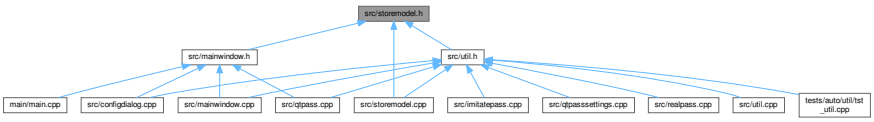
```
00239                               int row, int column, const QModelIndex &parent) {
00240   if (!canDropMimeData(data, action, row, column, parent))
00241     return false;
00242
00243   if (action == Qt::IgnoreAction) {
00244     return true;
00245   }
00246   QByteArray encodedData =
00247       data->data("application/vnd+qtpass.dragAndDropInfoPasswordStore");
00248
00249   QDataStream stream(&encodedData, QIODevice::ReadOnly);
00250   dragAndDropInfoPasswordStore info;
00251   stream » info;
00252   QModelIndex destIndex =
00253       this->index(parent.row(), parent.column(), parent.parent());
00254   QFileInfo destFileinfo = fs->fileInfo(mapToSource(destIndex));
00255   QFileInfo srcFileInfo = QFileInfo(info.path);
00256   QString cleanedSrc = QDir::cleanPath(srcFileInfo.absoluteFilePath());
00257   QString cleanedDest = QDir::cleanPath(destFileinfo.absoluteFilePath());
00258   if (info.isDir) {
00259     // dropped dir onto dir
00260     if (destFileinfo.isDir()) {
00261       QDir destDir = QDir(cleanedDest).filePath(srcFileInfo.fileName());
00262       QString cleanedDestDir = QDir::cleanPath(destDir.absolutePath());
00263       if (action == Qt::MoveAction) {
00264         QtPassSettings::getPass()->Move(cleanedSrc, cleanedDestDir);
00265       } else if (action == Qt::CopyAction) {
00266         QtPassSettings::getPass()->Copy(cleanedSrc, cleanedDestDir);
00267       }
00268     }
00269   } else if (info.isFile) {
00270     // dropped file onto a directory
00271     if (destFileinfo.isDir()) {
00272       if (action == Qt::MoveAction) {
00273         QtPassSettings::getPass()->Move(cleanedSrc, cleanedDest);
00274       } else if (action == Qt::CopyAction) {
00275         QtPassSettings::getPass()->Copy(cleanedSrc, cleanedDest);
00276       }
00277     } else if (destFileinfo.isFile()) {
00278       // dropped file onto a file
00279       int answer = QMessageBox::question(
00280           nullptr, tr("force overwrite?"),
00281           tr("overwrite %1 with %2?").arg(cleanedDest, cleanedSrc),
00282           QMessageBox::Yes | QMessageBox::No);
00283       bool force = answer == QMessageBox::Yes;
00284       if (action == Qt::MoveAction) {
00285         QtPassSettings::getPass()->Move(cleanedSrc, cleanedDest, force);
00286       } else if (action == Qt::CopyAction) {
00287         QtPassSettings::getPass()->Copy(cleanedSrc, cleanedDest, force);
00288       }
00289     }
00290   }
00291   return true;
00292 }
00293
00300 bool StoreModel::lessThan(const QModelIndex &source_left,
00301                           const QModelIndex &source_right) const {
00302 /* matches logic in QFileSystemModelSorter::compareNodes() */
00303 #ifndef Q_OS_MAC
00304   if (fs && (source_left.column() == 0 || source_left.column() == 1)) {
00305     bool leftD = fs->isDir(source_left);
00306     bool rightD = fs->isDir(source_right);
00307
00308     if (leftD ^ rightD)
00309       return leftD;
00310   }
00311 #endif
00312
00313   return QSortFilterProxyModel::lessThan(source_left, source_right);
00314 }
```

14.90 src/storemodel.h File Reference

`#include <QSortFilterProxyModel>`
Include dependency graph for storemodel.h:



This graph shows which files directly or indirectly include this file:



Classes

- class StoreModel

  The QSortFilterProxyModel for handling filesystem searches.
- struct dragAndDropInfoPasswordStore

14.91 storemodel.h

Go to the documentation of this file.

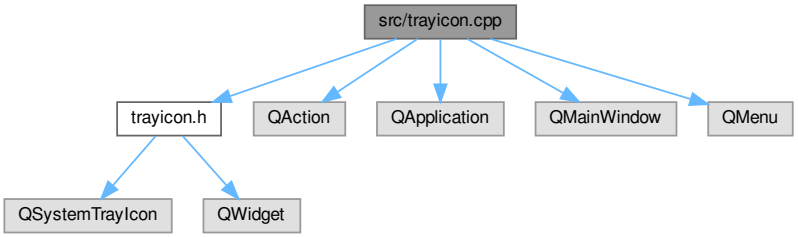
```
00001 #ifndef STOREMODEL_H_
00002 #define STOREMODEL_H_
00003
00004 #include <QSortFilterProxyModel>
00005
00010 class QFileSystemModel;
00011 class StoreModel : public QSortFilterProxyModel {
00012   Q_OBJECT
00013
00014 private:
00015   QFileSystemModel *fs;
00016   QString store;
00017
00018 public:
00019   StoreModel();
00020
00021   bool filterAcceptsRow(int, const QModelIndex &) const override;
00022   bool ShowThis(const QModelIndex) const;
00023   void setModelAndStore(QFileSystemModel *sourceModel, QString passStore);
00024   QVariant data(const QModelIndex &index, int role) const override;
00025   bool lessThan(const QModelIndex &source_left,
```

```
00026                const QModelIndex &source_right) const override;
00027
00028   // QAbstractItemModel interface
00029 public:
00030   Qt::DropActions supportedDropActions() const override;
00031   Qt::DropActions supportedDragActions() const override;
00032   Qt::ItemFlags flags(const QModelIndex &index) const override;
00033   QStringList mimeTypes() const override;
00034   QMimeData *mimeData(const QModelIndexList &indexes) const override;
00035   bool canDropMimeData(const QMimeData *data, Qt::DropAction action, int row,
00036                        int column, const QModelIndex &parent) const override;
00037   bool dropMimeData(const QMimeData *data, Qt::DropAction action, int row,
00038                     int column, const QModelIndex &parent) override;
00039 };
00045 struct dragAndDropInfoPasswordStore {
00046   bool isDir;
00047   bool isFile;
00048   QString path;
00049 };
00050
00051 #endif // STOREMODEL_H_
```

