

---

# CANalyzat0r Documentation

*Release 1.0*

**pschmied (SCHUTZWERK GmbH)**

**Jan 29, 2020**



## CONTENTS:

<b>1</b>	<b>Requirements</b>	<b>1</b>
1.1	Docker . . . . .	1
<b>2</b>	<b>CANAlYzat0r manual</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Usage: Tab by tab . . . . .	4
2.2.1	Main Tab . . . . .	4
2.2.1.1	Where's my interface?!?!1! . . . . .	5
2.2.1.2	Creating and selecting projects . . . . .	5
2.2.1.3	Log levels . . . . .	5
2.2.1.4	Where's my data being saved to?!!? . . . . .	5
2.2.1.5	But what if i want to export my data? . . . . .	5
2.2.2	General . . . . .	5
2.2.2.1	Why can I change interface settings in every tab and why is there a global interface? . . . . .	5
2.2.2.2	Well, how can I manage packets in between tabs? . . . . .	5
2.2.2.3	Ok Ok, but how can I import my SocketCAN dumps? . . . . .	6
2.2.2.4	What are known packets? . . . . .	6
2.2.2.5	I've discovered a bug, pls fix! . . . . .	6
2.2.3	Sniffer Tab . . . . .	6
2.2.3.1	How to sniff? . . . . .	6
2.2.3.2	Ignoring packets . . . . .	6
2.2.3.3	I get errors/no data when I try to sniff!!! . . . . .	6
2.2.3.4	The sniffer tab doesn't list the packets I'm sending . . . . .	6
2.2.4	Sender Tab . . . . .	7
2.2.4.1	What do the sub tabs do? . . . . .	7
2.2.4.2	Why does Loop do?!?! . . . . .	7
2.2.5	Fuzzer Tab . . . . .	7
2.2.5.1	What does this thing to? . . . . .	7
2.2.5.2	What are masks? . . . . .	7
2.2.5.3	Other fuzzers are much faster!!! . . . . .	7
2.2.5.4	What are the modes? . . . . .	7
2.2.6	Comparer Tab . . . . .	7
2.2.6.1	What can I compare? . . . . .	7
2.2.7	Searcher Tab . . . . .	8
2.2.7.1	What can I search for? . . . . .	8
2.2.7.2	It doesn't work!!! . . . . .	8
2.2.7.3	It still doesn't work :( . . . . .	8
2.2.7.4	I want to do it manually, how can this tool help me? . . . . .	8
2.2.8	Manager Tab . . . . .	8
2.2.8.1	What can I do using the dumps tab? . . . . .	8

2.2.8.2	I know packet XY has effect ZZ, do I create a dump or a known packet? . . . . .	8
2.2.8.3	Importing/Exporting projects . . . . .	8
2.2.9	Filter Tab . . . . .	9
2.2.9.1	What can I filter? . . . . .	9
2.2.9.2	How can I try this without a car or CAN device? . . . . .	9
<b>3</b>	<b>Contributing</b>	<b>11</b>
3.1	Guidelines . . . . .	11
3.2	I want to add a new tab, what do I have to do? . . . . .	11
<b>4</b>	<b>src package</b>	<b>13</b>
4.1	Subpackages . . . . .	13
4.2	Submodules . . . . .	13
4.3	CANalyzat0r.AboutTab module . . . . .	13
4.4	CANalyzat0r.CANData module . . . . .	13
4.5	CANalyzat0r.CANalyzat0r module . . . . .	13
4.6	CANalyzat0r.ComparerTab module . . . . .	13
4.7	CANalyzat0r.Database module . . . . .	14
4.8	CANalyzat0r.FilterTab module . . . . .	20
4.9	CANalyzat0r.FuzzerTab module . . . . .	23
4.10	CANalyzat0r.Globals module . . . . .	24
4.11	CANalyzat0r.ItemAdderThread module . . . . .	25
4.12	CANalyzat0r.KnownPacket module . . . . .	25
4.13	CANalyzat0r.Logger module . . . . .	26
4.14	CANalyzat0r.MainTab module . . . . .	27
4.15	CANalyzat0r.ManagerTab module . . . . .	29
4.16	CANalyzat0r.Packet module . . . . .	31
4.17	CANalyzat0r.PacketsDialog module . . . . .	31
4.18	CANalyzat0r.PacketSet module . . . . .	32
4.19	CANalyzat0r.PacketTableModel module . . . . .	33
4.20	CANalyzat0r.Project module . . . . .	36
4.21	CANalyzat0r.SearcherTab module . . . . .	36
4.22	CANalyzat0r.SenderTab module . . . . .	38
4.23	CANalyzat0r.SenderTabElement module . . . . .	39
4.24	CANalyzat0r.SenderThread module . . . . .	40
4.25	CANalyzat0r.Settings module . . . . .	41
4.26	CANalyzat0r.SnifferProcess module . . . . .	42
4.27	CANalyzat0r.SnifferTab module . . . . .	42
4.28	CANalyzat0r.SnifferTabElement module . . . . .	43
4.29	CANalyzat0r.Strings module . . . . .	44
4.30	CANalyzat0r.Toolbox module . . . . .	44
4.31	CANalyzat0r.mainWindow module . . . . .	47
4.32	CANalyzat0r.AbstractTab module . . . . .	47
<b>5</b>	<b>Used libraries and files</b>	<b>51</b>
5.1	Libs . . . . .	51
5.2	Misc . . . . .	51
<b>6</b>	<b>Indices and tables</b>	<b>53</b>
	<b>Python Module Index</b>	<b>55</b>
	<b>Index</b>	<b>57</b>

## REQUIREMENTS

To install all requirements, please execute the following commands:

- `sudo apt-get install python3.5 python3-pip python3-pyside can-utils ffmpeg`
- `sudo pip3 install sphinx_rtd_theme`
- `sudo pipenv install -e 'git+https://github.com/hardbyte/python-can.git@3.3.2#egg=can'`

Note: It's recommended to just execute `install.requirements.sh`.

### 1.1 Docker

There's also a docker container available. Check the docker folder.



## CANALYZAT0R MANUAL

### 2.1 Introduction

You can use **CANAllyzat0r** to quickly analyze the CAN bus in many ways. It's great.



This documentation will guide you through the usage of the application. Also, you can find the code documentation in this document if you want to extend and/or contribute to this project.

## 2.2 Usage: Tab by tab

### 2.2.1 Main Tab

Welcome to CANalyzat0rs main tab! Here you can change interface settings and creat/remove virtual CAN interfaces. Don't worry, the kernel modules should aready be loaded for you.



### 2.2.1.1 Where's my interface?!?!1!

If you can't find your attached CAN interface in the ComboBox, please check the output of `ifconfig -a`. In order to use your interface with CANalyzat0r, a SocketCAN device must be present. Maybe you have to load another kernel module/driver?

### 2.2.1.2 Creating and selecting projects

On a fresh startup, you should encounter a message saying that a new project should be created. You can still use this application without a selected project. However, one can't save dumps or known packets. To create a project, please refer to the manager tab. After you have created a project there, you can set it as active project in the main tab.

### 2.2.1.3 Log levels

You can set the minimum log level for which messages will be printed to the log box in this tab.

### 2.2.1.4 Where's my data being saved to?!?!?

By default, CANalyzat0r creates a SQLite database called "database.db" in the data folder. Please take care of this file as everything you discover is saved here.

### 2.2.1.5 But what if i want to export my data?

Please check the manager tab and learn on how to export projects and dumps.

## 2.2.2 General

### 2.2.2.1 Why can I change interface settings in every tab and why is there a global interface?

You can set a global interface in order to set the selected interface for every inactive tab. On top of that, you can override this setting for every tab individually using the button. This allows you to e.g. fuzz on `can0` and sniff on `vcan1` at the same time.

### 2.2.2.2 Well, how can I manage packets in between tabs?

You can:

- Select rows and copy them to another tab (if allowed)
- Delete rows by selecting rows and pressing `Del` on your keyboard

### 2.2.2.3 Ok Ok, but how can I import my SocketCAN dumps?

Just copy and paste them into the GUI tables <:

### 2.2.2.4 What are known packets?

Once you discovered that packet XY does Action ZZ on your car or setup, you can add this knowledge to the database using the manager tab. This adds the discovered information globally for a specific project. Using this, the “Description” column in the GUI tables is filled with data, so you can recognize a re-occurring packet.

### 2.2.2.5 I’ve discovered a bug, pls fix!

Please report bugs using GitHub issues, Thanks.

## 2.2.3 Sniffer Tab

### 2.2.3.1 How to sniff?

It’s simple. For every discovered interface you can find a sniffer tab here. If no tab for your desired interface is displayed, please re-check all available interfaces using the button on the main tab. Once you find your desired interface, just click on start.

### 2.2.3.2 Ignoring packets

You can add tab specific packets to the ignore list. Hint: Use a blank data field if you want to exclude whole IDs.

Another Hint: You can use the filtering buttons to remove background noise. It’s great.

### 2.2.3.3 I get errors/no data when I try to sniff!!1!

- Maybe your SocketCAN interface disappeared?
- Maybe you have selected a wrong bitrate?
- Have you tried turning it off and on again?

### 2.2.3.4 The sniffer tab doesn’t list the packets I’m sending

That’s normal. You will only see packets that you are receiving on a specific interface. This prevents the packets that you generate via fuzzing from being in your sniffed dump. If you really need all packets in one dump, you can use `candump`.

## 2.2.4 Sender Tab

### 2.2.4.1 What do the sub tabs do?

You can create multiple sender tabs to handle sending various packet sets.

### 2.2.4.2 Why does Loop do?!?!

One can either send packets once or in a loop with a specific packet gap. Imagine you have to send a packet every X seconds to keep a CAN device online: It's handy.

## 2.2.5 Fuzzer Tab

### 2.2.5.1 What does this thing to?

Using this tab you can send random packets into the CAN bus to discover things. You can tune settings that control random packet generation using the GUI elements.

### 2.2.5.2 What are masks?

You can write static values into the masks or put an X if that character should be randomized. Using this, you can freely control the payload of generated packets. Hint: You can change masks and lengths **while fuzzing**.

### 2.2.5.3 Other fuzzers are much faster!!1!

This is a python based fuzzer which also displays the packets on the GUI. This convenience costs performance. If you want the best performance you can use `cangen` of the `can-utils` package and import the created packets later.

### 2.2.5.4 What are the modes?

- User specified: You can freely specify ID and data masks
- 11 bit IDs / 29 bit IDs: Only short/extended IDs will be used

## 2.2.6 Comparer Tab

### 2.2.6.1 What can I compare?

You can compare two sets of packets. You will get all packets they have in common.

## 2.2.7 Searcher Tab

### 2.2.7.1 What can I search for?

Using this tab you can perform a binary packet search for a specific packet or a whole packet set that cause an effect. Let's suppose you've fuzzed and got a large packet dump that, when replayed, causes an effect on your CAN device / car. You now want to extract the relevant packet(s) out of that dump. Searcher tab to the rescue – Load the whole packet dump and let the analyzer routine guide you. Note: This first tries to search for **1** packet that causes an action. If this fails, the searcher tries to continuously minimize the packet set.

### 2.2.7.2 It doesn't work!!!

Don't give up too fast, try the following things: - Set the packet gap to a lower value, you can even try 0 - Just try again and hope for better shuffling - Use another dump/fuzz again, ... - Wait a few seconds after each chunk

### 2.2.7.3 It still doesn't work :(

CAN devices can be extremely tricky, for example speedometers. Depending on your dump, you may have to try it multiple times with the same dump because of packet timings and/or bad luck. If you replay your whole dump and see the desired action, you will be able to find it using the searcher tab.

