Pext Documentation

Release v0.11.1+dev

Sylvia van Os

Jan 27, 2018

General information:

CHAPTER 1

Pext

1.1 Contents

  - *Windows*
- *License*

1.2 Community

If you need support or just want to chat with our community, we have the following options:

- IRC: #pext on OFTC (webchat)
- Matrix: #pext:matrix.org (webchat)
- Telegram: @PextTool

All these channels are linked to each other, so there is no need to worry about missing out.

We can also be reached on Twitter: @PextTool

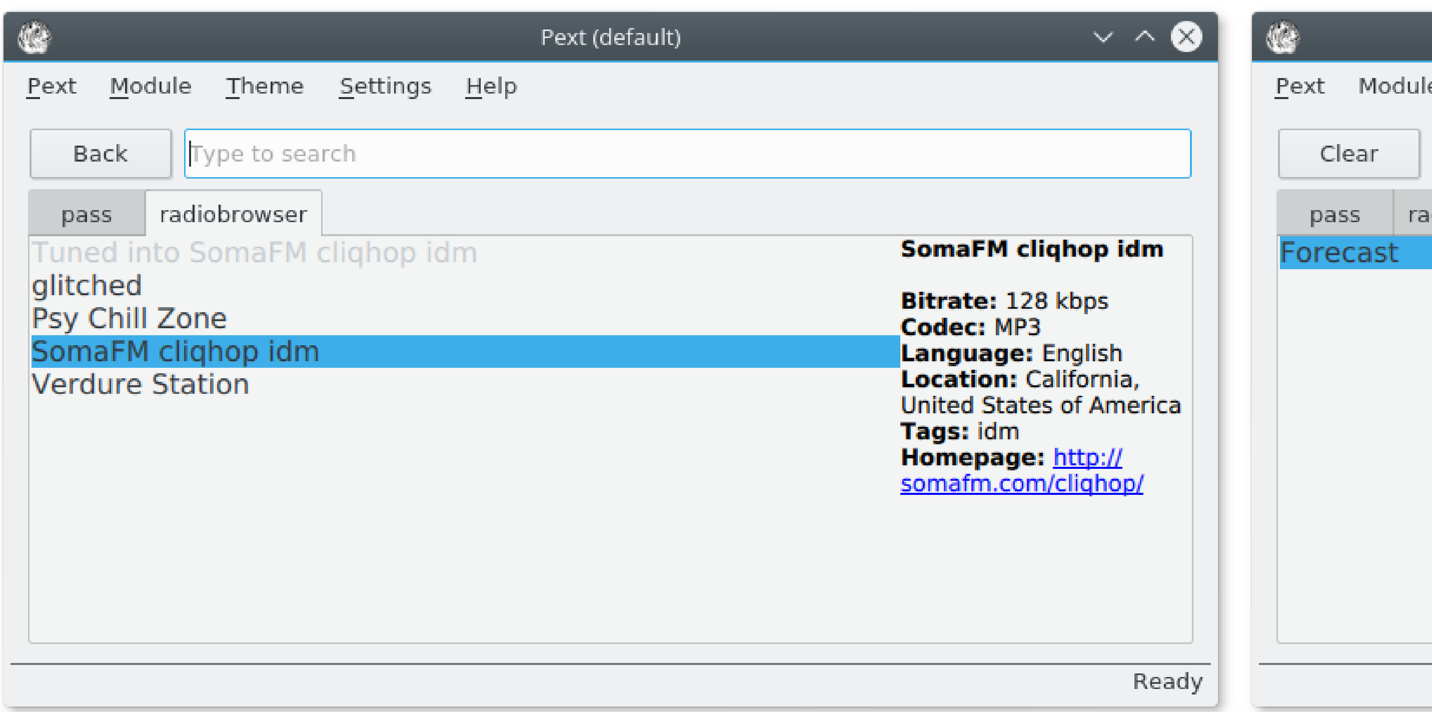
1.3 Introduction

Pext stands for **P**ython-based **ex**tendable **t**ool. It is built using Python 3 and Qt5 QML and has its behaviour decided by modules. Pext provides a simple window with a search bar, allowing modules to define what data is shown and how it is manipulated.

For example, say you want to use Pext as a password manager. You load in the pass module, and it will show you a list of your passwords which you can filter with the search bar. When you select a password in the list, it will copy the password to your clipboard and Pext will hide itself, waiting for you to ask for it again.