14.92 src/trayicon.cpp File Reference

```
#include "trayicon.h"
#include <QAction>
#include <QApplication>
#include <QMainWindow>
#include <QMenu>
```

Include dependency graph for trayicon.cpp:



14.93 trayicon.cpp

Go to the documentation of this file.

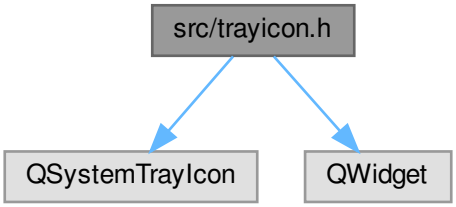
```
00001 #include "trayicon.h"
00002 #include <QAction>
00003 #include <QApplication>
00004 #include <QMainWindow>
00005 #include <QMenu>
00006
00007 #ifdef QT_DEBUG
00008 #include "debughelper.h"
00009 #endif
00010
00016 TrayIcon::TrayIcon(QMainWindow *parent)
00017     : showAction(nullptr), hideAction(nullptr), minimizeAction(nullptr),
00018       maximizeAction(nullptr), restoreAction(nullptr), quitAction(nullptr),
00019       sysTrayIcon(nullptr), trayIconMenu(nullptr), isAllocated(false) {
00020   parentwin = parent;
00021
00022   if (QSystemTrayIcon::isSystemTrayAvailable()) {
00023     createActions();
```

```
00024     createTrayIcon();
00025
00026     sysTrayIcon->setIcon(
00027         QIcon::fromTheme("qtpass-tray", QIcon(":/artwork/icon.png")));
00028
00029     sysTrayIcon->show();
00030
00031     QObject::connect(sysTrayIcon, &QSystemTrayIcon::activated, this,
00032                      &TrayIcon::iconActivated);
00033
00034     isAllocated = true;
00035   }
00036 #ifdef QT_DEBUG
00037   else {
00038     dbg() « "No tray icon for this OS possibly also not show options?";
00039   }
00040 #endif
00041 }
00042
00047 void TrayIcon::setVisible(bool visible) {
00048   if (visible)
00049     parentwin->show();
00050   else
00051     parentwin->hide();
00052 }
00053
00057 bool TrayIcon::getIsAllocated() { return isAllocated; }
00058
00062 void TrayIcon::createActions() {
00063   showAction = new QAction(tr("&Show"), this);
00064   connect(showAction, &QAction::triggered, parentwin, &QWidget::show);
00065   hideAction = new QAction(tr("&Hide"), this);
00066   connect(hideAction, &QAction::triggered, parentwin, &QWidget::hide);
00067
00068   minimizeAction = new QAction(tr("Mi&nimize"), this);
00069   connect(minimizeAction, &QAction::triggered, parentwin,
00070           &QWidget::showMinimized);
00071   maximizeAction = new QAction(tr("Ma&ximize"), this);
00072   connect(maximizeAction, &QAction::triggered, parentwin,
00073           &QWidget::showMaximized);
00074   restoreAction = new QAction(tr("&Restore"), this);
00075   connect(restoreAction, &QAction::triggered, parentwin, &QWidget::showNormal);
00076
00077   quitAction = new QAction(tr("&Quit"), this);
00078   connect(quitAction, &QAction::triggered, qApp, &QApplication::quit);
00079 }
00080
00084 void TrayIcon::createTrayIcon() {
00085   trayIconMenu = new QMenu(this);
00086   trayIconMenu->addAction(showAction);
00087   trayIconMenu->addAction(hideAction);
00088   trayIconMenu->addAction(minimizeAction);
00089   trayIconMenu->addAction(maximizeAction);
00090   trayIconMenu->addAction(restoreAction);
00091   trayIconMenu->addSeparator();
00092   trayIconMenu->addAction(quitAction);
00093
00094   sysTrayIcon = new QSystemTrayIcon(this);
00095   sysTrayIcon->setContextMenu(trayIconMenu);
00096 }
00097
00101 void TrayIcon::showHideParent() {
00102   if (parentwin->isVisible())
00103     parentwin->hide();
00104   else
00105     parentwin->show();
00106 }
00107
00112 void TrayIcon::iconActivated(QSystemTrayIcon::ActivationReason reason) {
00113   switch (reason) {
00114   case QSystemTrayIcon::Trigger:
00115   case QSystemTrayIcon::DoubleClick:
00116     showHideParent();
00117     break;
00118   case QSystemTrayIcon::MiddleClick:
00119     showMessage("test", "test msg", 1000);
00120     break;
00121   default: {
00122   }
00123   }
00124 }
00125
00132 void TrayIcon::showMessage(const QString &title, const QString &msg, int time) {
00133   sysTrayIcon->showMessage(title, msg, QSystemTrayIcon::Information, time);
00134 }
```