### 2.2.7.4 I want to do it manually, how can this tool help me?

Create a new sender, add the dump to it and send them in a loop. Minimize the packet set from the bottom using your "CTRL+C" and "DEL" and try again. If it didn't perform the desired action, paste the packets again and delete other packets.

## 2.2.8 Manager Tab

### 2.2.8.1 What can I do using the dumps tab?

You can save a set of packets that you want to keep (e.g. for further analysis) and save it to the database. This allows you to load the dump again at a later point. Hint: You can edit the values in the GUI table and update the values in the database using the update button.

### 2.2.8.2 I know packet XY has effect ZZ, do I create a dump or a known packet?

Just create a dump with one packet entry and the application will handle the rest for you.

### 2.2.8.3 Importing/Exporting projects

If you want to import/export projects, use the manager tab. It exports all saved data of a project to a editable textfile in JSON format. Go ahead and edit values if you want, but be careful and don't mess with the data integrity <:

## 2.2.9 Filter Tab

### 2.2.9.1 What can I filter?

Let's suppose you want to find (a) specific packet(s) that get generated when you (for example) press a button, accelerate or lock the cars doors. The captured CAN traffic contains so much data that you can't seem to find the packets easily. Let's use the filter tab:

- You can collect background noise containing CAN packets that are sent on the bus without any user interaction
- After that, a variable amount of samples get captured. You have to perform the desired action **in every sample** - e.g. lock the doors in every sample.
- As soon as all data has been captured, the filter tab begins to analyze it. It filters background noise out of each sample and tries to find packets that occur in every sample. These are most likely the packets your are looking for.

### 2.2.9.2 How can I try this without a car or CAN device?

Use ICSim!



## CONTRIBUTING

Here's some useful info if you want to contribute.

### 3.1 Guidelines

- Each tab has its own Class. If possible, inherit from `AbstractTab`.
  - To provide comatibility:
  - The displayed data should also be in a raw data list called `rawData` which is *always* up to date
  - `prepareUI` initializes all GUI elements
  - `active` manages the status of a tab
  - Tab specific `CANData` instances are called `CANData`
- Please log useful information using an own logger instance
- Use existing Toolbox methods if possible
- Use batch database operations using raw lists (not objects) for better performance
- Use docstrings
- Keep the `.ui` files clean: Always name new GUI elements properly according to existing ones
- Put new strings in the Strings file and reference it

### 3.2 I want to add a new tab, what do I have to do?

- Create a new tab on the GUI and stick to the already existing naming conventions
- Add a `QTableView` to display your data and other GUI elements
- Update `mainWindow.py` using `pyside-uic mainWindow.ui > mainWindow.py`.
- Add a new File and a new class which inherits from `AbstractTab`
- Call the parents constructor in your `__init__`
- Add the GUI elements from the `.ui` file to your code. You can refer to the other tabs to see how it's done. Also, add the click handlers here.
- Call `prepareUI` as last action in `__init__`
- If your tab needs an interface or displays interface values: Add your tab class or instance to `updateInterfaceLabels()` and/or `updateCANDataInstances()`.

- If your tab uses an instance: Add an instance to *Globals.py* and create one at startup (see *CANalyzat0r.py*).
- If your tab uses a static class: Call *prepareUI* at startup (see *CANalyzat0r.py*).



## SRC PACKAGE

### 4.1 Subpackages

### 4.2 Submodules

### 4.3 CANalyzat0r.AboutTab module

Created on Jun 26, 2017

@author: pschmied

**class** AboutTab.**AboutTab**

Bases: `object`

This class handles the logic of the about tab.

**static** **browseGitHub** (*event*)

Opens the SCHUTZWERK website.

**Parameters** **event** – Dummy, not used.

**static** **browseSW** (*event*)

Opens the SCHUTZWERK website.

**Parameters** **event** – Dummy, not used.

**static** **prepareUI** ()

This just sets up the GUI elements.

### 4.4 CANalyzat0r.CANData module

### 4.5 CANalyzat0r.CANalyzat0r module

### 4.6 CANalyzat0r.ComparerTab module

Created on Jun 30, 2017

@author: pschmied

```
class ComparerTab.ComparerTab (tabWidget)
```

Bases: *AbstractTab.AbstractTab*

This handles the logic of the comparer tab.

```
__init__ (tabWidget)
```

This just sets data and adds click handlers.

```
compare ()
```

Compares rawPacketSet1 and rawPacketSet2 and display the packet they have in common on the GUI.

```
getPacketSet (rawPacketList)
```

Opens a PacketsDialog to load selected packets into the passed raw packet list.

**Parameters** **rawPacketList** – The raw packet list

```
setPacketSet1 ()
```

Opens a PacketsDialog to load packet set 1 into rawPacketSet1.

```
setPacketSet2 ()
```

Opens a PacketsDialog to load packet set 2 into rawPacketSet2.

## 4.7 CANalyzat0r.Database module

The database model is as follows:

Note: The **bold** columns are NOT NULL

Created on May 17, 2017

@author: pschmied

```
class Database.Database
```

Bases: *object*

This class handles the database connection and is responsible for creating, deleting and updating values

```
__init__ ()
```

**This method does the following things:**

1. Setup logging
2. Create a DB connection and check the integrity
3. Create tables if necessary

```
checkDB ()
```

Checks if all the table count of the SQLite database matches the needed table count. If the check does pass the user will be notified to create a project if no project is existing yet. If the check does not pass the user will be prompted for an action: - Truncate the database and create an empty one - Keep the database and exit

**Returns** A boolean value indicating the database integrity status (True = good)

```
connect ()
```

Connect to the hard coded SQLite database path (see Settings).

Note: If a DatabaseError is encountered, the application will close with error code 1

**Returns** A SQLite3 connection object

**createTables ()**

Creates all needed tables. Check DatabaseStatements for the SQL statements.

**deleteFromTableByID (tableName, id)**

Delete a row from a table with a specific ID.

**Parameters**

- **tableName** – The table to delete from
- **id** – ID of the record to delete

**deleteFromTableByValue (tableName, column, value)**

Delete rows from a table with a specific value.

**Parameters**

- **tableName** – The table to delete from
- **column** – The column to check
- **value** – The value to search for

**deleteKnownPacket (knownPacket)**

Delete a KnownPacket object

**Parameters** **knownPacket** – The KnownPacket object to delete

**Returns****deletePacketSet (packetSet)**

Delete a PacketSet along with the associated packets

**Parameters** **packetSet** – The PacketSet object to delete

**Returns****deleteProjectAndData (project)**

Delete a project and all associated data.

**Parameters** **project** – The Project object to delete

**getKnownPackets (project=None)**

Get all known packets of a Project as objects. Uses the global project if no project is given.

**Parameters** **project** – Optional parameter to specify the project to use

**Returns** A list of all known packets as KnownPacket objects

**getOverallTableCount (tableName)**

Returns the count(\*) of a table.

**Parameters** **tableName** – The table to count the rows of

**Returns** The number of rows as integer

**getPacketSets (project=None)**

Get all packet sets of a Project as objects. Uses the global project if no project is given.

**Parameters** **project** – Optional parameter to specify the project to use

**Returns** A list of all known packets as PacketSet objects

**getPacketsOfPacketSet (packetSet, raw=False)**

Get all packets of a specific packet set. Note: Use raw=True for better performance.

**Parameters**

- **packetSet** – All returned packets will belong to this packet set
- **raw** – Boolean value to indicate if the packets will be returned as raw data list (True) or as list of objects (False)

**Returns** Depending on the value of raw: - True: List of value lists (raw data) - False: List of Packet objects

**getProjects** ()

Get all available projects as Project objects.

**Returns** A list of all projects as Project objects

**saveKnownPacket** (*knownPacket*)

Save a known packet to the database.

**Parameters** **knownPacket** – The KnownPacket object to save

**Returns** The database ID of the saved known packet

**savePacket** (*packet=None, packetSetID=None, CANID=None, data=None, timestamp="", iface="", commit=True*)

Save a packet to the database: Either by object Or by list-values → Faster for many values If the value in packet is not None: The passed object will be used Else: The separate values will be used

**Parameters**

- **packet** – Optional parameter: Packet object to save
- **packetSetID** – PacketSet ID of the saved packet
- **CANID** – CAN ID
- **data** – Payload data
- **timestamp** – Timestamp
- **iface** – The interface the packet was captured from
- **commit** – If the operation will be commit to the database or not. Batch operations use commit=False

**Returns** The database ID of the saved packet if commit is True. Else -1

**savePacketSet** (*packetSet*)

Save a PacketSet to the database. If there's a name collision, the user will be prompted for a new and unique name.

**Parameters** **packetSet** – The PacketSet to save

**savePacketSetWithData** (*packetSetName, rawPackets=None, project=None, packets=None*)

Save a packet set in the database with the given data. If no project is given, the global project will be used. You must specify `rawPackets` or `packets`.

**Parameters**

- **packetSetName** – The desired name of the PacketSet
- **rawPackets** – Optional: Raw Packet data (List of lists). You can also use `packets`.
- **project** – Optional parameter to specify a project. If this is not specified, the selected project will be used
- **packets** – Optional; List of packet objects to save to the packet set. If this is specified, `rawPackets` will be ignored

**Returns**



**static getDeleteByIDStatement** (*tableName, id*)

Returns an SQL delete statement using an ID where clause using *getDeleteByValueStatement()*

**Parameters**

- **tableName** – The table name to delete from
- **id** – Desired ID value for the where clause

**Returns**

The resulting SQL statement

e.g.: DELETE FROM TABLE1 WHERE ID = 1337

**static getDeleteByValueStatement** (*tableName, column, value*)

Returns an SQL delete statement using a where clause using *getDeleteByValueStatement()*

**Parameters**

- **tableName** – The table name to delete from
- **column** – Desired column for the where clause
- **value** – Desired column value for the where clause

**Returns**

The resulting SQL statement

e.g.: DELETE FROM TABLE1 WHERE NAME = 'BANANA'

**static getInsertKnownPacketStatement** (*projectID, CANID, data, description*)

Returns an SQL insert statement for a KnownPacket using *getInsertStatement()*.

**Parameters**

- **projectID** – The project this KnownPacket belongs to
- **CANID** – CAN ID
- **data** – Payload data
- **description** – What this specific Packet does

**Returns** The SQL insert statement with all KnownPacket specific values set

**static getInsertPacketSetStatement** (*projectID, name, date*)

Returns an SQL insert statement for a PacketSet using *getInsertStatement()*.

**Parameters**

- **projectID** – The project ID the record belongs to
- **name** – The name of the PacketSet
- **date** – Date

**Returns** The SQL insert statement with all PacketSet specific values set

**static getInsertPacketStatement** (*packetSetID, CANID, data, timestamp, iface*)

Returns an SQL insert statement for a Packet using *getInsertStatement()*.

**Parameters**

- **packetSetID** – The PacketSet this Packet belongs to
- **CANID** – CAN ID
- **data** – Payload data

- **timestamp** – Timestamp of the packet
- **iface** – Interface name from which this packet was captured from

**Returns** The SQL insert statement with all Packet specific values set

**static getInsertProjectStatement** (*name, description, date*)

Returns an SQL insert statement for a project using *getInsertStatement()*.

