HumBLE Explorer Documentation

Release 0.5.0.post1.dev3+g1a4c32a

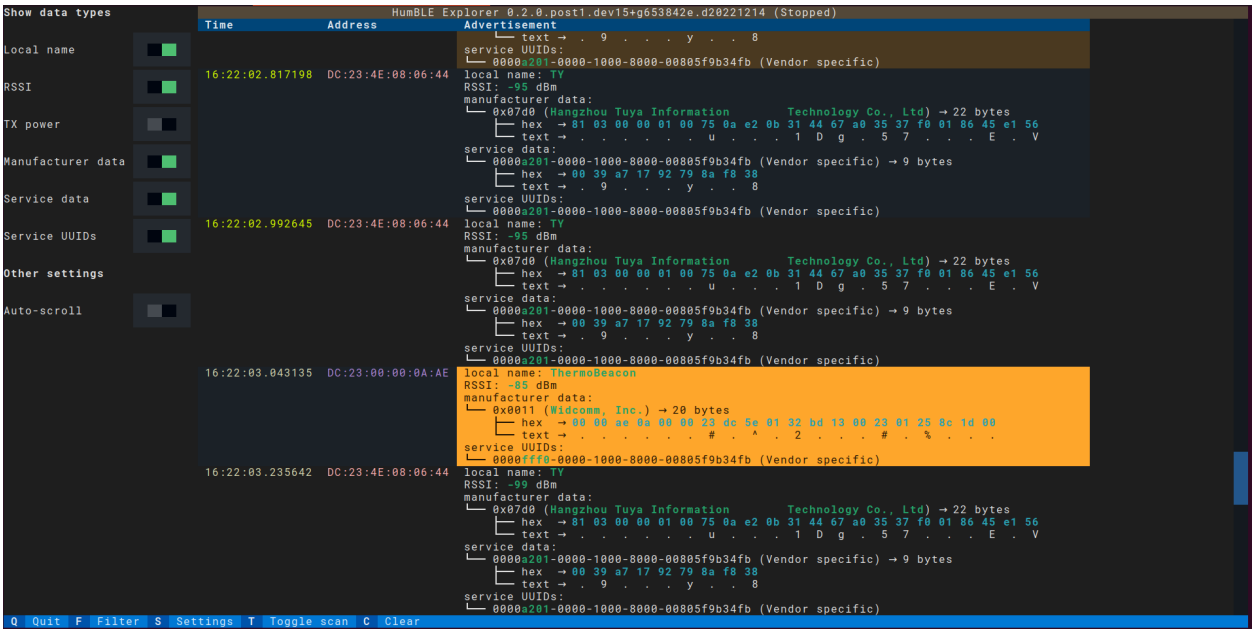
Koen Vervloesem

Jun 29, 2023

CONTENTS

Human-friendly Bluetooth Low Energy Explorer

This is a cross-platform (Windows, Linux, macOS) human-friendly program to scan for Bluetooth Low Energy (BLE) advertisements on the command line. It's mostly useful for people who develop BLE software or want to debug problems with BLE devices.



CHAPTER

ONE

CONTENTS

1.1 Usage

HumBLE Explorer runs on the command line with an interactive Text User Interface (TUI).

1.1.1 Installation

You can install HumBLE Explorer as a pip package from PyPI:

```
pip install humble-explorer
```

1.1.2 Command-line arguments

You can find all command-line arguments by running HumBLE Explorer with the `--help` option:

```
$ humble-explorer  --help
usage: humble-explorer [-h] [--version] [-a ADAPTER] [-s {active,passive}]

Human-friendly Bluetooth Low Energy Explorer

options:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -a ADAPTER, --adapter ADAPTER
                        Bluetooth adapter (e.g. hci1 on Linux)
  -s {active,passive}, --scanning-mode {active,passive}
                        Scanning mode (default: active)
  -m, --macos-use-address
                        Use Bluetooth address instead of UUID on macOS
```