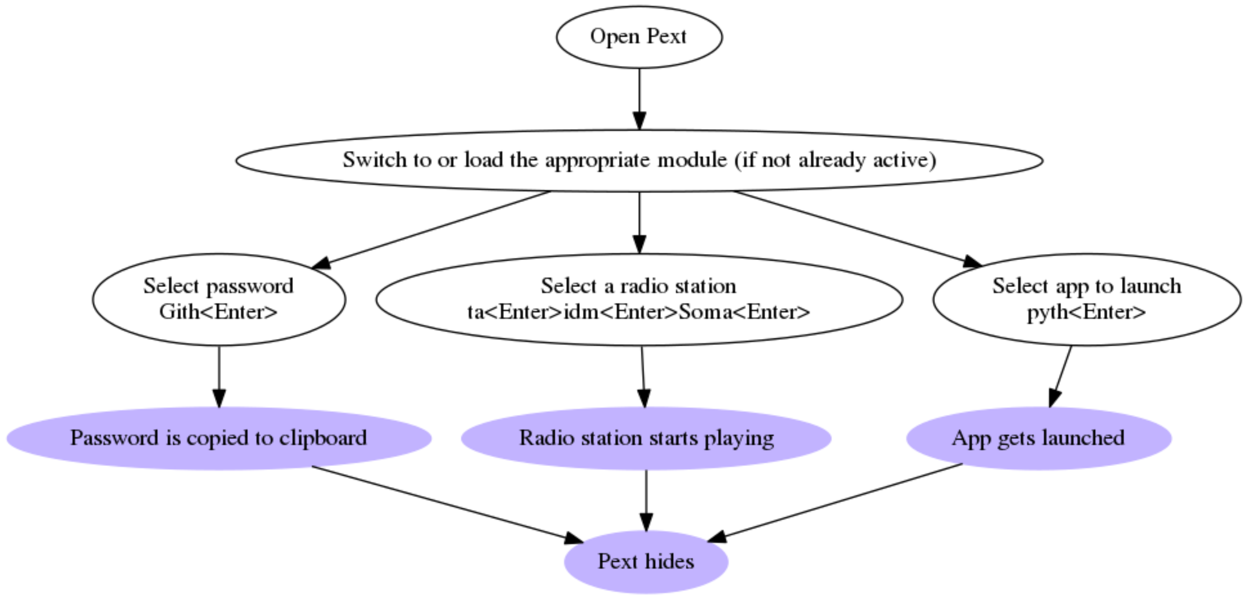
Depending on the module you choose, what entries are shown and what happens when you select an entry changes. So choose the module appropriate for what you want to do, and Pext makes it easy.

Several modules are available for effortless install right within Pext.



1.4 How it works

Pext is designed to quickly pop up and get out of your way as soon as you're done with something. It is recommended to bind Pext to some global hotkey, or possibly run multiple instances of it with different profiles under multiple hotkeys. Example Pext workflows look as follows:



Simply put:

- Open (Pext)
- Search (for something)
- Select (with Enter)
- Hide (automatically)

1.5 Installation

Note: If you run into any issues, please check out the troubleshooting section near the end of this document before reporting a bug.

1.5.1 GNU/Linux

Arch

The Arch packages are maintained by Ivan Semkin

For the stable version:

```
$ pacaur -S pext
```

For the git version:

```
$ pacaur -S pext-git
```

Other distros

For the stable version:

```
$ pip3 install pext --user
```

For the git version:

```
$ pip3 install git+https://github.com/Pext/Pext.git --user
```

On some systems, you may need to use pip instead of pip3.

Alternatively, you can install Pext from source

1.5.2 macOS

See Installing Pext from source

1.5.3 Windows (experimental)

See Installing Pext from source

1.6 Usage

To actually use Pext, you will first have to install one or more modules. Check out the Pext organisation on GitHub or use `Module` -> `Install module` -> `From online module list` in the application for a list of modules.

After installating at least one module, you can load it from the `Module` -> `Load module` menu. After that, experiment! Each module is different.

For command line options, use `--help`.