**Parameters**

- **name** – Projectname
- **description** – Project description
- **date** – Project date

**Returns** The SQL insert statement with all project specific values set

**static getInsertStatement** (*tableName, columnList, valuesList*)

Builds a SQL insert statement using the passed parameters

**Parameters**

- **tableName** – The table name to insert into
- **columnList** – List of column names that will be affected
- **valuesList** – List of values to put into the columns

**Returns**

An SQL insert statement with the desired values mapped to the columns

e.g. INSERT INTO TABLE1 (col1, col2) VALUES (1, 2)

**static getOverallCountStatement** (*tableName*)

Returns an SQL select count statement for the desired table using all rows

**Parameters** **tableName** – The table name to get the rowcount from

**Returns**

The resulting SQL statement

e.g.: SELECT COUNT (\*) FROM TABLE1

**static getSelectAllStatement** (*tableName*)

Returns an SQL select statement to get all data from a table.

**Parameters** **tableName** – The table name from which data will be selected

**Returns**

The resulting SQL select statement

e.g.: SELECT \* FROM TABLE1

**static getSelectAllStatementWhereEquals** (*tableName, column, value*)

Returns an SQL select statement to gather all data from a table using a where clause

**Parameters**

- **tableName** – The table name from which data will be selected
- **column** – The column which the where clause affects
- **value** – The desired value of the column

**Returns**

The resulting SQL statement with where clause

e.g.: `SELECT * FROM TABLE1 WHERE ID = 1337`

**static** `getUpdateByIDStatement` (*tableName*, *colValuePairs*, *ID*)

Builds a SQL update statement using the passed parameters **and an ID**

**Parameters**

- **tableName** – The table name to update
- **colValuePairs** – List of tuples: (column, desired value)
- **ID** – The ID of the record to update

**Returns**

An SQL update statement with the desired values mapped to the columns using the ID

e.g. `UPDATE TABLE1 SET col1 = 1, col2 = 2 WHERE ID = 1337`

```
knownPacketTableCANIDColName = 'CANID'
knownPacketTableDataColName = 'Data'
knownPacketTableDescriptionColName = 'Description'
knownPacketTableName = 'KnownPacket'
packetSetTableName = 'PacketSet'
packetTableCANIDColName = 'CANID'
packetTableDataColName = 'Data'
packetTableName = 'Packet'
projectTableDescriptionColName = 'Description'
projectTableName = 'Project'
projectTableNameColName = 'Name'
tableCount = 4
```

The Amount of tables that must be present

## 4.8 CANalyzat0r.FilterTab module

Created on Jun 02, 2017

@author: pschmied

**class** `FilterTab.DataAdderThread` (*snifferReceivePipe*, *sharedEnabledFlag*, *curSampleIndex*)

Bases: `PySide.QtCore.QThread`

This thread receives data from the sniffer process and emits a signal which causes the main thread to add the packets.

**\_\_init\_\_** (*snifferReceivePipe*, *sharedEnabledFlag*, *curSampleIndex*)

Initialize self. See help(type(self)) for accurate signature.

**frameToList** (*frame*)

Converts a received can.Message frame to raw list data. After that, the data is emitted using `signalSniffedPacket`



**Parameters** **frame** – can.Message CAN frame

**run()**

As long as `sharedEnabledFlag` is not set to 0 data will be received using the pipe and processed using `frameToList()`.

**signalSniffedPacket** = <PySide.QtCore.Signal object>

Emit a signal to the main thread when items are ready to be added This emits the packet and the current sample index

**staticMetaObject** = <PySide.QtCore.QMetaObject object>

**class** `FilterTab.FilterTab(tabWidget)`

Bases: `AbstractTab.AbstractTab`

This class handles the logic of the filter tab

**\_\_init\_\_**(*tabWidget*)

Initialize self. See help(type(self)) for accurate signature.

**addSniffedNoise**(*dummyIndex, packet*)

Adds the passed packet data to `noiseData`. This method gets called by a `DataAdderThread`.

**Parameters**

- **dummyIndex** – Not used, only exists to match the signature defined in the signal of the `DataAdderThread`
- **packet** – The packet object to extract and add data from

**addSniffedPacketToSample**(*curSampleIndex, packet*)

Adds a sniffed packet to the sample defined by `curSampleIndex`. Gets called by a `DataAdderThread`.

**Parameters**

- **curSampleIndex** – The sample index to get a list from `rawData[curSampleIndex]`
- **packet** – Packet data to add

**Returns**

**analyze**(*removeNoiseWithIDAndData=True*)

**Analyze captured data:**

1. Remove sorted noise data (if collected): For each sample: - Sort the sample to increase filtering speed - Remove noise
2. Find relevant packets: - Sort each sample to increase filtering speed - Assume that all packets of the first sample occurred in every other sample - Take every other sample and compare

Depending on `removeNoiseWithIDAndData` noise will be filtered by ID and data (default) or ID only

**Parameters** **removeNoiseWithIDAndData** – Optional value: Filter data by ID and Data or ID only

**clear**(*returnOldPackets=False*)

Clear the currently displayed data on the GUI and in the lists.

**collectNoise**(*seconds*)

Collect noise data and update `noiseData`. Uses the processes/threads started in `startSnifferAndAdder()`.

**Parameters** **seconds** – Amount of seconds to capture noise

**Returns** True if noise was captured. False if the user pressed “Cancel”

**getSampleData** (*sampleAmount*, *curSampleIndex*)

Collect sample data and add the sniffed data to `rawData[curSampleIndex]`. Uses the processes/threads started using `startSnifferAndAdder()`.

**Parameters**

- **sampleAmount** – Amount of samples to collect
- **curSampleIndex** – Index of the currently captured sample in `rawData`

**noiseData** = `None`

Noise that will be subtracted from the collected data

**outputRemainingPackets** (*remainingPackets*)

Output all remaining packets after filtering to the table view. Note: This also clears previous data

**Parameters** **remainingPackets** – Raw packet list to display

**sharedDataAdderEnabledFlag** = `None`

Shared process independent flag to terminate the data adder

**sharedSnifferEnabledFlag** = `None`

Shared process independent flag to terminate the sniffer process

**startFilter** ()

**Handle the filtering process:**

1. Collect noise
2. Record sample data
3. Analyze captured data

**startSnifferAndAdder** (*adderMethod*, *curSampleIndex=-1*)

Start a `DataAdderThread` and a `SnifferProcess` to collect data. They will communicate using a multiprocessing pipe to collect data without interrupting the GUI thread.

**Parameters**

- **adderMethod** – The `DataAdderThread` will call this method to handle the received data
- **curSampleIndex** – The index of the currently captured sample (-1 as default)

**stopSnifferAndAdder** ()

Stop the `DataAdderThread` and `SnifferProcess` using the shared integer variable.

**toggleGUIElements** (*state*)

{En, Dis}able all GUI elements that are used to change filter settings

**Parameters** **state** – Boolean value to indicate whether to enable or disable elements

**updateNoiseCollectProgress** (*progressDialog*, *value*)

Update the text and progressbar value displayed while collecting noise.

**Parameters**

- **progressDialog** – The `QProgressDialog` to update
- **value** – The value to set the progressbar to

## 4.9 CANalyzat0r.FuzzerTab module

Created on May 31, 2017

@author: pschmied

**class** FuzzerTab.**FuzzerTab** (*tabWidget*)

Bases: *AbstractTab.AbstractTab*

This class handles the logic of the fuzzer tab

**IDMask = None**

The ID is 8 chars max. - initialize it with only X chars

**IDMaskChanged** ()

This allows changing the ID mask values on the fly because a new value will only be set if the new value is valid.

**IDMaxValue = None**

Default: allow the max value of extended frames

**\_\_init\_\_** (*tabWidget*)

Initialize self. See help(type(self)) for accurate signature.

**addPacket** (*valueList, addAtFront=True, append=True, emit=True, addToRawDataOnly=False*)

Override the parents class method to add packets at front and to update the counter label

**clear** (*returnOldPackets=False*)

Clear the currently displayed data on the GUI and in the rawData list

**Parameters** **returnOldPackets** – If this is true, then the previously displayed packets will be returned as raw data list

**Returns** Previously displayed packets as raw data list (if returnOldPackets is True), else an empty list

**dataMask = None**

The data is 16 chars max.

**dataMaskChanged** ()

This allows changing the data mask values on the fly because a new value will only be set if the new value is valid.

**dataMaxLength = None**

This length corresponds the length when interpreted as bytes

**fuzzSenderThread = None**

Sending takes place in a loop in a separate thread

**fuzzingModeChanged** ()

This gets called if the ComboBox gets changed to update the active fuzzing mode. The other GUI elements will be set and enabled depending on the selected mode.

**fuzzingModeComboBoxValuePairs = None**

These values will be available in the fuzzing mode ComboBox

**generateRandomPacket** ()

This generates a random can.Message object using `tryBuildPacket()`

**Returns** can.Message object with random data (random ID, data length and data)

**itemAdderThread = None**

Adding items also takes place in a separate thread to avoid blocking the GUI thread

**packetBuildErrorCount = None**

Used to avoid spamming the log box when the user specified wrong parameters while sending

**prepareUI ()**

Prepare the tab specific GUI elements, add keyboard shortcuts and set a CANData instance

**sliderChanged ()**

This method gets called if one of the two length sliders (min. and max. value) are changed. dataMinLength and dataMaxLength will be directly updated and available to a running FuzzerThread.

**toggleFuzzing ()**

**This starts and stops fuzzing.**

- Starting: - Input values are read and validated - ItemAdderThread and FuzzSenderThread (see FuzzSenderThread) are started - Some GUI elements will be disabled
- Stopping: - The threads will be disabled - Disabled GUI elements will be enabled again

**toggleGUIElements (state)**

{En, Dis}able all GUI elements that are used to change fuzzer settings

**Parameters state** – Boolean value to indicate whether to enable or disable elements

**toggleLoopActive ()**

If there is a FuzzerThread sending then the tab title will be red.

**validateDataMaskInput ()**

**Validates the user specified data mask:**

- The length must be <= 16
- It must be a valid hex string
- Value will be padded to 16 chars (8 bytes)

**Returns** A validated data mask or None if it's not possible to validate the input

**validateIDMaskInput ()**

**Validates the user specified ID mask:**

- The length must be either 3 or 8
- It must be a valid hex string
- Has to be < 0x1FFFFFFF which is the max. value for extended frames

**Returns** A validated ID mask or None if it's not possible to validate the input

## 4.10 CANalyzat0r.Globals module

Created on May 17, 2017

@author: pschmied

**Globals.CANData = None**

Instance to interact with the bus

**Globals.db = None**

Object to handle db connections

```

Globals.knownPackets = {}
    Stores all known packets for the current project Key: CAN ID and data concatenated and separated with a “#”
    Value: Description

Globals.project = None
    Manage the currently selected project

Globals.textBrowserLogs = None
    Display logs in the GUI

Globals.ui = None
    The general UI

```

## 4.11 CANalyzat0r.ItemAdderThread module

Created on May 18, 2017

@author: pschmied

```

class ItemAdderThread.ItemAdderThread(receivePipe, tableModel, rawData, useTimes-
                                     tamp=True)
    Bases: PySide.QtCore.QThread

    This thread receives data from a process and emits a signal which causes the main thread to add the packets to
    the table.

    __init__(receivePipe, tableModel, rawData, useTimestamp=True)
        Initialize self. See help(type(self)) for accurate signature.

    appendRow = <PySide.QtCore.Signal object>
        Emit a signal to the main thread when items are ready to be added Parameters: valueList

    disable()
        This sets the enabled flag to False which causes the infinite loop in run() to exit.

    frameToRow(frame)
        Converts a can.Message object to a raw value list. This list will be emitted using the signal appendRow
        along with the table data and rawData list.

        Parameters frame – can.Message CAN frame

    run()
        As long as the thread is enabled: Receive a frame from the pipe and pass it to frameToRow().

    staticMetaObject = <PySide.QtCore.QMetaObject object>

```

## 4.12 CANalyzat0r.KnownPacket module

Created on May 22, 2017

@author: pschmied

```

class KnownPacket.KnownPacket(id, projectID, CANID, data, description)
    Bases: object

    This class is being used to handle known packet data. It's more comfortable to use a object to pass data than to
    use lists and list indexes. Please note that this costs much more performance, so please use lists if you have to
    deal with much data.