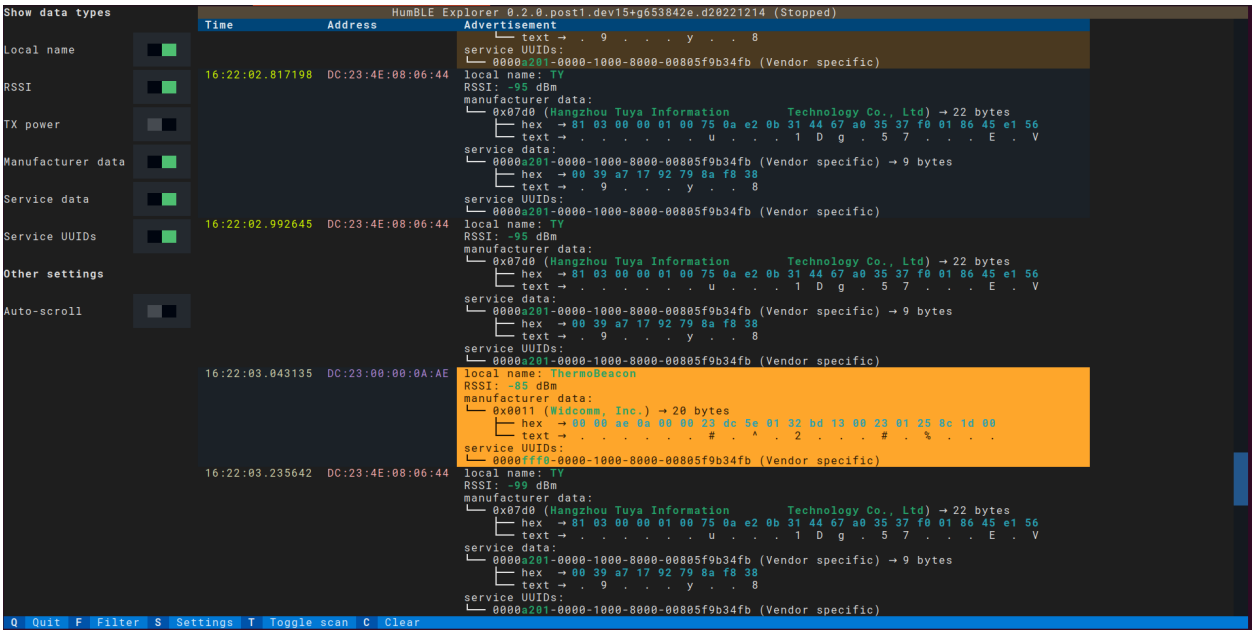
By default, HumBLE Explorer scans for BLE advertisements using your operating system's default Bluetooth adapter. You can change this with the `-a ADAPTER` option.

Also, by default HumBLE Explorer does *active scanning*. For every device the program finds, it requests extra information, with a `SCAN_REQ` packet directed at that device. The addressed device responds with a `SCAN_RSP` advertisement, which is also called *scan response data*. What data is returned for a `SCAN_RSP` packet depends on the type of device. It could be its device name, or manufacturer-specific data, or something else. If you want HumBLE Explorer to use *passive scanning*, use the `-s passive` option. The program then doesn't send `SCAN_REQ` packets, so devices don't respond with scan response data.

On macOS, users normally don't get access to the Bluetooth addresses of devices, but to a UUID. With the *-m* option, you get the actual Bluetooth address.

1.1.3 User interface

When you start HumBLE Explorer, it starts scanning for Bluetooth Low Energy advertisements and is showing them continuously in a scrollable table:



For each advertisement, the program shows columns for:

- **Time**: the time of receiving the advertisement
- **Address**: the device's Bluetooth address
- **Advertisement**: the decoded contents of the Bluetooth Low Energy advertisement

The first two columns are colored:

- Timestamps within the same second are shown in the same color.
- The same Bluetooth address is always shown in the same color.

The footer shows shortcut keys that are recognized by the program:

- Q: Quit the program
- F: Filter the devices that are shown
- S: Change settings
- T: Start or stop scan
- C: Clear all advertisements

1.1.4 Filtering devices

If you press the **F** key, an input widget appears where you can start typing a device filter. Currently one type of device filter is supported: an **address filter**. For instance, if you start typing `address=DC`, only advertisements from devices with their Bluetooth address beginning with `DC` are shown.