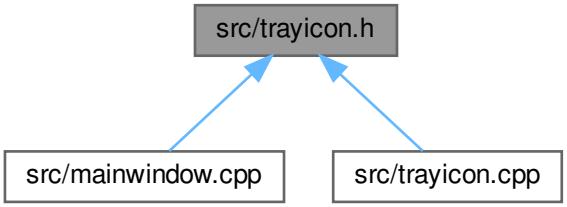
14.94 src/trayicon.h File Reference

```
#include <QSystemTrayIcon>
#include <QWidget>
```

Include dependency graph for trayicon.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TrayIcon

  Handles the systemtray icon and menu.

14.95 trayicon.h

Go to the documentation of this file.

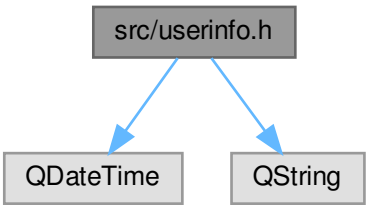
```
00001 #ifndef TRAYICON_H_
00002 #define TRAYICON_H_
00003
00004 #include <QSystemTrayIcon>
00005 #include <QWidget>
00006
00011 class QAction;
00012 class QMainWindow;
00013 class QMenu;
00014 class TrayIcon : public QWidget {
```

```
00015   Q_OBJECT
00016
00017 public:
00018   explicit TrayIcon(QMainWindow *parent);
00019   void showMessage(const QString &title, const QString &msg, int time);
00020   void setVisible(bool visible);
00021   bool getIsAllocated();
00022
00023 signals:
00024
00025 public slots:
00026   void showHideParent();
00027   void iconActivated(QSystemTrayIcon::ActivationReason reason);
00028
00029 private:
00030   void createActions();
00031   void createTrayIcon();
00032
00033   QAction *showAction;
00034   QAction *hideAction;
00035   QAction *minimizeAction;
00036   QAction *maximizeAction;
00037   QAction *restoreAction;
00038   QAction *quitAction;
00039
00040   QSystemTrayIcon *sysTrayIcon;
00041   QMenu *trayIconMenu;
00042   QMainWindow *parentwin;
00043
00044   bool isAllocated;
00045 };
00046
00047 #endif // TRAYICON_H_
```

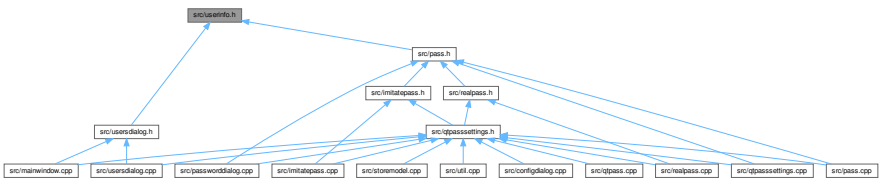
14.96 src/userinfo.h File Reference

```
#include <QDateTime>
#include <QString>
```

Include dependency graph for userinfo.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct UserInfo

  Stores key info lines including validity, creation date and more.

14.97 userinfo.h

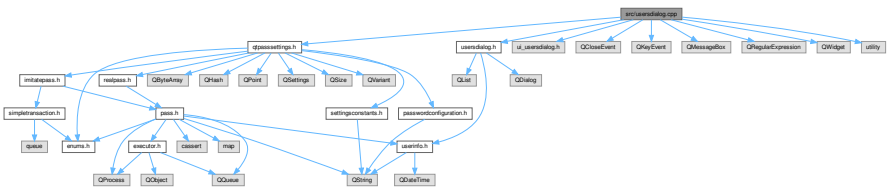
Go to the documentation of this file.

```
00001 #ifndef DATAHELPERS_H
00002 #define DATAHELPERS_H
00003
00004 #include <QDateTime>
00005 #include <QString>
00006
00011 struct UserInfo {
00012   UserInfo() : validity('-'), have_secret(false), enabled(false) {}
00013
00018   bool fullyValid() { return validity == 'f' || validity == 'u'; }
00023   bool marginallyValid() { return validity == 'm'; }
00027   bool isValid() { return fullyValid() || marginallyValid(); }
00028
00032   QString name;
00036   QString key_id;
00041   char validity;
00046   bool have_secret;
00050   bool enabled;
00054   QDateTime expiry;
00058   QDateTime created;
00059 };
00060
00061 #endif // DATAHELPERS_H
```

14.98 src/usersdialog.cpp File Reference

```
#include "usersdialog.h"
#include "qtpasssettings.h"
#include "ui_usersdialog.h"
#include <QCloseEvent>
#include <QKeyEvent>
#include <QMessageBox>
#include <QRegularExpression>
#include <QWidget>
#include <utility>
```

Include dependency graph for usersdialog.cpp:



14.99 usersdialog.cpp

Go to the documentation of this file.

```
00001 #include "usersdialog.h"
```

```
00002 #include "qtpasssettings.h"
00003 #include "ui_usersdialog.h"
00004 #include <QCloseEvent>
00005 #include <QKeyEvent>
00006 #include <QMessageBox>
00007 #include <QRegularExpression>
00008 #include <QWidget>
00009 #include <utility>
00010
00011 #ifdef QT_DEBUG
00012 #include "debughelper.h"
00013 #endif
00018 UsersDialog::UsersDialog(QString dir, QWidget *parent)
00019     : QDialog(parent), ui(new Ui::UsersDialog), m_dir(std::move(dir)) {
00020
00021   ui->setupUi(this);
00022
00023   QList<UserInfo> users = QtPassSettings::getPass()->listKeys();
00024   if (users.isEmpty()) {
00025     QMessageBox::critical(parent, tr("Keylist missing"),
00026                           tr("Could not fetch list of available GPG keys"));
00027     return;
00028   }
00029
00030   QList<UserInfo> secret_keys = QtPassSettings::getPass()->listKeys("", true);
00031   foreach (const UserInfo &sec, secret_keys) {
00032     for (auto &user : users)
00033       if (sec.key_id == user.key_id)
00034         user.have_secret = true;
00035   }
00036
00037   QList<UserInfo> selected_users;
00038   int count = 0;
00039
00040   QStringList recipients = QtPassSettings::getPass()->getRecipientString(
00041       m_dir.isEmpty() ? "" : m_dir, " ", &count);
00042   if (!recipients.isEmpty())
00043     selected_users = QtPassSettings::getPass()->listKeys(recipients);
00044   foreach (const UserInfo &sel, selected_users) {
00045     for (auto &user : users)
00046       if (sel.key_id == user.key_id)
00047         user.enabled = true;
00048   }
00049
00050   if (count > selected_users.size()) {
00051     // Some keys seem missing from keyring, add them separately
00052     QStringList recipients = QtPassSettings::getPass()->getRecipientList(
00053         m_dir.isEmpty() ? "" : m_dir);
00054     foreach (const QString recipient, recipients) {
00055       if (QtPassSettings::getPass()->listKeys(recipient).empty()) {
00056         UserInfo i;
00057         i.enabled = true;
00058         i.key_id = recipient;
00059         i.name = " ?? " + tr("Key not found in keyring");
00060         users.append(i);
00061       }
00062     }
00063   }
00064
00065   m_userList = users;
00066   populateList();
00067
00068   connect(ui->buttonBox, &QDialogButtonBox::accepted, this,
00069           &UsersDialog::accept);
00070   connect(ui->buttonBox, &QDialogButtonBox::rejected, this, &QDialog::reject);
00071   connect(ui->listWidget, &QListWidget::itemChanged, this,
00072           &UsersDialog::itemChange);
00073
00074   ui->lineEdit->setClearButtonEnabled(true);
00075 }
00076
00080 UsersDialog::~UsersDialog() { delete ui; }
00081
00082 Q_DECLARE_METATYPE(UserInfo *)
00083
00084
00087 void UsersDialog::accept() {
00088   QtPassSettings::getPass()->Init(m_dir, m_userList);
00089
00090   QDialog::accept();
00091 }
00092
00098 void UsersDialog::closeEvent(QCloseEvent *event) {
00099   // TODO(annejan) save window size or something
00100   event->accept();
00101 }
00102
```

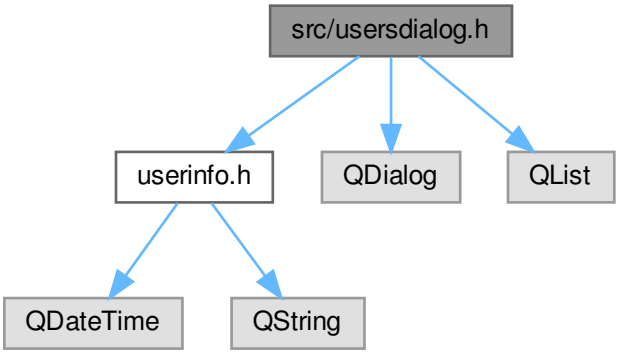
```
00108 void UsersDialog::keyPressEvent(QKeyEvent *event) {
00109   switch (event->key()) {
00110   case Qt::Key_Escape:
00111     ui->lineEdit->clear();
00112     break;
00113   default:
00114     break;
00115   }
00116 }
00117
00122 void UsersDialog::itemChange(QListWidgetItem *item) {
00123   if (!item)
00124     return;
00125   auto *info = item->data(Qt::UserRole).value<UserInfo *>();
00126   if (!info)
00127     return;
00128   info->enabled = item->checkState() == Qt::Checked;
00129 }
00130
00136 void UsersDialog::populateList(const QString &filter) {
00137   QRegularExpression nameFilter(
00138       QRegularExpression::wildcardToRegularExpression("*" + filter + "*"),
00139       QRegularExpression::CaseInsensitiveOption);
00140   ui->listWidget->clear();
00141   if (!m_userList.isEmpty()) {
00142     for (auto &user : m_userList) {
00143       if (filter.isEmpty() || nameFilter.match(user.name).hasMatch()) {
00144         if (!user.isValid() && !ui->checkBox->isChecked())
00145           continue;
00146         if (user.expiry.toSecsSinceEpoch() > 0 &&
00147             user.expiry.daysTo(QDateTime::currentDateTime()) > 0 &&
00148             !ui->checkBox->isChecked())
00149           continue;
00150         QString userText = user.name + "\n" + user.key_id;
00151         if (user.created.toSecsSinceEpoch() > 0) {
00152           userText +=
00153               " " + tr("created") + " " +
00154               QLocale::system().toString(user.created, QLocale::ShortFormat);
00155         }
00156         if (user.expiry.toSecsSinceEpoch() > 0)
00157           userText +=
00158               " " + tr("expires") + " " +
00159               QLocale::system().toString(user.expiry, QLocale::ShortFormat);
00160         auto *item = new QListWidgetItem(userText, ui->listWidget);
00161         item->setCheckState(user.enabled ? Qt::Checked : Qt::Unchecked);
00162         item->setData(Qt::UserRole, QVariant::fromValue(&user));
00163         if (user.have_secret) {
00164           // item->setForeground(QColor(32, 74, 135));
00165           item->setForeground(Qt::blue);
00166           QFont font;
00167           font.setFamily(font.defaultFamily());
00168           font.setBold(true);
00169           item->setFont(font);
00170         } else if (!user.isValid()) {
00171           item->setBackground(QColor(164, 0, 0));
00172           item->setForeground(Qt::white);
00173         } else if (user.expiry.toSecsSinceEpoch() > 0 &&
00174                    user.expiry.daysTo(QDateTime::currentDateTime()) > 0) {
00175           item->setForeground(QColor(164, 0, 0));
00176         } else if (!user.fullyValid()) {
00177           item->setBackground(QColor(164, 80, 0));
00178           item->setForeground(Qt::white);
00179         }
00180
00181         ui->listWidget->addItem(item);
00182       }
00183     }
00184   }
00185 }
00186
00191 void UsersDialog::on_lineEdit_textChanged(const QString &filter) {
00192   populateList(filter);
00193 }
00194
00198 void UsersDialog::on_checkBox_clicked() { populateList(ui->lineEdit->text()); }
```