1.7 Hotkeys

1.7.1 Entry management

- Escape: Go one level up
- Tab: Tab-complete the current input
- Enter / Left mouse button: Select entry or run command
- Ctrl+Shift+. / Right mouse button on header: Open state menu
- Ctrl+. / Right mouse button on any item: Open context menu
- Ctrl+J / Down arrow: Go one entry down
- Ctrl+H / Up arrow: Go one entry up

- Ctrl+F / Page down: Go one page down
- Ctrl+B / Page up: Go one page up

1.7.2 Tab management

- Ctrl+T: Open new tab
- Ctrl+W: Close current tab
- Ctrl+Tab: Switch to next tab
- Ctrl+Shift+Tab: Switch to previous tab
- Alt+`<number>`: Switch to tab `<number>`
- F5: Reload tab, including code changes to the module

1.7.3 Session management

- Ctrl+Q: Quit and save the currently loaded modules and settings to the profile
- Ctrl+Shift+Q: Quit without saving to the profile

1.8 Troubleshooting

1.8.1 GNU/Linux

Installing module dependencies fails

Your distribution may ship with an outdated version of pip. Run `pip install --upgrade pip` (possibly as root) in a terminal.

Pext's window is completely white

The proprietary NVIDIA driver is known to cause this issue on at least Ubuntu. You can work around this by running `sudo apt-get install python3-opengl`.

Pext user report: https://github.com/Pext/Pext/issues/11Ubuntu bug: https://bugs.launchpad.net/ubuntu/+source/python-qt4/+bug/941826

1.8.2 macOS

I cannot brew/pip install anymore

The Homebrew team completely broke pip's –target flag, which Pext depends on. To work around this, Pext automatically creates a `~/.pydistutils.cfg` file which resets the broken Homebrew pip defaults and deletes this file after its done installing module dependencies.

As a side effect, this means that using brew install or pip install while Pext is installing module dependencies may fail. If you cannot use brew install or pip install at all anymore after Pext crashed, please delete `~/.pydistutils.cfg` if it exists.

The Homebrew team refuses to fix this issue: https://github.com/Homebrew/brew/issues/837

1.8.3 Windows

The python or pip commands do not work/The PATH variable is wrong

In the installer, make sure that 'Include Python in PATH' or similar is checked. Then after the installation, start a new command prompt and type `python -V` or `pip3` to check if it was properly installed. If the version number and the help message are returned respectively, you are good to go further. If not, in case you already had `cmd.exe` open, restart it or execute `refreshenv` to reload environment variables. If it still does not work yet, check if the PATH was set in the GUI or manually with `cmd.exe`.

GUI:

- Start Menu > Computer (right click) > Properties > Advanced System Settings > Environment Variables
- Check the PATH for both the system and the current user and in one of them the Python installation directory should be present, which is normally `C:\Python36` and `C:\Python36\Scripts`

cmd.exe:

- Run `path` or `echo %PATH%` to check if the directory (`C:\Python36` and `C:\Python36\Scripts`) is included
- The path can then be set with `setx` but because the possibility for truncation and the merging of users and system path, the gui method is to be preferred. (more details: https://stackoverflow.com/questions/9546324/adding-directory-to-path-environment-variable-in-windows)

1.9 License

Pext is licensed under the GNU GPLv3+, with exception of artwork and documentation, which are licensed under the Creative Commons Attribution Share-Alike 4.0 license.

Under artwork and documentation fall:

- All files in the following directories:
  - docs/
  - pext/images/
  - screenshots/
  - .github/
- All Markdown files in the root directory.
- logo.png

When attributing the logo (which was donated by vaeringjar), it should be attributed as the Pext Logo by White Paper Fox. Please link to Pext with https://github.com/Pext/Pext or https://pext.hackerchick.me/ and to White Paper Fox with http://www.whitepaperfox.com/ where possible.

CHAPTER 2

Change Log

All notable changes to this project will be documented in this file.

The format is based on Keep a Changelog and this project adheres to Semantic Versioning.