When you click outside the filter widget or press **Tab** to bring the focus to the next visible widget, the filter widget disappears, but the filter is still applied to limit the shown advertisements. Just press **F** again to change the filter, for instance by removing the filter with **Backspace** or changing the Bluetooth address part to filter on.

The number of filtered and received advertisements are always shown in the app's title.

1.1.5 Changing settings

If you press the **S** key, you can choose which advertising data types are shown in the table. By default all data types are shown, but you can enable or disable each of them individually by clicking on the checkbox or focusing it with **Tab** and then press **Enter** or **Space** to toggle it. You can also change some other settings, such as autoscrolling.

1.1.6 Starting and stopping the scan

If you press the **T** key, you stop the scan if it's running and you start the scan if it's stopped. The scanning status is always shown in the app's title.

When the scan is running and autoscrolling is enabled in the settings, the program continuously scrolls the table with advertisements so you are always seeing the most recent results. When the scan isn't running or autoscrolling is disabled, you can scroll through the history of received advertisements with the scroll wheel, by dragging the scroll bar, or by pressing **PgUp**, **PgDown** or the arrow keys up and down when the table widget is focused.

1.1.7 Clearing all advertisements

If you press the **C** key, the program clears all received advertisements. The table is filled again with newly received advertisements.

1.1.8 Quitting the program

You quit the program by pressing **Q**.

1.2 Contributing

Welcome to the HumBLE Explorer contributor's guide.

This document focuses on getting any potential contributor familiarized with the development processes, but other kinds of contributions are also appreciated.

If you are new to using git or have never collaborated in a project previously, please have a look at contribution-guide.org. Other resources are also listed in the excellent guide created by FreeCodeCamp[1].

Please notice, all users and contributors are expected to be **open, considerate, reasonable, and respectful**. When in doubt, Python Software Foundation's Code of Conduct is a good reference in terms of behavior guidelines.

[1] Even though, these resources focus on open source projects and communities, the general ideas behind collaborating with other developers to collectively create software are general and can be applied to all sorts of environments, including private companies and proprietary code bases.

1.2.1 Issue Reports

If you experience bugs or general issues with HumBLE Explorer, please have a look on the issue tracker. If you don't see anything useful there, please feel free to fire an issue report.

Tip: Please don't forget to include the closed issues in your search. Sometimes a solution was already reported, and the problem is considered **solved**.

New issue reports should include information about your programming environment (e.g., operating system, Python version) and steps to reproduce the problem. Please try also to simplify the reproduction steps to a very minimal example that still illustrates the problem you are facing. By removing other factors, you help us to identify the root cause of the issue.

1.2.2 Contributions of data

HumBLE Explorer tries to show a human-friendly description for as much data as possible, such as descriptions of the UUIDs in Bluetooth advertisements. Not all UUIDs are known, however. If you see the program showing a UUID with the description **Unknown** and you know what the description should be, please contribute it to the bluetooth-numbers project by opening an issue there. HumBLE Explorer uses the bluetooth-numbers library for its descriptions, and updating the bluetooth-numbers library with new descriptions will add them to HumBLE Explorer too.

1.2.3 Documentation Improvements

You can help improve HumBLE Explorer's docs by making them more readable and coherent, or by adding missing information and correcting mistakes.

HumBLE Explorer's documentation uses Sphinx as its main documentation compiler. This means that the docs are kept in the same repository as the project code, and that any documentation update is done in the same way was a code contribution.

The documentation is written in the reStructuredText markup language.

Tip: Please notice that the GitHub web interface provides a quick way of propose changes in HumBLE Explorer's files. While this mechanism can be tricky for normal code contributions, it works perfectly fine for contributing to the docs, and can be quite handy.

If you are interested in trying this method out, please navigate to the `docs` folder in the source repository, find which file you would like to propose changes and click in the little pencil icon at the top, to open GitHub's code editor. Once you finish editing the file, please write a message in the form at the bottom of the page describing which changes have you made and what are the motivations behind them and submit your proposal.

When working on documentation changes in your local machine, you can compile them using `tox`:

```
tox -e docs
```

and use Python's built-in web server for a preview in your web browser (`http://localhost:8000`):

```
python3 -m http.server --directory 'docs/_build/html'
```

1.2.4 Code Contributions

HumBLE Explorer is using Textual for its text user interface and Bleak for Bluetooth Low Energy support. It's recommended to make yourself comfortable with both libraries if you want to contribute to this project. Both projects have excellent documentation and example code.