```

**\_\_init\_\_** (*id, projectID, CANID, data, description*)

Initialize self. See help(type(self)) for accurate signature.

**static fromJSON** (*importJSON*)

This class method creates a KnownPacket object using a JSON string.

**Parameters** **importJSON** – The JSON string containing the object data

**Returns** A KnownPacket object with the values set accordingly

**toComboBoxString** ()

Calculate a string that will be displayed in a ComboBox

**Returns** String representation of a KnownPacket object

**toJSON** ()

To export a KnownPacket, all data is being formatted as JSON. The internal attribute `__dict__` is being used to gather the data. This data will then be formatted.

**Returns** The formatted JSON data of the object

## 4.13 CANalyzat0r.Logger module

Created on May 17, 2017

@author: pschmied

**class** `Logger.LogHandler`

Bases: `logging.Handler`

To manage different log levels and custom logging to the log box/text browser, this class is needed.

**\_\_init\_\_** ()

Initializes the instance - basically setting the formatter to None and the filter list to empty.

**emit** (*record*)

This checks to loglevel and also logs to log box/text browser.

**Parameters** **record** – The record to log

**class** `Logger.Logger` (*className*)

Bases: `object`

This class implements a simple logger with a formatter.

**\_\_init\_\_** (*className*)

Along with set statements, a formatter is being applied here.

**Parameters** **className** – The tag for the logger

**getLogger** ()

**minLogLevel** = 20

## 4.14 CANalyzat0r.MainTab module

Created on May 22, 2017

@author: pschmied

**class** MainTab.MainTab

Bases: object

This class handles the logic of the main tab

**static** FDCheckboxChanged()

Clickhandler for the FD CheckBox which causes the FD SpinBox to be toggled

**static** VCANCheckboxChanged()

Clickhandler for the VCAN CheckBox which causes the SpinBox to be toggled.

**static** addApplicationStatus(*status*)

Add a new status to the status bar by name (ordered). If no status is present, it will display Strings.  
statusBarReady

**Parameters** *status* – The new status to add

**static** addVCANInterface()

Manually add a virtual CAN interface. This uses a syscall to `ip link`. If this call succeeds, a new `CANDataInstance` will be created using `createCANDataInstance()`. The detected CAN interfaces will be refreshed, too.

**static** applyGlobalInterfaceSettings()

Set the currently selected interface as the global interface. Also, the bitrate will be updated and GUI elements will be toggled. The `CANData` instances of all **inactive** tabs will also be set to the global interface.

**static** applyLogLevelSetting()

Set the minimum logging level to display messages for.

**static** detectCANInterfaces(*updateLabels=True*)

Detect CAN and VCAN interfaces available in the system. A syscall to `/sys/class/net` is being used for this. For every detected interface a new `CANData` instance will be created using `createCANDataInstance()`.

Also, interface labels and the global interface ComboBox will be updated.

**Parameters** *updateLabels* – Whether to update the interface labels or not

**static** easterEgg(*event*)

Nothing to see here :return: fun

**static** loadKernelModules()

Load kernel modules to interact with CAN networks (`can` and `vcn`).

**logger** = <Logger CANalyzat0r.MainTab (DEBUG)>

The tab specific logger

**static** populateProjects(*keepCurrentIndex=False*)

This populates the project ComboBox in the main tab.

**Parameters** *keepCurrentIndex* – If this is set to True, the previously selected index will be re-selected in the end

**static** prepareUI()

1. Setup the status bar
2. Detect CAN interfaces and preselect the VCAN CheckBox

3. Populate project ComboBoxes

4. Add the logo

**static preselectUseBitrateCheckBox ()**

Preselect the FD and VCAN CheckBox states.

**static removeApplicationStatus (status)**

Remove a status from the status bar. For statuses with multiple possible values (e.g. Sending (X Threads)) the search will be done using a substring search

**Parameters** **status** – The status to remove

**Returns**

**static removeVCANInterface ()**

This removes the currently selected VCAN interface. This uses a syscall to `ip link`. If the removed interface was the current global interface, the global interface will become None. :return:

**static setGlobalInterfaceStatus (red=False)**

Sets the text of the global interface status in the status bar. If the global CANData instance is None then the text will read “None”.

**Parameters** **red** – Optional; If this is set to True, the text will appear red. Else black.

**static setProject (wasDeleted=False, setNone=False)**

This sets the current project to the currently selected project in the corresponding ComboBox. Also, the status bar and project specific ComboBoxes and GUI Elements will be updated.

**Parameters**

- **wasDeleted** – This is set to True if the current selected project was deleted. This causes `Globals.project` to become None, too.
- **wasNull** – This is set to True, if the project has to be set to None. Default: False

**static setProjectStatus (projectName, red=False)**

Sets the text of the project status in the status bar.

**Parameters**

- **projectName** – The text to put as the new project name
- **red** – Optional; If this is set to True, the text will appear red. Else black.

**static setupStatusBar ()**

Add labels to the status bar and prepare it.

**statusBarActiveStatueses = []**

These text will appear in the status bar

**statusBarApplicationStatus = None**

Statusbar labels

**statusBarInterface = None**

**statusBarProject = None**

**static updateVCANButtons ()**

Update the text of the buttons to add and remove VCAN interfaces.



## 4.15 CANalyzat0r.ManagerTab module

Created on May 22, 2017

@author: pschmied

**class** ManagerTab.**ManagerTab** (*tabWidget*)

Bases: *AbstractTab.AbstractTab*

This class handles the logic of the manager tab

**\_\_init\_\_** (*tabWidget*)

Initialize self. See help(type(self)) for accurate signature.

**addKnownPacket** (*CANID=None, data=None, description=None*)

Save a known packet to the current project (and database) Default: Get values from the GUI elements. But you can also specify the values using the optional parameters.

### Parameters

- **CANID** – Optional: CAN ID
- **data** – Optional: Payload data
- **description** – Optional: Description of the known packet

### Returns

**clear** (*returnOldPackets=False*)

Clear the GUI table displaying PacketSets along with data lists.

**createDump** (*rawPackets=None*)

Save a new dump to the database. This creates a new PacketSet along with associated Packet objects. If only one packet is saved, the user will be asked if he wants to create a known packet entry for the just saved packet.

**Parameters rawPackets** – Optional: If this is not None, the values from *rawPackets* will be used instead of the data that is currently being displayed in the GUI table.

**createProject** ()

Create a new project and save it to the database. This also updates the project ComboBoxes.

**deleteDump** ()

Delete the currently selected PacketSet from the database. This also re-populates the table with the data of another dump (if existing)

**deleteProject** ()

Delete a project along with associated data. This also updates the project ComboBoxes.

**dumpRowIDs = None**

Kepps track between the association of table row <-> database id of the packet e.g. row 2 - database ID 5

**editKnownPacket** ()

Update a known packet with new specified values.

**editProject** ()

Update a project with new specified values.

**exportProject** ()

Export a project as JSON string to a textfile. The *toJSON()* method is called for every object to be exported.

**getDump** ()

Display the data of the selected PacketSet in the GUI table. This also updates *rawData*

**getKnownPacketsForCurrentProject ()**

(Re-)Populate the dictionary `Globals.knownPackets` with up-to-date data. If no project is set, the dictionary will be cleared only.

**handleCopy ()**

Pass the copy event to the Toolbox, but only if no data is being loaded

**handlePaste ()**

Pass the paste event to the Toolbox, but only if no data is being loaded

**importProject ()**

Import a project from a JSON file. The `fromJSON ()` method of every class is called to re-create objects.

**loadingData = None**

Disallow copying while loading data

**populateKnownPacketEditLineEdits ()**

Sets the GUI elements concerning editing known packets to the current selected known packet data.

**populateKnownPackets** (*keepCurrentIndex=False*)

Populate the known packet ComboBoxes (delete and edit).

**Parameters** **keepCurrentIndex** – Optional: Reselect a specific KnownPacket in the ComboBox

**populatePacketSets** (*IDtoChoose=None*)

Populate the dumps ComboBox in the dumps tab. Reloading dump data is being handled by the triggered event

**Parameters** **IDtoChoose** – Optional: Preselect a specific PacketSet in the ComboBox

**populateProjectEditLineEdits ()**

Sets the GUI elements concerning editing projects to the current selected project data.

**populateProjects** (*keepCurrentIndex=False*)

Populate the project ComboBoxes (delete, Edit, export project).

**Parameters** **keepCurrentIndex** – If this is set to True, the previously selected index will be reselected

**prepareUI ()**

Prepare the tab specific GUI elements, add keyboard shortcuts and set a CANData instance

**removeKnownPacket ()**

Remove a known packet from the current project (and the database)

**removeSelectedPackets ()**

Pass the remove requested event to the super class. After that, add the **database** IDs of the deleted packets to `dumpsDeletedPacketIDs`. The deleted rows will be removed from `dumpsRowIDs`, too.

**saveToFile ()**

Save the packets in the GUI table to a file in SocketCAN format.

**updateDump ()**

Users can change the data displayed in the GUI table. This method allows the changed data to be saved to the database.

## 4.16 CANalyzat0r.Packet module

Created on May 19, 2017

@author: pschmied

**class** `Packet.Packet` (*packetSetID, CANID, data, timestamp="", iface="", length=None, id=None*)

Bases: `object`

This class is being used to handle packet data. It's more comfortable to use a object to pass data than to use lists and list indexes. Please note that this costs much more performance, so please use lists if you have to deal with much data.

**\_\_init\_\_** (*packetSetID, CANID, data, timestamp="", iface="", length=None, id=None*)

The parameters `CANID` and `data` must be valid hex strings. If `length` is not specified, it will be calculated automatically.

**static fromJSON** (*importJSON*)

This class method creates a `Packet` object using a JSON string.

**Parameters** `importJSON` – The JSON string containing the object data

**Returns** A packet object with the values set accordingly

**static getDisplayDataLength** (*CANID, hexData*)

This makes sure that the displayed length is correct. If `CANID` is empty, the length will also be empty. If the length if `hexData` is odd, the length will read "INVALID" which prevents saving to the database. Else the length will be the amount of chars / 2

**Parameters**

- **CANID** – CAN ID
- **hexData** – Payload data as hex string

**Returns** The correct length as string

**lengthStringToInt** (*string*)

This makes sure that the specified length is an int :return: The length as integer (if possible) - else None by exception) :raises `ValueError` if the length isn't an integer

**toJSON** ()

To export a `Packet`, all data is being formatted as JSON. The internal attribute `__dict__` is being used to gather the data. This data will then be formatted.

**Returns** The formatted JSON data of the object

## 4.17 CANalyzat0r.PacketsDialog module

Created on Jun 23, 2017

@author: pschmied

**class** `PacketsDialog.PacketsDialog` (*packets=None, rawPacketList=None, returnPacketsAsRawList=True*)

Bases: `AbstractTab.AbstractTab`

This class handles the logic of the "manage packets" dialog. For example, this can be found in the `SnifferTabElement`.

`__init__` (*packets=None, rawPacketList=None, returnPacketsAsRawList=True*)

This basically just sets data and reads the widget from the `.ui` file.