2.1 [Unreleased]

2.1.1 Added

- Support renaming profiles
- Switching profile from the GUI
- Opening a second instance with another profile from the GUI
- Basic profile management from the GUI
- Ability to change language through the UI
- `--list-locales` argument to show supported languages

2.1.2 Changed

- Profile name is no longer displayed if default
- Trying to create a profile that already exists throws an error
- Trying to delete a profile that is currently in use throws an error
- Use argparse for argument parsing instead of getopt
- Update checking now happens if the last check was over 24 hours, instead of each app launch
- Combine all menu groups in settings for organizational purposes
- Relicensed documentation under CC BY-SA 4.0

2.1.3 Fixed

- Pext crash when module tries to empty context_menu_base
- Inconsistent font sizing
- Improved main screen resizing and logo showing

2.1.4 Removed

- Removed manpage

2.2 [0.11.1] - 2017-12-19

2.2.1 Packaging

- Fix missing translation files

2.3 [0.11] - 2017-12-19

2.3.1 Packaging changes

- Pext now depends on dulwich
- Pext no longer depends on pygit2

2.3.2 Translation updates

- Added Norwegian Bokmål (thanks, Allan Nordhøy!)
- Update Chinese (Traditional) translation
- Update Spanish translation
- Update Hungarian translation
- Update Dutch translation

2.3.3 Fixed

- Ubuntu/Debian compatibility for git operations over HTTPS
- Install module from URL screen not working (regression from adding theming support for 0.9)
- Theme selector now correctly displays current theme before switching
- Pext no longer creates an empty theme file for the system theme and doesn't show it in the list of themes

2.4 [0.10] - 2017-11-11

2.4.1 Packaging changes

- Pext now depends on pygit2, which uses libgit2, instead of git

2.4.2 API changes

- Bump API version to 0.7.0
- Add set_entry_info queue call
- Add replace_entry_info_dict queue call
- Add set_command_info queue call
- Add replace_command_info_dict queue call
- Add set_base_info queue call
- Add set_entry_context queue call
- Add replace_entry_context_dict queue call
- Add set_command_context queue call
- Add replace_command_context_dict queue call
- Add set_base_context queue call
- Add extra_info_request function
- Add a none SelectionType
- Made more parameters optional

2.4.3 Added

- Add info panels which modules can use to show extra info on the current status on selected entry
- Add context panels for state changes and extra actions for entries/commands
- Traceback is now printing when an exception is triggered
- Last updated info for modules
- Version info for modules
- Windows support
- Support for checking for updates (stable versions only)

2.4.4 Changed

- Command mode no longer locks onto the first entry
- Commands are always displayed in italics, instead of using italics for whatever is unfocused
- Versioning is now more precise
- Check if a module/theme has an update before updating it

- Pext now auto-restarts after changing the theme
- Pext now displays less broken when the height is higher than the width
- Removed tray menu because it can't be translated due to PyQt limitations
- Make clicking the tray icon toggle visibility on macOS
- Minimizing normally after Pext is done is now the default on all platforms
- Module requesting window hide will only reset the selection of that module instead of all
- The –exit option got removed, Pext now will only start the UI if no options were given or all options were session-related

2.4.5 Fixed

- Regression introduced in 0.9 which could cause selections to trigger wrongly when emptying the search bar
- Page up and down causing QML errors when used too close to the start/end of the list
- Minimizing behaviour didn't always work
- Git commands are now properly limited to Pext directories
- Desktop notifications now also show when Pext is minimized normally
- Modules no longer lock up Pext while making a selection
- Direct Git URL clone ending in / no longer creates an undeletable module
- Modules now always properly get localization info
- Ugly line between entries and entry info in some themes
- No themes available dialog now correctly shows
- Modules can't crash Pext by throwing an exception on stopping on Pext exit

2.5 [0.9] - 2017-08-23

2.5.1 API changes

- Whenever the state changes (either by the user going back, selecting something or set_selection being called), the queue is now emptied
- ask_input and ask_input_password now ask for a prefill before the identifier

2.5.2 Translation updates

- Added traditional Chinese (thanks, Jeff Huang!)
- Added Spanish (thanks, Emily Lau!)
- Updated Dutch (thanks, Heimen Stoffels!)