If you want to learn more about Bluetooth Low Energy development, read the book Develop your own Bluetooth Low Energy Applications for Raspberry Pi, ESP32 and nRF52 with Python, Arduino and Zephyr and the accompanying GitHub repository koenvervloesem/bluetooth-low-energy-applications. The Python examples in the book are using Bleak, and it also gives a lot of practical explanations about how BLE works.

Submit an issue

Before you work on any non-trivial code contribution it's best to first create a report in the issue tracker to start a discussion on the subject. This often provides additional considerations and avoids unnecessary work.

Create an environment

Before you start coding, we recommend creating an isolated virtual environment to avoid any problems with your installed Python packages. This can easily be done via either `virtualenv`:

```
virtualenv <PATH TO VENV>
source <PATH TO VENV>/bin/activate
```

or Miniconda:

```
conda create -n humble_explorer python=3 six virtualenv pytest pytest-cov
conda activate humble_explorer
```

Clone the repository

1. Create an user account on GitHub if you do not already have one.
2. Fork the project repository: click on the *Fork* button near the top of the page. This creates a copy of the code under your account on GitHub.
3. Clone this copy to your local disk:

   ```
   git clone git@github.com:YourLogin/humble-explorer.git
   cd humble-explorer
   ```

4. You should run:

   ```
   pip install -U pip setuptools -e .
   ```

   to be able to import the package under development in the Python REPL.

5. Install `pre-commit`:

   ```
   pip install pre-commit
   pre-commit install
   ```

   HumBLE Explorer comes with a lot of hooks configured to automatically help the developer to check the code being written.

Implement your changes

1. Create a branch to hold your changes:

   ```
   git checkout -b my-feature
   ```

   and start making changes. Never work on the main branch!

2. Start your work on this branch. Don't forget to add docstrings to new functions, modules and classes, especially if they are part of public APIs.

3. Add yourself to the list of contributors in `AUTHORS.rst`.

4. When you're done editing, do:

   ```
   git add <MODIFIED FILES>
   git commit
   ```

   to record your changes in git.

   Please make sure to see the validation messages from `pre-commit` and fix any eventual issues. This should automatically use flake8/black to check/fix the code style in a way that is compatible with the project.

   Important: Don't forget to add unit tests and documentation in case your contribution adds an additional feature and is not just a bugfix.

   Moreover, writing a descriptive commit message is highly recommended. In case of doubt, you can check the commit history with:

   ```
   git log --graph --decorate --pretty=oneline --abbrev-commit --all
   ```

   to look for recurring communication patterns.

   ---

5. Please check that your changes don't break any unit tests with:

   ```
   tox
   ```

   (after having installed `tox` with `pip install tox` or `pipx`).

   You can also use `tox` to run several other pre-configured tasks in the repository. Try `tox -av` to see a list of the available checks.

Submit your contribution

1. If everything works fine, push your local branch to GitHub with:

   ```
   git push -u origin my-feature
   ```

2. Go to the web page of your fork and click "Create pull request" to send your changes for review.

   Find more detailed information in creating a PR. You might also want to open the PR as a draft first and mark it as ready for review after the feedbacks from the continuous integration (CI) system or any required fixes.

Troubleshooting

The following tips can be used when facing problems to build or test the package:

1. Make sure to fetch all the tags from the upstream repository. The command `git describe --abbrev=0 --tags` should return the version you are expecting. If you are trying to run CI scripts in a fork repository, make sure to push all the tags. You can also try to remove all the egg files or the complete egg folder, i.e., `.eggs`, as well as the `*.egg-info` folders in the `src` folder or potentially in the root of your project.