#### Parameters

- **packets** – Optional: List that contains the elements that will be pre loaded into the GUI table in the following format: `<CAN ID>#<Data>`. This is used for the `SnifferTabElement`
- **rawPacketList** – Optional: Raw packet list that contains the elements that will be pre loaded into the GUI table. If this is specified, `packets` will be ignored.
- **returnPacketsAsRawList** – Boolean value indicating whether the displayed packets will be returned as raw packet list. If this is `False`, the values will be returned as list in the following format: `<CAN ID>#<Data>`.

**displayUniquePackets** (*IDOnly=False*)

Filter the currently displayed data for unique packets and display them on the table. :param `IDOnly`: If this is `True`, only the ID will be matched to compare data. This allows wildcard ignores.

**getUniqueIDs** ()

Filters all unique IDs out of `rawData` and displays them on the GUI table. Unique ID means that the data column will be ignored and left blank for wildcard ignores. This uses `displayUniquePackets()`.

**getUniquePackets** ()

Filters all unique packets out of `rawData` and displays them on the GUI table. This uses `displayUniquePackets()`.

**static getUniqueRawPackets** (*rawPacketList*)

Helper method to extract unique raw packets out of a given raw packet list which has been cleaned before (Only necessary data fields are present, all others are set to an empty string)

**Parameters** `rawPacketList` – The cleaned list of raw packets

**Returns** A list of unique raw packet lists

**open** ()

Show the widget, extract data and return it

**Returns** Raw packet list if the user didn't press Cancel and if `returnPacketsAsRawList` is `True`. Else: list of values of the following form: `<CAN ID>#<Data>` if the user didn't press Cancel. Else `None`.

**prepareUI** ()

Prepare the GUI elements and add keyboard shortcuts. Also, pre populate the table

## 4.18 CANalyzat0r.PacketSet module

Created on May 19, 2017

@author: pschmied

**class** `PacketSet.PacketSet` (*id, projectID, name, date=None*)

Bases: `object`

This class is being used to handle packet set data. It's more comfortable to use an object to pass data than to use lists and list indexes. Please note that this costs much more performance, so please use lists if you have to deal with much data.

`__init__` (*id, projectID, name, date=None*)

The date of a `PacketSet` will be automatically set to the current date as string

**static fromJSON** (*importJSON*)

This class method creates a PacketSet object using a JSON string.

**Parameters** *importJSON* – The JSON string containing the object data

**Returns** A PacketSet object with the values set accordingly

**toComboBoxString** ()

Calculate a string that will be displayed in a ComboBox

**Returns** String representation of a PacketSet object

**toJSON** ()

To export a PacketSet, all data is being formatted as JSON. The internal attribute `__dict__` is being used to gather the data. This data will then be formatted.

**Returns** The formatted JSON data of the object

## 4.19 CANalyzat0r.PacketTableModel module

Created on May 31, 2017

@author: pschmied

**class** PacketTableModel.**PacketTableModel** (*parent, dataList, header, readOnlyCols, IDColIndex=1, dataColIndex=2, lengthColIndex=3, timestampColIndex=4, descriptionColIndex=5, \*args*)

Bases: PySide.QtCore.QAbstractTableModel, PySide.QtCore.QObject

A custom TableModel is needed to allow efficient handling of **many** values.

**\_\_init\_\_** (*parent, dataList, header, readOnlyCols, IDColIndex=1, dataColIndex=2, lengthColIndex=3, timestampColIndex=4, descriptionColIndex=5, \*args*)

Initialize self. See help(type(self)) for accurate signature.

**appendRow** (*dataList=[], addAtFront=False, emit=True, resolveDescription=False*)

Inserts the dataList list into self.dataList to add a whole row with values at once.

**Parameters**

- **dataList** – The list containing data. The length must be equal to `rowCount()`.
- **addAtFront** – Values will be added to the front of self.dataList if this is True. Else: They will be appended at the end
- **emit** – Optional: If the GUI will be notified of the data change or not. This is used for batch imports where the GUI isn't notified after each row to increase speed Default: True (Emit everytime)
- **resolveDescription** – If this is set to True, the description of the potential known packet will be resolved. Default: False

**Returns** The description of the known packet. If `resolveDescription` is False, an empty string is returned. Else None.

This also emits the `dataChanged` and `layoutChanged` signals to let the GUI know that the data/layout has been changed.

**appendRows** (*rowList, addAtFront=False, resolveDescriptions=True*)

This allows appending a whole set of rows at once using the best possible speed

**Parameters**

- **rowList** – List of raw data lists to append
- **addAtFront** – Values will be added to the front of `self.dataList` if this is `True`. Else: They will be appended at the end
- **resolveDescriptions** – If this is set to `true`, the description for every packet will be resolved. Default: `True`

**Returns** If `resolveDescriptions` is `True`, a list of known packet descriptions will be returned. If no description for a particular packet can be resolved, an empty string will be inserted in the list to keep indexes. Else `None` will be returned

**cellChanged** = `<PySide.QtCore.Signal object>`

Emits `rowIndex` and `columnIndex` of the changed cell

**clear()**

Clears all managed data from the `dataList`. This is a shortcut to `setRowCount()` with parameter `0`.

**columnCount** (*parent=None*)

Returns the current column count by returning the length of the header list.

**Parameters** **parent** – Dummy parameter to keep the needed signature

**Returns** The column count as integer

**data** (*index, role*)

Return managed data depending on the `role` parameter.

**Parameters**

- **index** – Index object containing row and column index
- **role** – The display role that requests data

**Returns**

- If the index is invalid: `None`
- `AlignCenter` if `role = TextAlignmentRole`
- Column data if `role = DisplayRole` or `EditRole`

**flags** (*index*)

Return the flags for cell at a given index.

**Parameters** **index** – Index object containing row and column index

**Returns** A flags object containing whether an object is editable, selectable or enabled

**getValue** (*rowIndex, colIndex*)

Get the data from the table at the given indexes.

**Parameters**

- **rowIndex** – Row index
- **colIndex** – Column index

**Returns** The data at the specified index (if possible); Else `None`

**headerData** (*headerIndex, orientation, role*)

Returns the header data to properly display the managed data on the GUI.

**Parameters**

- **headerIndex** – Which column of the data is requested
- **orientation** – This can be either `Horizontal` or `Vertical`:

- Horizontal: Return a value from `self.header`
- Vertical: Return the `headerIndex`
- **role** – This is always expected to be `DisplayRole`

**Returns** See `orientation`. `None` is returned if `orientation` or `role` do not match

**insertRow** (*dataList=[]*)

This is just an alias to `appendRow()` for compatibility.

**Parameters** **dataList** – A list that stores the data that will be added

**removeRow** (*rowIndex*)

Removes the specified row from the table model. This also emits the `layoutChanged` signal to let the GUI know that the layout has been changed.

**Parameters** **rowIndex** – The row index to delete

**removeRows** (*rowIndexes*)

Remove multiple rows at once.

**Parameters** **rowIndexes** – The rows that will be deleted

**rowCount** (*parent=None*)

Returns the current row count by returning the length of the data list.

**Parameters** **parent** – Dummy parameter to keep the needed signature

**Returns** The row count as integer

**setData** (*index, value, role=PySide.QtCore.Qt.ItemDataRole.EditRole*)

This gets called to change the element on the GUI at the given indexes. This also emits the `layoutChanged` signal to let the GUI know that the layout has been changed.

**Parameters**

- **index** – Index object containing row and column index
- **value** – The new value
- **role** – Optional: The role calling this method. Default: `EditRole`

**Returns** True if the operation succeeded

**setRowCount** (*count*)

Sets the row count by removing lines / adding empty lines. This also emits the `layoutChanged` signal to let the GUI know that the layout has been changed.

**Parameters** **count** – The desired amount of rows

**setText** (*rowIndex, colIndex, data*)

Sets the text of at the given indexes. If `data` is `None` the text will be an empty string. This method also emits the `dataChanged` and `layoutChanged` signals to let the GUI know that the data/layout has been changed.

**Parameters**

- **rowIndex** – Row index
- **colIndex** – Column index
- **data** – New data for the column

**sort** (*colIndex, order*)

Sort the data by given column number. This also emits the `layoutChanged` signal to let the GUI know that the layout has been changed.

**Parameters**

- **colIndex** – The column index to sort for
- **order** – Either `DescendingOrder` or `AscendingOrder`

**staticMetaObject** = `<PySide.QtCore.QMetaObject object>`

## 4.20 CANalyzat0r.Project module

Created on May 19, 2017

@author: pschmied

**class** `Project.Project` (*id, name, description*)

Bases: `object`

This class is being used to handle project data. It's more comfortable to use a object to pass data than to use lists and list indexes. Please note that this costs much more performance, so please use lists if you have to deal with much data.

**\_\_init\_\_** (*id, name, description*)

This just sets the attributes to the passed parameters.

**static fromJSON** (*importJSON*)

This class method creates a project object using a JSON string.

**Parameters** **importJSON** – The JSON string containing the object data

**Returns** A project object with the values set accordingly

**toComboBoxString** ()

Calculate a string which will be displayed in the ComboBoxes.

**Returns** String representation of the object

**toJSON** ()

To export a Project, all data is being formatted as JSON. The internal attribute `__dict__` is being used to gather the data. This data will then be formatted.

**Returns** The formatted JSON data of the object

## 4.21 CANalyzat0r.SearcherTab module

Created on May 17, 2017

@author: pschmied

**class** `SearcherTab.SearcherTab` (*tabWidget*)

Bases: `AbstractTab.AbstractTab`

This class handles the logic of the filter tab

**\_\_init\_\_** (*tabWidget*)

Initialize self. See `help(type(self))` for accurate signature.

**askActionPerformed** ()

Ask the user if the action has been performed using a `MessageBox`

**Returns** True if the user pressed yes, else False



**askWhichAction ()**

**Ask the user what to do if no chunk worked:**

- Try again
- Re-test the current last working chunk
- Cancel

**Returns** An integer value indicating the pressed button: - 0 if the user wants to try again - 1 if the user wants to re-test - 2 if the user wants to cancel

**beep ()**

To play a sound after sending has been finished.

**clear** (*returnOldPackets=False*)

Clear the GUI table and all associated data lists

**downwardsSearch = None**

We first search downwards in the binary search tree. If this doesn't succeed, we use randomization and begin to search upwards.

**enterWhenReady ()**

Block the GUI thread until the user pressed the button on the MessageBox.

**lastWorkingChunk = None**

The currently smallest known set of packets that cause a specific action. Values are lists with raw data

**outputRemainingPacket** (*packet*)

Show the passed packet on the GUI table.

**Parameters** **packet** – The raw packet to display

**outputRemainingPackets** (*rawPackets*)

Show all passed packets on the GUI table. Note: This also sets `rawData` to the passed set of packets.

**Parameters** **packet** – List of raw packets to display

**searchPackets ()**

**This starts the whole searching routine and sets up things first:**

1. Set a `CANData` instance
2. Get user input values
3. Walk down the binary search tree: Try to find a specific packet for an action
4. If 1 packet has been found: output the packet
5. If not: Get the last working chunk of packets that worked. Use shuffling and new values for the chunk amount to find a minimal set of packets

**sendAndSearch** (*chunkAmount=2*)

**Use the remaining data to search for relevant packets:**

1. Setup a progress bar
2. Split the raw packet list in the desired amount of chunks
3. Test each chunk (Newest packets first) and ask the user if it worked
4. If it worked: Set `lastWorkingChunk` to the last tested chunk and return `True`
5. Else: Return `False` if all other chunks failed too.