2.5.3 Added

- Theming support based on QPalette
- UI option to choose minimizing behaviour
- UI option to choose sorting behaviour
- UI toggle to enable/disable tray icon
- –background command line option to make Pext not launch/foreground the UI

2.5.4 Changed

- The design philosophy is now explained in the empty state screen
- pyqt5 is added as install_requires
- The about dialog now thanks translators
- Info-only CLI parameters will no longer launch Pext as well (–help, –version, –list-styles, –list-modules, –list-themes)
- Closing the main window will now quit Pext and save state

2.5.5 Fixed

- pext_dev's generated base file now leaves the copyright open for the author to fill in
- Not being able to select an entry until the list is fully loaded
- Selection constantly resetting while items are being added
- Loading and reloading a module while text is in search now applies the filter correctly
- Fix crash in command mode when pressing enter while hovering over a wrong entry

2.6 [0.8] - 2017-04-28

2.6.1 API changes

- The settings variable now contains _api_version ([major, minor, patch]) and _locale by default
- Queue requests that cause process_response to be called can now optionally give an identifier to receive when process_response is called
- Modules must now declare their settings in metadata.json

2.6.2 Added

- Simple pext_dev command to help module development
- Support metadata.json for showing info on installed modules
- i18n support
- Dutch translation

2.6.3 Changed

- Move all UI code to QML
- Improved installation dialogs
- Improve load module dialog
- Get rid of update and uninstall dialogs in favor for a central module management dialog
- Check module functions parameter length on module load to prevent some runtime crashes for modules
- Module settings is no longer a freeform input field
- Display "Waiting" instead of "Ready" in the statusbar when not processing and the active module has not sent anything yet

2.6.4 Fixed

- Crash when picking a command while there are also other entries to display

2.6.5 Removed

- config.ini for editing Pext config directory (use $XDG_CONFIG_HOME or $HOME instead)

2.7 [0.7] - 2017-04-10

2.7.1 Added

- Clear/back button in the UI

2.7.2 Changed

- Minor UI font size changes
- Pext's QML now uses StandardKeys in most places

2.7.3 Fixed

- Fix Debian detection (no longer incorrectly detects openSUSE as Debian)
- Fix nonsense load/update/uninstall dialogs if no modules are installed

2.8 [0.6.1] - 2017-04-01

2.8.1 Fixed

- Clicking the tray icon no longer toggles visibility on macOS
- XDG_CONFIG_HOME is now correctly used when available

- The environment is no longer cleared when doing the initial git clone (security: the old behaviour would cause a proxy defined in the environment to be ignored)

2.9 [0.6] - 2017-03-27

2.9.1 Added

- Install dependencies automatically if the module provides a requirements.txt file

2.9.2 Fixed

- If module installation fails, the module directory is removed, so a subsequent installation doesn't instantly fail
- Modules are now correctly unloaded when they raise a critical error
- Added workaround for Ubuntu systems running the proprietary nvidia driver (https://github.com/Pext/Pext/issues/11)

2.10 [0.5] - 2017-03-22

2.10.1 API changes

- Remove Action.notify_message and Action.notify_error, which are synonyms for add_message and add_error

2.10.2 Added

- Documentation
- Repository for third-party modules

2.10.3 Changed

- Give more information upon installing modules and warn the user that they are code
- User commands will now be auto-completed to the selected command

2.10.4 Fixed

- Files unexpectedly existing in ~/.config/pext/modules/ no longer causes a crash

2.11 [0.4.1] - 2017-03-05

2.11.1 Changed

- The default window is no longer explicitly borderless
- The logo now has a white background which improves readability on dark themes

2.11.2 Fixed

- An error occuring when retrieving the list of downloadable modules no longer causes a crash
- Selecting a command entry after the entry list no longer causes a crash

2.12 [0.4] - 2017-02-20

2.12.1 Added

- Basic Qt5 theming support using installed system themes
- Allow for a pext/config.ini file to overwrite some default configuration
- Allow the user to disable tray icon creation
- Add entry to open homepage from help menu (so the user can find support)

2.12.2 Changed

- Get list of installable modules from pext.hackerchick.me instead of pext.github.io

2.13 [0.3] - 2016-12-29

2.13.1 API changes

- The entry and command list will now be emptied each time just before selection_made is called

2.13.2 Added

- Busy indicator when the list of entries is empty for a more responsive look
- Support for getting a list of installable modules from pext.github.io

2.13.3 Fixed

- All commands now correctly show up after emptying the search bar
- Module lists are now sorted alphabetically
- selection_made is no longer unnecessarily triggered when closing the window

2.14 [0.2] - 2016-11-21

2.14.1 Added

- System tray icon

2.15 [0.1] - 2016-11-19

Initial release

CHAPTER 3

Pext module development

3.1 Setting up the environment

Setting up the development environment is very easy. Simply installing Pext as per the *README* will also install pext_dev, containing all you need to easily develop modules.