14.100 src/usersdialog.h File Reference

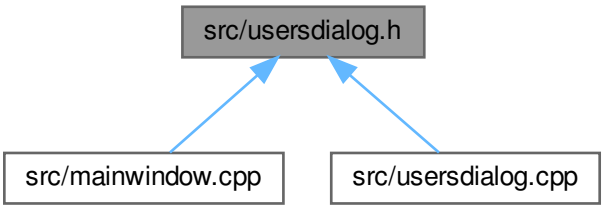
```
#include "userinfo.h"
#include <QDialog>
```

```
#include <QList>
```

Include dependency graph for usersdialog.h:



This graph shows which files directly or indirectly include this file:



Classes

- class UsersDialog

  Handles listing and editing of GPG users.

Namespaces

- namespace Ui

14.101 usersdialog.h

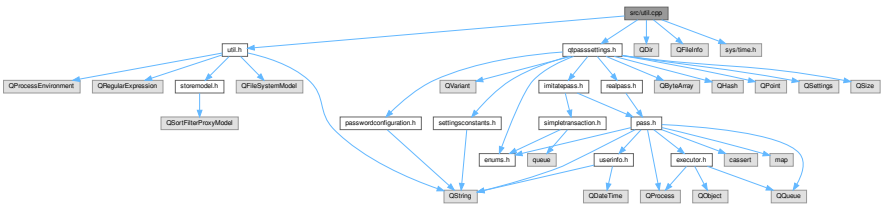
Go to the documentation of this file.

```
00001 #ifndef USERSDIALOG_H_
00002 #define USERSDIALOG_H_
00003
00004 #include "userinfo.h"
00005
00006 #include <QDialog>
00007 #include <QList>
00008
00009 namespace Ui {
00010 class UsersDialog;
00011 }
00012
00013 class QCloseEvent;
00014 class QKeyEvent;
00015 class QListWidgetItem;
00016
00023 class UsersDialog : public QDialog {
00024   Q_OBJECT
00025
00026 public:
00027   explicit UsersDialog(QString dir, QWidget *parent = nullptr);
00028   ~UsersDialog();
00029
00030 public slots:
00031   void accept();
00032
00033 protected:
00034   void closeEvent(QCloseEvent *event);
00035   void keyPressEvent(QKeyEvent *event);
00036
00037 private slots:
00038   void itemChange(QListWidgetItem *item);
00039   void on_lineEdit_textChanged(const QString &filter);
00040   void on_checkBox_clicked();
00041
00042 private:
00043   Ui::UsersDialog *ui;
00044   QList<UserInfo> m_userList;
00045   QString m_dir;
00046
00047   void populateList(const QString &filter = QString());
00048 };
00049
00050 #endif // USERSDIALOG_H_
```

14.102 src/util.cpp File Reference

```
#include "util.h"
#include <QDir>
#include <QFileInfo>
#include <sys/time.h>
#include "qtpasssettings.h"
```

Include dependency graph for util.cpp:



14.103 util.cpp

Go to the documentation of this file.

```
00001 #include "util.h"
00002 #include <QDir>
00003 #include <QFileInfo>
00004 #ifdef Q_OS_WIN
00005 #include <windows.h>
00006 #else
00007 #include <sys/time.h>
00008 #endif
00009 #include "qtpasssettings.h"
00010
00011 #ifdef QT_DEBUG
00012 #include "debughelper.h"
00013 #endif
00014
00015 QProcessEnvironment Util::_env;
00016 bool Util::_envInitialised;
00017
00022 void Util::initialiseEnvironment() {
00023   if (!_envInitialised) {
00024     _env = QProcessEnvironment::systemEnvironment();
00025 #ifdef __APPLE__
00026     QString path = _env.value("PATH");
00027     if (!path.contains("/usr/local/MacGPG2/bin") &&
00028         QFile("/usr/local/MacGPG2/bin").exists())
00029       path += ":/usr/local/MacGPG2/bin";
00030     if (!path.contains("/usr/local/bin"))
00031       path += ":/usr/local/bin";
00032     _env.insert("PATH", path);
00033 #endif
00034 #ifdef Q_OS_WIN
00035     QString path = _env.value("PATH");
00036     if (!path.contains("C:\\Program Files\\WinGPG\\x86") &&
00037         QFile("C:\\Program Files\\WinGPG\\x86").exists())
00038       path += ";C:\\Program Files\\WinGPG\\x86";
00039     if (!path.contains("C:\\Program Files\\GnuPG\\bin") &&
00040         QFile("C:\\Program Files\\GnuPG\\bin").exists())
00041       path += ";C:\\Program Files\\GnuPG\bin";
00042     _env.insert("PATH", path);
00043 #endif
00044 #ifdef QT_DEBUG
00045     dbg() « _env.value("PATH");
00046 #endif
00047     _envInitialised = true;
00048   }
00049 }
00050
00056 QString Util::findPasswordStore() {
00057   QString path;
00058   initialiseEnvironment();
00059   if (_env.contains("PASSWORD_STORE_DIR")) {
00060     path = _env.value("PASSWORD_STORE_DIR");
00061   } else {
00062 #ifdef Q_OS_WIN
00063     path = QDir::homePath() + QDir::separator() + "password-store" +
00064            QDir::separator();
00065 #else
00066     path = QDir::homePath() + QDir::separator() + ".password-store" +
00067            QDir::separator();
00068 #endif
00069   }
00070   return Util::normalizeFolderPath(path);
00071 }
00072
00079 QString Util::normalizeFolderPath(QString path) {
00080   if (!path.endsWith("/") && !path.endsWith(QDir::separator()))
00081     path += QDir::separator();
00082   return QDir::toNativeSeparators(path);
00083 }
00084
00090 QString Util::findBinaryInPath(QString binary) {
00091   initialiseEnvironment();
00092
00093   QString ret = "";
00094
00095   binary.prepend(QDir::separator());
00096
00097   if (_env.contains("PATH")) {
00098     QString path = _env.value("PATH");
00099
00100     QStringList entries;
00101 #ifndef Q_OS_WIN
00102     entries = path.split(':');
```

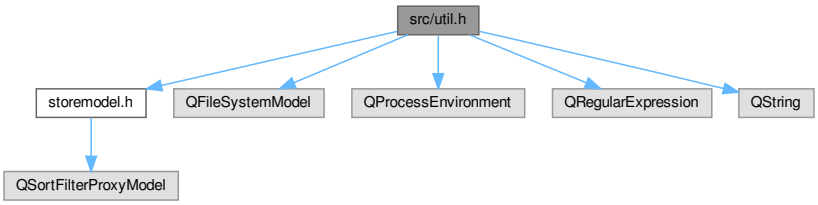
```
00103     if (entries.length() < 2) {
00104 #endif
00105       entries = path.split(';');
00106 #ifndef Q_OS_WIN
00107     }
00108 #endif
00109
00110     foreach (QString entry, entries) {
00111       QScopedPointer<QFileInfo> qfi(new QFileInfo(entry.append(binary)));
00112 #ifdef Q_OS_WIN
00113       if (!qfi->exists())
00114         qfi.reset(new QFileInfo(entry.append(".exe")));
00115
00116 #endif
00117       if (!qfi->isExecutable())
00118         continue;
00119
00120       ret = qfi->absoluteFilePath();
00121       break;
00122     }
00123   }
00124 #ifdef Q_OS_WIN
00125   if (ret.isEmpty()) {
00126     binary.remove(0, 1);
00127     binary.prepend("wsl ");
00128     QString out, err;
00129     if (Executor::executeBlocking(binary, {"--version"}, &out, &err) == 0 &&
00130         !out.isEmpty() && err.isEmpty())
00131       ret = binary;
00132   }
00133 #endif
00134
00135   return ret;
00136 }
00137
00142 bool Util::checkConfig() {
00143   return !QFile(QDir(QtPassSettings::getPassStore()).filePath(".gpg-id"))
00144               .exists() ||
00145          (QtPassSettings::isUsePass()
00146               ? !QtPassSettings::getPassExecutable().startsWith("wsl ") &&
00147                     !QFile(QtPassSettings::getPassExecutable()).exists()
00148               : !QtPassSettings::getGpgExecutable().startsWith("wsl ") &&
00149                     !QFile(QtPassSettings::getGpgExecutable()).exists());
00150 }
00151
00160 QString Util::getDir(const QModelIndex &index, bool forPass,
00161                      const QFileSystemModel &model,
00162                      const StoreModel &storeModel) {
00163   QString abspath =
00164       QDir(QtPassSettings::getPassStore()).absolutePath() + QDir::separator();
00165   if (!index.isValid())
00166     return forPass ? "" : abspath;
00167   QFileInfo info = model.fileInfo(storeModel.mapToSource(index));
00168   QString filePath =
00169       (info.isFile() ? info.absolutePath() : info.absoluteFilePath());
00170   if (forPass) {
00171     filePath = QDir(abspath).relativeFilePath(filePath);
00172   }
00173   filePath += QDir::separator();
00174   return filePath;
00175 }
00176
00182 void Util::copyDir(const QString &src, const QString &dest) {
00183   QDir srcDir(src);
00184   if (!srcDir.exists()) {
00185     return;
00186   }
00187   srcDir.mkpath(dest);
00188   foreach (QString dir, srcDir.entryList(QDir::Dirs | QDir::NoDotAndDotDot)) {
00189     copyDir(src + QDir::separator() + dir, dest + QDir::separator() + dir);
00190   }
00191
00192   foreach (QString file, srcDir.entryList(QDir::Files)) {
00193     QFile::copy(src + QDir::separator() + file,
00194                 dest + QDir::separator() + file);
00195   }
00196 }
00197
00198 const QRegularExpression &Util::endsWithGpg() {
00199   static const QRegularExpression expr{"\\.gpg$"};
00200   return expr;
00201 }
00202
00203 const QRegularExpression &Util::protocolRegex() {
00204   static const QRegularExpression regex{
00205       "((?:https?|ftp|ssh|sftp|ftps|webdav|webdavs)://\\S+)"};
00206   return regex;
```

```
00207 }
```