**Parameters** **chunkAmount** – The amount of chunks to generate from the given data list

**Returns** True if a specific chunk worked, else False

**splitLists** (*lst*, *chunkAmount=2*)

Split a list into a specific amount of chunks

**Parameters**

- **lst** – The list to split
- **chunkAmount** – Desired amount of chunks

**Returns** List of chunks (List of lists in this case)

**toggleGUIElements** (*state*)

{En, Dis}able all GUI elements that are used to change searcher settings

**Parameters** **state** – Boolean value to indicate whether to enable or disable elements

## 4.22 CANalyzat0r.SenderTab module

Created on May 22, 2017

@author: pschmied

**class** `SenderTab.SenderTab`

Bases: `object`

This class handles the logic of the sender tab. Subtabs are being handled in `SenderTabElement`.

**CANData** = `None`

The tab specific `CANData` instance

**active** = `False`

**static** `addSender` (*senderTabName=None*)

Appends a new sender tab to the sub tab bar.

**Parameters** **senderTabName** – Optional; The displayed name of the tab. If this is `None`, the user is requested to enter a name

**static** `addSenderWithData` (*listOfRawPackets=None*, *listOfPackets=None*)

Uses `addSender()` to add a new sender tab with data already filled in into the GUI table. You must specify `listOfRawPackets` **or** `listOfPackets`. If both are specified, `listOfRawPackets` will be used.

**Parameters**

- **listOfRawPackets** – Optional; List of raw packets to add to the table.
- **listOfPackets** – Optional; List of packet objects to add to the table.

**Returns**

**currentlySendingTabs** = `0`

Used to handle the font color of the sender tab

**classmethod** `handleInterfaceSettingsDialog()`

This invokes `handleInterfaceSettingsDialog()` for the class

**indexInMainTabBar** = `2`

The index of the sender tab in the main tab bar

**labelInterfaceValue = None**

**logger = <Logger CANalyzat0r.SenderTab (DEBUG)>**  
The tab specific logger

**static prepareUI ()**  
Prepare the tab specific GUI elements, add sender tab and keyboard shortcuts. Also set a CANData instance.

**static removeSender (senderTabElement)**  
Remove a sender from the sub tab bar. This method gets called from an instance of SenderTabElement by removeSender ().

**Parameters senderTabElement** – The SenderTabElement instance to remove

**static sendSinglePacket ()**  
Sends a single packet using the specified interface. All packet values are read from the GUI elements.

**senderTabs = []**  
Consists of all SenderTabElements

**classmethod setInitialCANData ()**  
This invokes setInitialCANData () for the class

**static toggleActive ()**  
If there is at least one tab sending then the tab bar title will be red.

**static toggleGUIElements (state)**  
{En, Dis}able all GUI elements that are used to change filter settings

**Parameters state** – Boolean value to indicate whether to enable or disable elements

**classmethod updateCANDataInstance (CANDataInstance, delegate=False)**  
This invokes updateCANDataInstance () for the class

**Parameters**

- **CANDataInstance** – The new CANData instance
- **delegate** – Boolean indicating if all sender sub tabs will be updated too. Default: False

**classmethod updateInterfaceLabel ()**  
This invokes updateInterfaceLabel () for the class

## 4.23 CANalyzat0r.SenderTabElement module

Created on May 23, 2017

@author: pschmied

**class** SenderTabElement.**SenderTabElement** (*tabWidget, tabName*)  
Bases: *AbstractTab.AbstractTab*

This class handles the logic of the sender sub tab. The main tab is being handled in SenderTab.

**\_\_init\_\_** (*tabWidget, tabName*)  
Set all passed data. Also, add the own send button to SenderTab.sendButtonList to allow managing it globally.

**Parameters tabWidget** – The element in the tab bar. **Not** the table widget.

**amountThreadsRunning = 0**  
Amount of sending threads running to display in the status bar

**getTabIndex ()**

Get the **current** tab index of the sub tab element

**Returns** The tab index of the sender tab

**loopSenderThread = None**

The thread that runs when sending takes place in a loop

**prepareUI ()**

Prepare the tab specific GUI elements, add keyboard shortcuts and set a CANData instance

**removeSender ()**

This gets called when the remove sender button is pressed on the sub tab. This stops the sender thread and calls the parents method (`removeSender ()`) to remove the sender form the tab bar.

**sendAll ()**

Send all packets in the GUI table. By default, this just sends the packet once using simple calls. If the user requests to send the packets in a loop, an instance of `LoopSenderThread` is being used to send the packets. Also, GUI elements like the status bar are being updated.

**sendButtonList = []**

Static attribute of send buttons to manage {en, dis}abled states

**setSendButtonState (state)**

This sets the enabled state of the send button.

**Parameters state** – The desired enabled state as boolean value

**stopSending ()**

This stops the currently running instance of `LoopSenderThread` from sending. Also, GUI elements like the status bar are being updated.

**toggleGUIElements (state)**

{En, Dis}able all GUI elements that are used to change filter settings

**Parameters state** – Boolean value to indicate whether to enable or disable elements

**toggleLoopActive ()**

Toggles the current sub tab to (in)active. This also calls `toggleActive ()` to manage the color of the main tab (parent tab).

**updateStatusBar ()**

Updates the status bar label to display the correct amount of sending tabs (if any)

## 4.24 CANalyzat0r.SenderThread module

Created on Jun 08, 2017

@author: pschmied

**class** `SenderThread.FuzzSenderThread (sleepTime, fuzzerSendPipe, CANData, threadName)`

Bases: `PySide.QtCore.QThread`

Spawns a new thread that will send random data in a loop.

**\_\_init\_\_ (sleepTime, fuzzerSendPipe, CANData, threadName)**

Initialize self. See `help(type(self))` for accurate signature.

**disable ()**

This sets the `enabled` flag to `False` which causes the main loop to terminate.

```

run ()
    Send the packets in a loop and wait accordingly until the thread is disabled.

staticMetaObject = <PySide.QtCore.QMetaObject object>

class SenderThread.LoopSenderThread (packets, sleepTime, CANData, threadName)
    Bases: PySide.QtCore.QThread

    Spawns a new thread that will send the passed packets in a loop.

    __init__ (packets, sleepTime, CANData, threadName)
        Initialize self. See help(type(self)) for accurate signature.

    disable ()
        This sets the enabled flag to False which causes the main loop to terminate.

run ()
    Send the packets in a loop and wait accordingly until the thread is disabled.

staticMetaObject = <PySide.QtCore.QMetaObject object>

```

## 4.25 CANalyzat0r.Settings module

Created on May 17, 2017

@author: pschmied

```

Settings.APP_NAME = 'CANalyzat0r'
    The application name

```

```

Settings.APP_VERSION = '1.0'
    The application version

```

```

Settings.DB_NAME = '../data/database.db'
    The relative path of the SQLite database file

```

```

Settings.DB_PATH = '../data/database.db'
    Optionally we can specify a different database path

```

```

Settings.FORKME_PATH = './ui/icon/forkme.png'
    Where to find the “Fork Me” icon

```

```

Settings.GITHUB_URL = 'https://github.com/schutzwerk/CANalyzat0r'
    Where to find this project in GitHub

```

```

Settings.ICON_PATH = './ui/icon/icon.png'
    Where to find the app icon

```

```

Settings.LOGO_PATH = './ui/icon/swlogo_small.png'
    Where to find the company logo

```

## 4.26 CANalyzat0r.SnifferProcess module

Created on May 18, 2017

@author: pschmied

```
class SnifferProcess.SnifferProcess (snifferSendPipe, sharedEnabledFlag, snifferName, CAN-  
                                     Data=None)
```

Bases: multiprocessing.context.Process

Spawn a new process that will sniff packets from the specified CANData instance. Captured data will be transmitted via the snifferSendPipe.

```
__init__ (snifferSendPipe, sharedEnabledFlag, snifferName, CANData=None)  
    Set the passed parameters.
```

### Parameters

- **snifferSendPipe** – The multiprocessing pipe to send received data to
- **sharedEnabledFlag** – The multiprocessing value to handle disabling
- **snifferName** – The name of the sniffer process, used for logging
- **CANData** – Optional: The CANData instance to query for data. If this is not specified, the global interface is being used

```
run ()
```

As long as the process hasn't been disabled: Read a frame using readPacketAsync() and transmit the received can.Message object via the pipe.

## 4.27 CANalyzat0r.SnifferTab module

Created on May 18, 2017

@author: pschmied

```
class SnifferTab.SnifferTab
```

Bases: object

This class handles the logic of the sniffer tab. Subtabs are being handled in SnifferTabElement.

```
static addSniffer (snifferTabName)
```

Appends a new sniffer tab to the sub tab bar.

**Parameters** **snifferTabName** – The displayed name of the tab. Normally, this corresponds to the CAN interface the tab is managing

```
static clearAndAddPlaceholder ()
```

Add a placeholder where normally sniffer tabs are displayed

```
indexInMainTabBar = 1
```

The index of the sniffer tab in the main tab bar

```
logger = <Logger CANalyzat0r.SnifferTab (DEBUG)>
```

The tab specific logger

```
static prepareUI ()
```

This adds a placeholder if no instance of SnifferTabElement was created previously.

**static removeSniffer** (*snifferTabElement=None, snifferTabName=None*)

Remove a sniffer from the sub tab bar. This method gets called from an instance of `SnifferTabElement` by `removeSender()`. One can either specify `snifferTabElement` or `snifferTabName` to delete a tab. If both are used, the object parameter is used.

#### Parameters

- **senderTabElement** – Optional: The `SnifferTabElement` instance to remove
- **snifferTabName** – Optional: The name of the `SenderTabElement` instance to remove

**snifferTabs** = {}

Consists of all `SnifferTabElements`, interface names as key

**static toggleActive** ()

If there is at least one tab sniffing then the tab bar title will be red.

**classmethod updateInterfaceLabel** ()

This invokes `updateInterfaceLabel()` every sniffer tab

## 4.28 CANalyzat0r.SnifferTabElement module

Created on Jun 16, 2017

@author: pschmied

**class** `SnifferTabElement.SnifferTabElement` (*tabWidget, tabName, ifaceName=None*)

Bases: `AbstractTab.AbstractTab`

This class handles the logic of the sniffer sub tab. The main tab is being handled in `SnifferTab`.

**\_\_init\_\_** (*tabWidget, tabName, ifaceName=None*)

Set parameters and initialize the `CANData` instance

#### Parameters

- **tabWidget** – The element in the tab bar. **Not** the table widget.
- **tabName** – The name of the tab
- **ifaceName** – Optional: The interface name. If this is `None`, then the `tabName` will be used

**addPacket** (*valueList, addAtFront=True, append=True, emit=True, addToRawDataOnly=False, ignorePacketRate=False*)

Override the parents method to add packets at front and to update the counter label. If too much data is received, the data will be added after sniffing to prevent freezes. Also, only add packets if the data isn't present in `self.ignoredPackets`

**Parameters ignorePacketRate** – Additional optional parameter: Boolean value indicating whether the rate of packets per second will be ignored or not. This is set to `False` by Default. We need to set it to `True` to process `self.valueList` after sniffing if too much data was received.

**amountThreadsRunning** = 0

Amount of sniffing threads running to display in the status bar

**clear** (*returnOldPackets=False*)

Clear the currently displayed data on the GUI and in the lists.