Once you have installed Pext, simply navigate to the directory you want to start developing in and run `pext_dev init` to create the base files in the current directory or `pext_dev init <directory>` to create them in a new directory.

3.2 Starting module development

After running `pext_dev init` and answering its questions you will have a directory with the following files in it:

- __init__.py
- metadata.json
- LICENSE

The generated *__init__.py* file is the main entry point to your Pext module and the main and in many cases only file to edit. For editing, you can choose any editor you like, so make sure to choose one that's comfortable for you to work in.

If you open this file, you will see a few imports and a class named *Module* which contains 4 functions. These are the core of any Pext module and you need to fill these in with the Python 3 code you want to use. For more information about the exact purpose of each of these functions, see *helpers/pext_base.py*.

The *metadata.json* file contains general information on your module, used by Pext to show the user who developed the module, its intended purpose and more, both when the user is about to install the module and when they already installed it.

LICENSE contains the license for your project. `pext_dev init` puts the GPLv3 into this file, as it defaults to using the GPLv3+ for generated projects, because Pext itself is licensed under this license. The GPLv3 is a copyleft license which is meant to allow people to use, modify and distribute modified versions, as long as they do so under the same license. This allows for a healthy ecosystem where Pext modules can be improved by anyone, even if they are not the original author, and stay Free and Open Source Software.

3.3 Additional requirements

Sometimes you may want to use some Python libraries that don't come with Python itself. In this case, you may place a file named *requirements.txt* in your module's directory, listing one Python module you want per line. Python modules can be found on PyPI. Make sure you check the license of modules you want to use for compatibility with your module's license.

More advanced information on using a *requirements.txt* file can be found on https://pip.readthedocs.io/en/latest/reference/pip_install/#requirements-file-format.

3.4 Testing

To test your module, simply run `pext_dev run` in the module directory. This will launch a completely clean instance of Pext and install your module from scratch, including dependencies defined in *requirements.txt*. This way you can be reasonably sure your module will work for others too. When you're done testing, simply close the Pext instance that popped up and pext_dev will clean everything up again.

3.5 Publishing your module

To publish your module, put it on a git hosting site such as NotABug. Make sure to add a README file so users who find your module online know what it's about!

Whenever you make a change to your module, you can push it to git and, as long as it's on the master branch, Pext will update to the new version as soon as the user asks for module updates. Simple as that.

If you want your module to be listed in Pext under *Other Developers*, please get in touch.

CHAPTER 4

helpers/pext_base.py

Pext Module Base.

This file contains the definition of the Pext module base, which all Pext modules must implement. This is basically the API of Pext.

class pext_base.**ModuleBase**

Introduced in API version 0.1.0.

The base all Pext modules must implement.

extra_info_request (*selection: typing.List[typing.Dict[pext_helpers.SelectionType, str]]*) → None

Introduced in API version 0.3.0.

Called when the user selects a different entry.

The syntax of selection is the same as in selection_made.

If desired, this function could be used to update the current entry by putting a new set_info request in the queue.

init (*settings: typing.Dict*, *q: queue.Queue*) → None

Introduced in API version 0.1.0.

Called when the module is first loaded.

In this function, the application should initialize all its data and use Action.replace_entry_list and Action.replace_command_list to populate the main list. Please avoid using Action.add_entry and Action.add_command, especially at initialization time, unless it would take unreasonably long for the module to load otherwise, as these methods are significantly slower if there are a lot of entries to add.

The settings variable is a dictionary containing all "module settings". For example, if the user enters "foo=bar foobar=fubar" in the custom module settings dialog, this dictionary will have {"foo": "bar", "foobar": "fubar"} as values.

The settings variable also contains special Pext settings, starting with a single underscore: - _api_version: The API version as [major, minor, patch] - _locale: The current Pext locale

The q variable contains the queue that actions can be put it. It is very important to keep a reference to this variable so that you can do anything on the UI at all.

process_response (*response: typing.Union[bool, str], identifier: typing.Any*)
Introduced in API version 0.1.0.

Process a response to a requested action.

Called when a response is given as a result of an Action being put into the queue. Not all Actions return a response.

When no specific identifier was given in a queue request, identifier is called with None. Otherwise, it is called with the identifier previously put in the queue.

selection_made (*selection: typing.List[typing.Dict[pext_helpers.SelectionType, str]]*) → None
Introduced in API version 0.1.0.