2. Sometimes `tox` misses out when new dependencies are added, especially to `setup.cfg` and `docs/requirements.txt`. If you find any problems with missing dependencies when running a command with `tox`, try to recreate the `tox` environment using the `-r` flag. For example, instead of:

   ```
   tox -e docs
   ```

   Try running:

   ```
   tox -r -e docs
   ```

3. Make sure to have a reliable `tox` installation that uses the correct Python version (e.g., 3.8+). When in doubt you can run:

   ```
   tox --version
   # OR
   which tox
   ```

   If you have trouble and are seeing weird errors upon running `tox`, you can also try to create a dedicated virtual environment with a `tox` binary freshly installed. For example:

   ```
   virtualenv .venv
   source .venv/bin/activate
   .venv/bin/pip install tox
   .venv/bin/tox -e all
   ```

4. Pytest can drop you in an interactive session in the case an error occurs. In order to do that you need to pass a `--pdb` option (for example by running `tox -- -k <NAME OF THE FALLING TEST> --pdb`). You can also setup breakpoints manually instead of using the `--pdb` option.

5. You can debug HumBLE Explorer with Textual's debug console.

   To use the console, open up **two** terminal emulators. Run the following in one of the terminals:

   ```
   textual console
   ```

   You should see the Textual devtools welcome message.

   In the other console, go to the `src` directory and run HumBLE Explorer with:

   ```
   TEXTUAL=devtools python3 humble_explorer.__main__
   ```

1.2.5 Maintainer tasks

Releases

If you are part of the group of maintainers and have correct user permissions on PyPI, the following steps can be used to release a new version for HumBLE Explorer:

1. Make sure all unit tests are successful.
2. Tag the current commit on the main branch with a release tag, e.g., `v1.2.3`.
3. Push the new tag to the upstream repository, e.g., `git push upstream v1.2.3`
4. Clean up the `dist` and `build` folders with `tox -e clean` (or `rm -rf dist build`) to avoid confusion with old builds and Sphinx docs.
5. Run `tox -e build` and check that the files in `dist` have the correct version (no `.dirty` or git hash) according to the git tag. Also check the sizes of the distributions, if they are too big (e.g., > 500KB), unwanted clutter may have been accidentally included.
6. Run `tox -e publish -- --repository pypi` and check that everything was uploaded to PyPI correctly.

1.3 License

1.4 Contributors

- Koen Vervloesem

1.5 Changelog

1.5.1 Version 0.5.0: Bluetooth addresses on macOS (2023-03-19)

On macOS, users normally don't get access to the Bluetooth addresses of devices, but to a UUID. This release adds the `-m` option to get the actual Bluetooth address.

New features

- Add option to access Bluetooth addresses on macOS by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/51

Fixes

- Fix for Textual 0.14 by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/50

1.5.2 Version 0.4.1: Switch to switches (2023-02-16)

This is a bugfix release for a breaking change in Textual 0.11.0.

- Add GitHub profile to authors page by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/40
- Update pyscaffold v4.4 by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/45
- Use furo theme for documentation by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/46
- Autoupdate pre-commit by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/47
- Change Checkbox to Switch for Textual 0.11.0 by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/48

1.5.3 Version 0.4.0: Minor user interface improvements (2022-12-28)

This release adds some minor user interface improvements. The description **Unknown** for unknown company IDs and UUIDs is now shown in red. If the Bluetooth address has a known OUI, the vendor name is shown. And the number of filtered and received advertisements is now shown in the title.

The documentation now also tells what to do if you want to contribute descriptions for unknown UUIDs. You can contribute these to the bluetooth-numbers project.

New features

- Color "Unknown" in red for company IDs and UUIDs by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/35
- Put the number of shown (filtered) and received advertisements in title by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/36
- Show vendor of Bluetooth address if it has a known OUI by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/37

Miscellaneous

- Adds and updates pre-commit hooks by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/34
- Add documentation about contributing data such as UUID descriptions by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/38

1.5.4 Version 0.3.3: Python 3.7 works! (2022-12-25)

This is a bugfix release. The most visible fix is for a bug that let HumBLE Explorer fail on Python 3.7.

Fixes

- Minor code style fixes by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/30
- Run tests with the correct Python version in CI by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/31
- Fix byte separator in RichHexData by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/32

1.5.5 Version 0.3.2: macOS works! (2022-12-23)

This is a bugfix release. The most important fix is that HumBLE Explorer now finally works on macOS too.

Fixes

- Fix typing for optional argument in SettingsWidget constructor by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/26
- Create BleakScanner object on mount by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/27
- Disable duplicate detection of advertisement data on Linux by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/28

1.5.6 Version 0.3.1 (2022-12-22)

This is a maintenance release with some fixes under the hood.