**Parameters** **returnOldPackets** – Optional: If this is True then the previously displayed data will be returned as raw data list. Default is False

**getPacketCount ()**

This uses a call to `/sys/class/net/<ifaceName>/statistics/rx_packets` to return the number of total received packets of the current interface

**Returns** Received packet count of the current interface as integer

**handleInterfaceSettingsDialog** (*allowOnlyOwnInterface=True*)

Override the parents method to only allow the currently set CAN interface

**handleManageIgnoredPacketsDialog ()**

Open a dialog to manage ignored packets when sniffing

**removeSniffer ()**

This gets called when associated interface disappears after a re-check. This stops the sniffer thread and calls the parents method (`removeSender ()`) to remove the sniffer from the tab bar.

**tabIndex ()**

Get the **current** tab index of the sub tab element

**Returns** The tab index of the sniffer tab

**terminateThreads ()**

This stops the processes/threads called `snifferProcess` and `itemAdderThread`. Also, the CAN-Data instance will be set to inactive and GUI elements will be toggled.

**toggleActive ()**

Toggles the current sub tab to (in)active. This also calls `toggleActive ()` to manage the color of the main tab (parent tab).

**toggleSniffing ()**

This starts and stops sniffing for a specific sniffer tab. Instances of `ItemAdderThread` and `SnifferProcess` are used to gather and display data.

**updateStatusBar ()**

Updates the status bar label to display the correct amount of sniffer tabs (if any)

## 4.29 CANalyzat0r.Strings module

Created on May 17, 2017

@author: pschmied

## 4.30 CANalyzat0r.Toolbox module

Created on May 22, 2017

@author: pschmied

**class** `Toolbox.Toolbox`

Bases: `object`

This class offers helpful static methods that every tab can use to unify program logic and simplify code.

**static** `askUserConfirmAction ()`

Spawns a `MessageBox` that asks the user to confirm an action



**Returns** True if the user has clicked on Yes, else False

**static checkProjectIsNone** (*project=-1*)

Checks if a project is None and displays a MessageBox if True.

**Parameters** **project** – Optional. The project to check. Default: -1 wich causes the global project to be checked

**Returns** Boolean value indicating whether the checked project is None

**static getKnownPacketDescription** (*CANID, data*)

Get a description for a known packet. This will use the dictionary defined in `Globals` to find data

**Parameters**

- **CANID** – CAN ID
- **data** – Data

**Returns** The description if one can be found, else an empty string

**static getPacketDictIndex** (*CANID, data*)

Calculates the index of a packet with a specific CAN ID and data in a dictionary.

**Parameters**

- **CANID** – CAN ID
- **data** – Data

**Returns** The index of a packet in a dictionary

**static getSaveFileName** (*dialogTitle*)

Spawns a “save file dialog” to get a file path to save to.

**Parameters** **dialogTitle** – The displayed dialog title

**Returns** The file path of the selected file

**static getWorkingDialog** (*text*)

Generates a working dialog object which blocks the UI.

**Parameters** **text** – Text to display while working

**Returns** The created working dialog widget

**interfaceDialogWidget = None**

The dialog widget to set the interface settings

**static interfaceSettingsDialog** (*currentCANData, CANDataOverrideValues=None*)

Handles the logic of the interface settings dialog.

**Parameters** **CANDataOverrideValues** – Optional: List of CANData instances that will be selectable instead of all values.

**Returns** A new CANData instances with the selected values. None if no editable CANData instance exists

**static interfaceSettingsDialogCheckBoxChanged** (*state*)

Gets called when the use VCAN CheckBox of the interface dialog gets changed to handle the enabled state of the bitrate SpinBox

**Parameters** **state** – Not used, state is determined by `isChecked`

**static interfaceSettingsDialogComboBoxChanged** ()

Gets called when the interface ComboBox of the interface dialog gets changed to pre-populate the GUI elements accordingly.

**static interfaceSettingsDialogFDCheckBoxChanged** (*state*)

Gets called when the “use FD” CheckBox of the interface dialog gets changed to handle the enabled state of the FD bitrate SpinBox

**Parameters** *state* – Not used, state is determined by *isChecked*

**static isHexString** (*hexString*)

Checks if a *hexString* is a valid hex string of base 16

**Parameters** *hexString* – The hex string

**Returns** Boolean value indicating the correctness of the hex string

**logger** = <Logger CANalyzat0r.Toolbox (DEBUG)>

The toolbox also has its own logger instance

**mp3Processes** = {}

Used to keep track of the processes that play mp3 files. Key = filepath, value: multiprocessing Process object

**static playMP3** (*filePath*)

Plays an mp3 sound file using ffmpeg

**Parameters** *filePath* – Path of the mp3 file

**static populateInterfaceComboBox** (*comboBoxWidget*, *reselectCurrentItem=True*, *ignoreActiveInstances=False*)

Inserts all available interface values into the passed ComboBox widget

**Parameters**

- **comboBoxWidget** – The GUI element to fill with items
- **reselectCurrentItem** – Optional: If this is true, the previously selected index will be re-selected Default: True

**static stopMP3** (*filePath*)

Stops the playback of a given mp3 file :param *filePath*: Path of the mp3 file

**static tableExtractAllRowData** (*table*)

Get **all** contents of a GUI table

**Parameters** *table* – The *QTableView* object to gather data from

**Returns** A list of raw row data -> List of lists

**static tableExtractSelectedRowData** (*table*)

Get the **selected** contents of a GUI table

**Parameters** *table* – The *QTableView* object to gather data from

**Returns** A list of raw row data -> List of lists

**static toggleDisabledProjectGUIElements** ()

This toggles specific GUI elements that should only be active if a project has been selected

**static toggleDisabledSenderGUIElements** ()

This toggles specific GUI elements that should only be active if a *CANData* instance is present

**static updateCANDataInstances** (*CANDataInstance*)

Calls *updateCANDataInstance* for every tab.

**Parameters** *CANDataInstance* – The new *CANData* instance

**static updateInterfaceLabels** ()

Calls *updateInterfaceLabel* for every tab.

**static widgetFromUIFile** (*filePath*)

Reads an `.ui` file and creates a new widget object from it.

**Parameters** `filePath` – Where to find the `.ui` file

**Returns** The new created widget

**static yesNoBox** (*title, text*)

Spawns a `MessageBox` that asks the user a Yes-No question.

**Returns** True if the user has clicked on Yes, else False

## 4.31 CANalyzat0r.mainWindow module

## 4.32 CANalyzat0r.AbstractTab module

Created on Jun 26, 2017

@author: pschmied

```
class AbstractTab.AbstractTab(tabWidget, loggerName, readOnlyCols, packetTableView-
                             Name, labelInterfaceValueName=None, CANData=None,
                             hideTimestampCol=True, sendToSenderContextMenu=True,
                             saveAsPacketSetContextMenu=True, allowTableCopy=True,
                             allowTablePaste=True, allowTableDelete=True)
```

Bases: `object`

This is a base class for *most* tabs. If you're using a tab that uses the following things, you can use this class:  
 - Non-static tab - you create instances from it - a `QTableView` to display data - `rawData` as raw packet list -  
 Copy, paste and/or delete actions by shortcuts and context menus

Just take care of `packetTableViewName` and `labelInterfaceValueName` as they're necessary for this to work.

**CANData = None**

The tab specific `CANData` instance

```
__init__(tabWidget, loggerName, readOnlyCols, packetTableViewName, labelInterfaceVal-
         ueName=None, CANData=None, hideTimestampCol=True, sendToSenderCon-
         textMenu=True, saveAsPacketSetContextMenu=True, allowTableCopy=True, al-
         lowTablePaste=True, allowTableDelete=True)
```

Initialize self. See `help(type(self))` for accurate signature.

**active = None**

Whether the tab is currently active (using `CANData`)

```
addPacket(valueList, addAtFront=False, append=True, emit=True, addToRawDataOnly=False)
```

Add a packet to the GUI table.

**Parameters**

- **valueList** – Packet data as raw value list
- **addAtFront** – Optional. Indicates whether the packets will be inserted at the start of `rawData`. Default: False
- **append** – Optional. Indicates whether data will be added to `self.rawData` or not. Default: True
- **emit** – Optional. Indicates whether the GUI will be notified using signals. Default: True

- **addToRawDataOnly** – Optional. Indicates whether only `self.rawData` will be updated, ignoring the packet table model. Default: False

**applyNewKnownPackets** ()

Apply new known packets which have been saved in the mean time. This reloads the packets into the GUI table.

**clear** (*returnOldPackets=False*)

Clear the currently displayed data on the GUI and in the rawData list

**Parameters** **returnOldPackets** – If this is true, then the previously displayed packets will be returned as raw data list

**Returns** Previously displayed packets as raw data list (if returnOldPackets is True), else an empty list

**handleCellChanged** (*rowIndex, colIndex*)

To update the rawData element and to put the length of the changed data field into the length field.

**Parameters**

- **rowIndex** – The changed row
- **colIndex** – The changed column

**Returns**

**handleCopy** ()

Handle copying **selected** rows from a GUI table. This copies the raw data list **to the clipboard**.

**handleInterfaceSettingsDialog** (*allowOnlyOwnInterface=False*)

Open a dialog to change interface settings and set the updated CANData instance.

**Parameters** **allowOnlyOwnInterface** – If this is true, you can only edit the CANData instance that is already selected for the current tab. This is being used for the sniffer tabs.

**handlePaste** ()

Handle pasting rows into a GUI table. Data is being gathered from the clipboard and be of the following types: - Raw data list (list of lists which consist of column data) - Parsing takes place asynchronously - SocketCAN format (see `SocketCANFormat`)

**handleRightClick** ()

This spawns a custom context menu right next to the cursor if a user has right clicked on a table cell.

**loggerName** = None

The tab specific logger

**manualAddPacket** ()

Manually add an empty packet row to the GUI table. This also updates `rawData`.

**packetTableModel** = None

Custom packet model of the GUI table

**prepareUI** ()

Prepare the tab specific GUI elements, add keyboard shortcuts and set a CANData instance

**rawData** = None

Raw packet data that corresponds to the data displayed in the GUI table

**readOnlyCols** = None

Tab specific read only columns as list of indexes

**removeSelectedPackets ()**

Remove selected rows from a table and also delete those rows from a `rowData` list. :return: A list of indexes of the removed rows. None if no rows have been selected

**setInitialCANData ()**

Try to get initial an initial CANData instance. This method also updates GUI elements.

**Returns** A boolean value which indicates the success

**tabWidget = None**

The specific GUI tab

**toggleGUIElements (state)****updateCANDataInstance (CANDataInstance)**

Updates the tab specific CANData instance to the passed parameter. This only takes place if the tab is not active to prevent errors. This also calls [updateInterfaceLabel \(\)](#) to update the label.

**Parameters CANDataInstance** – The new CANData instance

**updateInterfaceLabel ()**

Set the text of the interface label to the updated value, if the label is present. Uses the text “None” if no interface is set.