Called when the user makes a selection.

The selection variable contains a list of the selection tree and the type, which can be either entry or command.

For example, if the user chooses the entry "Audio settings" in the main screen, the value of selection is [{type: SelectionType.entry, value: "Audio settings"}]. If the user then runs the command "volume 50", this function is called again, with the value of selection being [{type: SelectionType.entry, value: "Audio settings"}, {type: SelectionType.command, value: "volume 50"}].

stop ()
Introduced in API version 0.1.0.

Called when the module gets unloaded.

If necessary, the module should clean itself up nicely.

CHAPTER 5

helpers/pext_helpers.py

Pext Helpers.

This file contains various functionality that is relevant to both Pext and modules and helps keep the API consistent.

class pext_helpers.**Action**

Introduced in API version 0.1.0.

The list of actions a module can request.

A module can request any of these actions of the core by putting it in the queue. All of these actions need to be accompanied by a list of arguments. In these examples, we assume that you have assigned the queue variable to self.q, a common practice in Pext modules.

critical_error Introduced in API version 0.1.0.

Show an error message on the screen and unload the module. This function is also called when the module throws an exception.

message – error message to show

Example: self.q.put([Action.critical_error, "Something went wrong!"])

add_message Introduced in API version 0.1.0.

Show a message on the screen.

message – message to show

Example: self.q.put([Action.add_message, "We did a thing"])

add_error Introduced in API version 0.1.0.

Show an error message on the screen.

message – error message to show

Example: self.q.put([Action.add_error, "We did a thing, but it went wrong"])

add_entry Introduced in API version 0.1.0.

Add an entry to the entry list.

entry – the entry

Example: self.q.put([Action.add_entry, "Audio settings"])

prepend_entry Introduced in API version 0.1.0.

Prepend an entry to the entry list.

entry – the entry

Example: self.q.put([Action.prepend_entry, "Audio settings"])

remove_entry Introduced in API version 0.1.0.

Remove an entry from the entry list.

entry – the entry

Example: self.q.put([Action.remove_entry, "Audio settings"])

replace_entry_list Introduced in API version 0.1.0.

Replace the list of entries with the given list.

list – the new list of entries

Example: self.q.put([Action.replace_entry_list, ["Audio settings", "Video settings"]])

add_command Introduced in API version 0.1.0.

Add an entry to the command list.

entry – the entry

Example: self.q.put([Action.add_command, "download"])

prepend_command Introduced in API version 0.1.0.

Prepend an entry to the command list.

entry – the entry

Example: self.q.put([Action.prepend_command, "download"])

remove_command Introduced in API version 0.1.0.

Remove a command from the entry list.

entry – the entry

Example: self.q.put([Action.remove_command, "download"])

replace_command_list Introduced in API version 0.1.0.

Replace the list of commands with the given list.

list – the new list of entries

Example: self.q.put([Action.replace_command_list, ["download", "upload"]])

set_header Introduced in API version 0.1.0.

Set or replace the text currently in the header bar.

If header is not given, the header will be removed.

header – the new header text

Example: self.q.put([Action.set_header, "Weather for New York"])

set_filter Introduced in API version 0.1.0.

Replace the text currently in the search bar.

filter – the new text to put in the search bar

Example: self.q.put([Action.set_header, “Weather for New York”])

ask_question_default_yes Introduced in API version 0.1.0.

Ask a yes/no question, with the default value being yes.

question – the question to ask identifier – an optional identifier which gets passed back to process_response

Example: self.q.put([Action.ask_question_default_yes, “Are you sure you want to continue?”, 0])

ask_question_default_no Introduced in API version 0.1.0.

Ask a yes/no question, with the default value being no.

question – the question to ask identifier – an optional identifier which gets passed back to process_response

Example: self.q.put([Action.ask_question_default_no, “Are you sure you want to continue?”, 0])

ask_input Introduced in API version 0.1.0. Changed in API version 0.2.0.

Ask the user to input a single line of text.

text – the text to show the user prefill – the text to already put into the input field identifier – an optional identifier which gets passed back to process_response

Example: self.q.put([Action.ask_input, “Please choose a new name for this entry”, “Example name”, 0])

ask_input_password: Introduced in API version 0.1.0. Changed in API version 0.2.0.