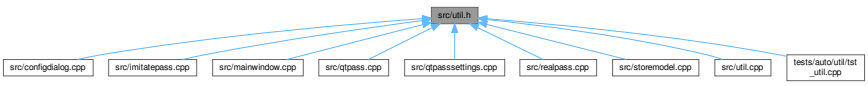
14.104 src/util.h File Reference

```
#include "storemodel.h"
#include <QFileSystemModel>
#include <QProcessEnvironment>
#include <QRegularExpression>
#include <QString>
```

Include dependency graph for util.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Util

  Some static utilities to be used elsewhere.

14.105 util.h

Go to the documentation of this file.

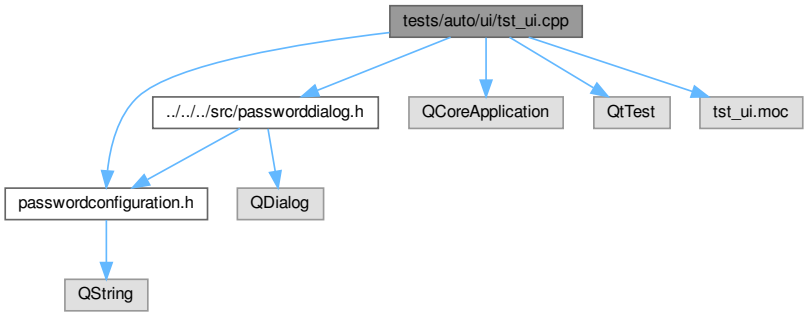
```
00001 #ifndef UTIL_H_
00002 #define UTIL_H_
00003
00004 #include "storemodel.h"
00005 #include <QFileSystemModel>
00006 #include <QProcessEnvironment>
00007 #include <QRegularExpression>
00008 #include <QString>
00009
00010 class StoreModel;
00011
00016 class Util {
00017 public:
00018   static QString findBinaryInPath(QString binary);
00019   static QString findPasswordStore();
00020   static QString normalizeFolderPath(QString path);
00021   static bool checkConfig();
```

```
00022   static QString getDir(const QModelIndex &index, bool forPass,
00023                         const QFileSystemModel &model,
00024                         const StoreModel &storeModel);
00025   static void copyDir(const QString &src, const QString &dest);
00026   static const QRegularExpression &endsWithGpg();
00027   static const QRegularExpression &protocolRegex();
00028
00029 private:
00030   static void initialiseEnvironment();
00031   static QProcessEnvironment _env;
00032   static bool _envInitialised;
00033 };
00034
00035 #endif // UTIL_H_
```

14.106 tests/auto/ui/tst_ui.cpp File Reference

```
#include "../../../src/passworddialog.h"
#include "passwordconfiguration.h"
#include <QCoreApplication>
#include <QtTest>
#include "tst_ui.moc"
```

Include dependency graph for tst_ui.cpp:



Classes

- class tst_ui

  The tst_ui class is our first unit test.

14.107 tst_ui.cpp

Go to the documentation of this file.