- Improve documentation by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/19
- Add module and method docstrings by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/20
- Typing fixes by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/21
- More typing fixes by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/22
- Use bluetooth-numbers package to translate UUIDs to names by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/23
- Upgrade to bluetooth-numbers 1.0 API by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/24

1.5.7 Version 0.3.0 (2022-12-14)

This version adds a lot of user interface improvements. You can filter advertisements on device addresses, you can choose which advertisement data types are shown, you can enable or disable autoscrolling and you can clear the list of received advertisements.

New features

- Add address filter by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/12
- Automatically hide and focus filter widget by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/14
- Add sidebar to select data to show by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/15
- UI improvements by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/18

Miscellaneous

- Add unit tests for renderables by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/11
- Set 5% threshold for codecov by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/13
- Add usage docs by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/16
- Make address filter a reactive attribute by @koenvervloesem in https://github.com/koenvervloesem/humble-explorer/pull/17

Full Changelog: https://github.com/koenvervloesem/humble-explorer/compare/v0.2.0...v0.3.0

1.5.8 Version 0.2.0 (2022-12-02)

- Timestamps within the same second are rendered with the same color.
- Each Bluetooth address is rendered with its own color for easier recognition of devices.

1.5.9 Version 0.1.1 (2022-12-01)

Fixes a ModuleNotFoundError.

1.5.10 Version 0.1.0 (2022-12-01)

Initial version of HumBLE Explorer.

1.6 humble_explorer

1.6.1 humble_explorer package

HumBLE Explorer: Human-Friendly Bluetooth Low Energy Explorer.

This is a cross-platform (Windows, Linux, macOS) human-friendly program to scan for Bluetooth Low Energy (BLE) advertisements on the command line.

Submodules

humble_explorer.app module

Module with the Textual app that scans for Bluetooth Low Energy advertisements.

class humble_explorer.app.**BLEScannerApp**(*cli_args: Namespace*)

Bases: App[None]

A Textual app to scan for Bluetooth Low Energy advertisements.

BINDINGS: ClassVar[list[BindingType]] = [('q', 'quit', 'Quit'), ('f', 'toggle_filter', 'Filter'), ('s', 'toggle_settings', 'Settings'), ('t', 'toggle_scan', 'Toggle scan'), ('c', 'clear_advertisements', 'Clear')]

CSS_PATH: ClassVar[CSSPathType | None] = 'app.css'

File paths to load CSS from.

action_clear_advertisements() → None

Clear the list of received advertisements.

action_toggle_filter() → None

Enable or disable filter input widget.

async action_toggle_scan() → None

Start or stop BLE scanning.

action_toggle_settings() → None

Enable or disable settings widget.

add_advertisement_to_table(*table: DataTable*, *now:* RichTime, *device_address:* RichDeviceAddress, *rich_advertisement:* RichAdvertisement) → None

Add new row to table with time, address and advertisement.

Parameters

- **table** (*textual.widgets.DataTable*) – The table to add an advertisement to.
- **now** (RichTime) – The time.
- **device_address** (RichDeviceAddress) – The device address.
- **rich_advertisement** (RichAdvertisement) – The advertisement.

compose() → Iterable[Widget]

Create child widgets for the app.

Returns

The child widgets for the app.

Return type
textual.app.ComposeResult

dark: Reactive[bool]

Use a dark theme if *True*, otherwise use a light theme.

Modify this attribute to switch between light and dark themes.

Example

```
`python self.app.dark = not self.app.dark # Toggle dark mode `
```

async on_advertisement(*device: BLEDevice*, *advertisement_data: AdvertisementData*) → None

Show advertisement data on detection of a BLE advertisement.

Parameters

- **device** (*BLEDevice*) – The device advertising the data.
- **advertisement_data** (*AdvertisementData*) – The advertised data.

on_input_changed(*message: Changed*) → None

Filter advertisements with user-supplied filter.

Parameters
message (*textual.widgets.Input.Changed*) – The message with the user's changed input.

async on_mount() → None

Initialize interface and start BLE scan.

on_switch_changed(*message: Changed*) → None

React when the switch is ticked or unticked.