Ask the user to input a single line of text into a password field.

text – the text to show the user prefill – the text to already put into the input field (hidden behind asterisks, of course) identifier – an optional identifier which gets passed back to process_response

Example: self.q.put([Action.ask_input_password, “Please enter your password”, “Current password”, 0])

ask_input_multi_line Introduced in API version 0.1.0.

Ask the user to input one or more lines of text.

text – the text to show the user prefill – the text to already put into the input field identifier – an optional identifier which gets passed back to process_response

The prefill may contain newline characters.

Example: self.q.put([Action.ask_input_multi_line, “List your favourite animals”, “Cat and dog”, 0])

copy_to_clipboard Introduced in API version 0.1.0.

Copy data to the clipboard.

text – the text to copy to the clipboard

Example: self.q.put([Action.copy_to_clipboard, “I like Pext”])

set_selection Introduced in API version 0.1.0.

Change the internal Pext selection for this module.

The internal Pext selection contains a list of all options and commands the user chose and typed since the last time the window was closed and looks something like this: [{type: SelectionType.entry, value: “Audio settings”}, {type: SelectionType.command, value: “volume 50”}].

To go a single level up, simply remove the last entry from this list. To reset to the main screen, use an empty list.

After set_selection is called, selection_made in ModuleBase will be called with the new values.

list – the selection list

Example: self.q.put([Action.set_selection, [{type: SelectionType.entry, value: "Audio settings"}])

close: Introduced in API version 0.1.0.

Close the window.

Call this when the user is done. For example, when the user made a selection.

Example: self.q.put([Action.close])

set_entry_info: Introduced in API version 0.3.1.

Set additional info for a certain entry, either in plain text or HTML.

key – the entry to set it for value – the value to set it to

Example: self.q.put([Action.set_entry_info, "Audio settings", "Change the audio settings")

replace_entry_info_dict: Introduced in API version 0.5.

Set all entry info at once by passing a dictionary.

Example: self.q.put([Action.set_entry_info,

{"Audio settings": "Change the audio settings", "Video settings": "Change the video settings"}])

set_command_info: Introduced in API version 0.3.1.

Set additional info for a certain command, either in plain text or HTML.

key – the command to set it for value – the value to set it to

Example: self.q.put([Action.set_command_info, "volume", "Set the volume to the desired percentage (0 - 100)")

replace_command_info_dict: Introduced in API version 0.5.

Set all command info at once by passing a dictionary.

Example: self.q.put([Action.set_entry_info,

{"volume": "Set the volume to the desired percentage (0 - 100)", "video": "Turn video on or off"}])

set_base_info: Introduced in API version 0.6.

Set an info block to always show regardless of the active selection.

Example: self.q.put([Action.set_base_info, "Type stop to stop listening to radio"])

set_entry_context: Introduced in API version 0.4.

Add a context menu to a certain entry.

key – the entry to set it for value – the list of context entries

Example: self.q.put([Action.set_entry_context, "Audio settings", ["Disable", "Decrease volume", "Increase volume"])

replace_entry_context_dict: Introduced in API version 0.5.

Set all entry context menu entries at once by passing a dictionary.

Example: self.q.put([Action.replace_entry_context,

{"Audio settings": ["Disable"], "Video quality": ["High", "Low"]}

set_command_context: Introduced in API version 0.4.

Add a context menu to a certain command.

key – the command to set it for value – the value to set it to

Example: self.q.put([Action.set_command_context, "volume", ["0%", "20%", "40%", "60%", "80%", "100%"])

replace_command_context_dict: Introduced in API version 0.5.

Set all command context menu entries at once by passing a dictionary.

Example: self.q.put([Action.replace_command_context_dict,

{"volume": ["0%", "20%", "40%", "60%", "80%", "100%"], "video": ["on", "off"]}])

set_base_context: Introduced in API version 0.6.

Set the base context, reachable by right-clicking the header text or Ctrl+Shift+..

Example: self.q.put([Action.set_base_context, ["Mute", "Stop"]])

class `pext_helpers.`**`SelectionType`**

Introduced in API version 0.1.0.

A list of possible selection types.

entry Introduced in API version 0.1.0.

An entry in the entry list was chosen.

command Introduced in API version 0.1.0.

A valid command was typed (valid commands start with an entry in the command list).

none Introduced in API version 0.6.

The selection is not relevant to any entry or command.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

p

Index