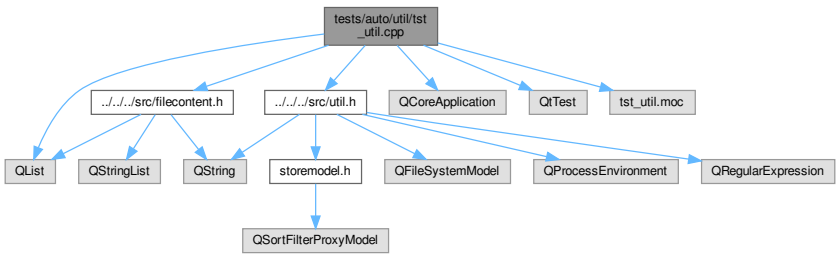
```
00001 #include "../../../src/passworddialog.h"
00002 #include "passwordconfiguration.h"
00003 #include <QCoreApplication>
00004 #include <QtTest>
00005
00009 class tst_ui : public QObject {
00010   Q_OBJECT
00011
00012 private Q_SLOTS:
00013   void contentRemainsSame();
00014 };
00015
```

```
00021 void tst_ui::contentRemainsSame() {
00022   QScopedPointer<PasswordDialog> d(
00023       new PasswordDialog(PasswordConfiguration{}, nullptr));
00024   d->setTemplate("", false);
00025   QString input = "pw\n";
00026   d->setPass(input);
00027   QCOMPARE(d->getPassword(), input);
00028
00029   d.reset(new PasswordDialog(PasswordConfiguration{}, nullptr));
00030   input = "pw\nname: value\n";
00031   d->setPass(input);
00032   QCOMPARE(d->getPassword(), input);
00033
00034   d.reset(new PasswordDialog(PasswordConfiguration{}, nullptr));
00035   d->setTemplate("name", false);
00036   d->setPass(input);
00037   QCOMPARE(d->getPassword(), input);
00038
00039   d.reset(new PasswordDialog(PasswordConfiguration{}, nullptr));
00040   d->setTemplate("name", true);
00041   d->setPass(input);
00042   QCOMPARE(d->getPassword(), input);
00043
00044   d.reset(new PasswordDialog(PasswordConfiguration{}, nullptr));
00045   d->setTemplate("", false);
00046   d->templateAll(true);
00047   d->setPass(input);
00048   QCOMPARE(d->getPassword(), input);
00049
00050   d.reset(new PasswordDialog(PasswordConfiguration{}, nullptr));
00051   d->setTemplate("", true);
00052   d->templateAll(true);
00053   d->setPass(input);
00054   QCOMPARE(d->getPassword(), input);
00055
00056   d.reset(new PasswordDialog(PasswordConfiguration{}, nullptr));
00057   d->setTemplate("name", true);
00058   d->templateAll(true);
00059   d->setPass(input);
00060   QCOMPARE(d->getPassword(), input);
00061 }
00062
00063 QTEST_MAIN(tst_ui)
00064 #include "tst_ui.moc"
```

14.108 tests/auto/util/tst_util.cpp File Reference

```
#include "../../../src/filecontent.h"
#include "../../../src/util.h"
#include <QCoreApplication>
#include <QList>
#include <QtTest>
#include "tst_util.moc"
```

Include dependency graph for tst_util.cpp:



Classes

- class tst_util

The tst_util class is our first unit test.

Functions

- bool operator== (const NamedValue &a, const NamedValue &b)

14.108.1 Function Documentation

14.108.1.1 operator==()

```
bool operator== (
            const NamedValue & a,
            const NamedValue & b )
```

Definition at line 28 of file tst_util.cpp.

14.109 tst_util.cpp

Go to the documentation of this file.

```
00001 #include "../../../src/filecontent.h"
00002 #include "../../../src/util.h"
00003 #include <QCoreApplication>
00004 #include <QList>
00005 #include <QtTest>
00006
00010 class tst_util : public QObject {
00011   Q_OBJECT
00012
00013 public:
00014   tst_util();
00015   ~tst_util() override;
00016
00017 public Q_SLOTS:
00018   void init();
00019   void cleanup();
00020
00021 private Q_SLOTS:
00022   void initTestCase();
00023   void cleanupTestCase();
00024   void normalizeFolderPath();
00025   void fileContent();
00026 };
00027
00028 bool operator==(const NamedValue &a, const NamedValue &b) {
00029   return a.name == b.name && a.value == b.value;
00030 }
00031
00035 tst_util::tst_util() = default;
00036
00040 tst_util::~tst_util() = default;
00041
00045 void tst_util::init() {}
00046
00050 void tst_util::cleanup() {}
00051
00055 void tst_util::initTestCase() {}
00056
00060 void tst_util::cleanupTestCase() {}
00061
00066 void tst_util::normalizeFolderPath() {
00067   QCOMPARE(Util::normalizeFolderPath("test"),
00068            QDir::toNativeSeparators("test/"));
00069   QCOMPARE(Util::normalizeFolderPath("test/"),
00070            QDir::toNativeSeparators("test/"));
```

```
00071 }
00072
00073 void tst_util::fileContent() {
00074   NamedValue key = {"key", "val"};
00075   NamedValue key2 = {"key2", "val2"};
00076   QString password = "password";
00077
00078   FileContent fc = FileContent::parse("password\n", {}, false);
00079   QCOMPARE(fc.getPassword(), password);
00080   QCOMPARE(fc.getNamedValues(), {});
00081   QCOMPARE(fc.getRemainingData(), QString());
00082
00083   fc = FileContent::parse("password", {}, false);
00084   QCOMPARE(fc.getPassword(), password);
00085   QCOMPARE(fc.getNamedValues(), {});
00086   QCOMPARE(fc.getRemainingData(), QString());
00087
00088   fc = FileContent::parse("password\nfoobar\n", {}, false);
00089   QCOMPARE(fc.getPassword(), password);
00090   QCOMPARE(fc.getNamedValues(), {});
00091   QCOMPARE(fc.getRemainingData(), QString("foobar\n"));
00092
00093   fc = FileContent::parse("password\nkey: val\nkey2: val2", {"key2"}, false);
00094   QCOMPARE(fc.getPassword(), password);
00095   QCOMPARE(fc.getNamedValues(), {key2});
00096   QCOMPARE(fc.getRemainingData(), QString("key: val"));
00097
00098   fc = FileContent::parse("password\nkey: val\nkey2: val2", {"key2"}, true);
00099   QCOMPARE(fc.getPassword(), password);
00100   QCOMPARE(fc.getNamedValues(), NamedValues({key, key2}));
00101   QCOMPARE(fc.getRemainingData(), QString());
00102 }
00103
00104 QTEST_MAIN(tst_util)
00105 #include "tst_util.moc"
```

Index