Show or hide advertisement data depending on the state of the switches.

Parameters
message (*textual.widgets.Switch.Changed*) – The message with the changed switch.

recreate_table() → None

Recreate table with advertisements.

scroll_if_autoscroll() → None

Scroll to the end if autoscroll is enabled.

set_title() → None

Set the title of the app with a description of the scanning status.

show_data_config() → dict[str, bool]

Return dictionary with which advertisement data to show.

Returns

Each key has the value True if this advertisement
type should be shown and False if not.

Return type
dict[str, bool]

async start_scan() → None

Start BLE scan.

async stop_scan() → None

Stop BLE scan.

sub_title: Reactive[str]

The sub-title for the application.

The initial value for *sub_title* will be set to the *SUB_TITLE* class variable if it exists, or an empty string if it doesn't.

Sub-titles are typically used to show the high-level state of the app, such as the current mode, or path to the file being worker on.

Assign a new value to this attribute to change the sub-title. The new value is always converted to string.

title: Reactive[str]

The title for the application.

The initial value for *title* will be set to the *TITLE* class variable if it exists, or the name of the app if it doesn't.

Assign a new value to this attribute to change the title. The new value is always converted to string.

watch_address_filter(*old_filter: str*, *new_filter: str*) → None

React when the reactive attribute address_filter changes.

This recreates the table.

Parameters

- **old_filter** (*str*) – The old value of the filter.
- **new_filter** (*str*) – The new value of the filter.

humble_explorer.renderables module

Module with Rich renderables for HumBLE Explorer's user interface.

class humble_explorer.renderables.**RichAdvertisement**(*data: AdvertisementData*, *show_data: dict[str, bool]*)

Bases: object

Rich renderable that shows advertisement data.

height() → int

Return the number of lines this Rich renderable uses.

class humble_explorer.renderables.**RichCompanyID**(*cic: int*)

Bases: object

Rich renderable that shows company ID and name.

class humble_explorer.renderables.**RichDeviceAddress**(*address: str*)

Bases: object

Rich renderable that shows a Bluetooth device address aand OUI description.

Every address is rendered in its own color.

height() → int

Return the number of lines this Rich renderable uses.

class humble_explorer.renderables.**RichHexData**(*data: bytes*)

Bases: object

Rich renderable that shows hex data.

class humble_explorer.renderables.**RichHexString**(*data: bytes*)

Bases: object

Rich renderable that shows hex data as a string.

Non-printable characters are replaced by a dot.

class humble_explorer.renderables.**RichRSSI**(*rssi: int*)

Bases: object

Rich renderable that shows RSSI of a device.

class humble_explorer.renderables.**RichTime**(*time: datetime*)

Bases: object

Rich renderable that shows a time.

All times within the same second are rendered in the same color.

class humble_explorer.renderables.**RichUUID**(*uuid128: str*)

Bases: object

Rich renderable that shows a UUID with description and colors.

humble_explorer.utils module

This module contains utility functions for HumBLE Explorer.

humble_explorer.utils.**hash8**(*message: str*) → int

Compute an 8-bit hash from the message with Pearson hashing.

Parameters
message (*str*) – The message to hash.

Returns
An 8-bit hash value for *message*.

Return type
int

humble_explorer.widgets module

This module contains Textual widgets for HumBLE Explorer's user interface.

class humble_explorer.widgets.**FilterWidget**(*placeholder: str = ''*)

Bases: Input

A Textual widget to filter Bluetooth Low Energy advertisements.

can_focus: bool = True

Widget may receive focus.

can_focus_children: bool = True

Widget's children may receive focus.

max_size: reactive[int | None]

on_blur() → None

Automatically hide widget on losing focus.

suggester: Suggester | None

The suggester used to provide completions as the user types.

class humble_explorer.widgets.**SettingsWidget**(*id: str | None*)

Bases: Static

A Textual widget to let the user choose settings.

can_focus: bool = False

Widget may receive focus.

can_focus_children: bool = True

Widget's children may receive focus.

compose() → ComposeResult

Show switches.

on_blur() → None

Automatically hide widget on losing focus.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

h

INDEX

R

S

T

W