## USED LIBRARIES AND FILES

### 5.1 Libs

- pythoh-can
- PySide: Python for Qt
- `ffmpeg` <<https://ffmpeg.org/>`\_
- can-utils
- Sphinx
- Sphinx RTD Theme

### 5.2 Misc

- Car Icon Flat
- Orange flame 2 icon
- Soylent red flame 2 icon





## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### a

AboutTab, [13](#)  
AbstractTab, [47](#)

### c

ComparerTab, [13](#)

### d

Database, [14](#)

### f

FilterTab, [20](#)  
FuzzerTab, [23](#)

### g

Globals, [24](#)

### i

ItemAdderThread, [25](#)

### k

KnownPacket, [25](#)

### l

Logger, [26](#)

### m

MainTab, [27](#)  
mainWindow, [47](#)  
ManagerTab, [29](#)

### p

Packet, [31](#)  
PacketsDialog, [31](#)  
PacketSet, [32](#)  
PacketTableModel, [33](#)  
Project, [36](#)

### s

SearcherTab, [36](#)  
SenderTab, [38](#)  
SenderTabElement, [39](#)

SenderThread, [40](#)

Settings, [41](#)

SnifferProcess, [42](#)

SnifferTab, [42](#)

SnifferTabElement, [43](#)

Strings, [44](#)

### t

Toolbox, [44](#)



## Symbols

\_\_init\_\_() (*AbstractTab.AbstractTab* method), 47  
 \_\_init\_\_() (*ComparerTab.ComparerTab* method), 14  
 \_\_init\_\_() (*Database.Database* method), 14  
 \_\_init\_\_() (*FilterTab.DataAdderThread* method), 20  
 \_\_init\_\_() (*FilterTab.FilterTab* method), 21  
 \_\_init\_\_() (*FuzzerTab.FuzzerTab* method), 23  
 \_\_init\_\_() (*ItemAdderThread.ItemAdderThread* method), 25  
 \_\_init\_\_() (*KnownPacket.KnownPacket* method), 25  
 \_\_init\_\_() (*Logger.LogHandler* method), 26  
 \_\_init\_\_() (*Logger.Logger* method), 26  
 \_\_init\_\_() (*ManagerTab.ManagerTab* method), 29  
 \_\_init\_\_() (*Packet.Packet* method), 31  
 \_\_init\_\_() (*PacketSet.PacketSet* method), 32  
 \_\_init\_\_() (*PacketTableModel.PacketTableModel* method), 33  
 \_\_init\_\_() (*PacketsDialog.PacketsDialog* method), 31  
 \_\_init\_\_() (*Project.Project* method), 36  
 \_\_init\_\_() (*SearcherTab.SearcherTab* method), 36  
 \_\_init\_\_() (*SenderTabElement.SenderTabElement* method), 39  
 \_\_init\_\_() (*SenderThread.FuzzSenderThread* method), 40  
 \_\_init\_\_() (*SenderThread.LoopSenderThread* method), 41  
 \_\_init\_\_() (*SnifferProcess.SnifferProcess* method), 42  
 \_\_init\_\_() (*SnifferTabElement.SnifferTabElement* method), 43

## A

AboutTab (class in *AboutTab*), 13  
 AboutTab (module), 13  
 AbstractTab (class in *AbstractTab*), 47  
 AbstractTab (module), 47  
 active (*AbstractTab.AbstractTab* attribute), 47  
 active (*SenderTab.SenderTab* attribute), 38  
 addApplicationStatus() (*MainTab.MainTab* static method), 27

addKnownPacket() (*ManagerTab.ManagerTab* method), 29  
 addPacket() (*AbstractTab.AbstractTab* method), 47  
 addPacket() (*FuzzerTab.FuzzerTab* method), 23  
 addPacket() (*SnifferTabElement.SnifferTabElement* method), 43  
 addSender() (*SenderTab.SenderTab* static method), 38  
 addSenderWithData() (*SenderTab.SenderTab* static method), 38  
 addSniffedNoise() (*FilterTab.FilterTab* method), 21  
 addSniffedPacketToSample() (*FilterTab.FilterTab* method), 21  
 addSniffer() (*SnifferTab.SnifferTab* static method), 42  
 addVCANInterface() (*MainTab.MainTab* static method), 27  
 amountThreadsRunning (*SenderTabElement.SenderTabElement* attribute), 39  
 amountThreadsRunning (*SnifferTabElement.SnifferTabElement* attribute), 43  
 analyze() (*FilterTab.FilterTab* method), 21  
 APP\_NAME (in module *Settings*), 41  
 APP\_VERSION (in module *Settings*), 41  
 appendRow (*ItemAdderThread.ItemAdderThread* attribute), 25  
 appendRow() (*PacketTableModel.PacketTableModel* method), 33  
 appendRows() (*PacketTableModel.PacketTableModel* method), 33  
 applyGlobalInterfaceSettings() (*MainTab.MainTab* static method), 27  
 applyLogLevelSetting() (*MainTab.MainTab* static method), 27  
 applyNewKnownPackets() (*AbstractTab.AbstractTab* method), 48  
 askActionPerformed() (*SearcherTab.SearcherTab* method), 36  
 askUserConfirmAction() (*Toolbox.Toolbox* static method), 44  
 askWhichAction() (*SearcherTab.SearcherTab*

*method*), 36

## B

`beep()` (*SearcherTab.SearcherTab method*), 37

`browseGitHub()` (*AboutTab.AboutTab static method*), 13

`browseSW()` (*AboutTab.AboutTab static method*), 13

## C

`CANData` (*AbstractTab.AbstractTab attribute*), 47

`CANData` (*in module Globals*), 24

`CANData` (*SenderTab.SenderTab attribute*), 38

`cellChanged` (*PacketTableModel.PacketTableModel attribute*), 34

`checkDB()` (*Database.Database method*), 14

`checkProjectIsNone()` (*Toolbox.Toolbox static method*), 45

`checkTablesPresentStatement` (*Database.DatabaseStatements attribute*), 17

`clear()` (*AbstractTab.AbstractTab method*), 48

`clear()` (*FilterTab.FilterTab method*), 21

`clear()` (*FuzzerTab.FuzzerTab method*), 23

`clear()` (*ManagerTab.ManagerTab method*), 29

`clear()` (*PacketTableModel.PacketTableModel method*), 34

`clear()` (*SearcherTab.SearcherTab method*), 37

`clear()` (*SnifferTabElement.SnifferTabElement method*), 43

`clearAndAddPlaceholder()` (*SnifferTab.SnifferTab static method*), 42

`collectNoise()` (*FilterTab.FilterTab method*), 21

`columnCount()` (*PacketTableModel.PacketTableModel method*), 34

`compare()` (*ComparerTab.ComparerTab method*), 14

`ComparerTab` (*class in ComparerTab*), 13

`ComparerTab` (*module*), 13

`connect()` (*Database.Database method*), 14

`createDump()` (*ManagerTab.ManagerTab method*), 29

`createKnownPacketTableStatement` (*Database.DatabaseStatements attribute*), 17

`createPacketSetTableStatement` (*Database.DatabaseStatements attribute*), 17

`createPacketTableStatement` (*Database.DatabaseStatements attribute*), 17

`createProject()` (*ManagerTab.ManagerTab method*), 29

`createProjectTableStatement` (*Database.DatabaseStatements attribute*), 17

`createTables()` (*Database.Database method*), 14

`createTableStatementsList` (*Database.DatabaseStatements attribute*), 17

`currentlySendingTabs` (*SenderTab.SenderTab attribute*), 38

## D

`data()` (*PacketTableModel.PacketTableModel method*), 34

`DataAdderThread` (*class in FilterTab*), 20

`Database` (*class in Database*), 14

`Database` (*module*), 14

`DatabaseStatements` (*class in Database*), 17

`dataMask` (*FuzzerTab.FuzzerTab attribute*), 23

`dataMaskChanged()` (*FuzzerTab.FuzzerTab method*), 23

`dataMaxLength` (*FuzzerTab.FuzzerTab attribute*), 23

`db` (*in module Globals*), 24

`DB_NAME` (*in module Settings*), 41

`DB_PATH` (*in module Settings*), 41

`deleteDump()` (*ManagerTab.ManagerTab method*), 29

`deleteFromTableByID()` (*Database.Database method*), 15

`deleteFromTableByValue()` (*Database.Database method*), 15

`deleteKnownPacket()` (*Database.Database method*), 15

`deletePacketSet()` (*Database.Database method*), 15

`deleteProject()` (*ManagerTab.ManagerTab method*), 29

`deleteProjectAndData()` (*Database.Database method*), 15

`detectCANInterfaces()` (*MainTab.MainTab static method*), 27

`disable()` (*ItemAdderThread.ItemAdderThread method*), 25

`disable()` (*SenderThread.FuzzSenderThread method*), 40

`disable()` (*SenderThread.LoopSenderThread method*), 41

`displayUniquePackets()` (*PacketsDialog.PacketsDialog method*), 32

`downwardsSearch` (*SearcherTab.SearcherTab attribute*), 37

`dumpsRowIDs` (*ManagerTab.ManagerTab attribute*), 29

## E

`easterEgg()` (*MainTab.MainTab static method*), 27

`editKnownPacket()` (*ManagerTab.ManagerTab method*), 29

`editProject()` (*ManagerTab.ManagerTab* method), 29  
`emit()` (*Logger.LogHandler* method), 26  
`enterWhenReady()` (*SearcherTab.SearcherTab* method), 37  
`exportProject()` (*ManagerTab.ManagerTab* method), 29

## F

`FDCheckboxChanged()` (*MainTab.MainTab* static method), 27  
`FilterTab` (class in *FilterTab*), 21  
`FilterTab` (module), 20  
`flags()` (*PacketTableModel.PacketTableModel* method), 34  
`FORKME_PATH` (in module *Settings*), 41  
`frameToList()` (*FilterTab.DataAdderThread* method), 20  
`frameToRow()` (*ItemAdderThread.ItemAdderThread* method), 25  
`fromJSON()` (*KnownPacket.KnownPacket* static method), 26  
`fromJSON()` (*Packet.Packet* static method), 31  
`fromJSON()` (*PacketSet.PacketSet* static method), 32  
`fromJSON()` (*Project.Project* static method), 36  
`FuzzerTab` (class in *FuzzerTab*), 23  
`FuzzerTab` (module), 23  
`fuzzingModeChanged()` (*FuzzerTab.FuzzerTab* method), 23  
`fuzzingModeComboBoxValuePairs` (*FuzzerTab.FuzzerTab* attribute), 23  
`FuzzSenderThread` (class in *SenderThread*), 40  
`fuzzSenderThread` (*FuzzerTab.FuzzerTab* attribute), 23

## G

`generateRandomPacket()` (*FuzzerTab.FuzzerTab* method), 23  
`getDeleteByIDStatement()` (*Database.DatabaseStatements* static method), 17  
`getDeleteByValueStatement()` (*Database.DatabaseStatements* static method), 18  
`getDisplayDataLength()` (*Packet.Packet* static method), 31  
`getDump()` (*ManagerTab.ManagerTab* method), 29  
`getInsertKnownPacketStatement()` (*Database.DatabaseStatements* static method), 18  
`getInsertPacketSetStatement()` (*Database.DatabaseStatements* static method), 18

`getInsertPacketStatement()` (*Database.DatabaseStatements* static method), 18  
`getInsertProjectStatement()` (*Database.DatabaseStatements* static method), 19  
`getInsertStatement()` (*Database.DatabaseStatements* static method), 19  
`getKnownPacketDescription()` (*Toolbox.Toolbox* static method), 45  
`getKnownPackets()` (*Database.Database* method), 15  
`getKnownPacketsForCurrentProject()` (*ManagerTab.ManagerTab* method), 29  
`getLogger()` (*Logger.Logger* method), 26  
`getOverallCountStatement()` (*Database.DatabaseStatements* static method), 19  
`getOverallTableCount()` (*Database.Database* method), 15  
`getPacketCount()` (*SnifferTabElement.SnifferTabElement* method), 44  
`getPacketDictIndex()` (*Toolbox.Toolbox* static method), 45  
`getPacketSet()` (*ComparerTab.ComparerTab* method), 14  
`getPacketSets()` (*Database.Database* method), 15  
`getPacketsOfPacketSet()` (*Database.Database* method), 15  
`getProjects()` (*Database.Database* method), 16  
`getSampleData()` (*FilterTab.FilterTab* method), 22  
`getSaveFileName()` (*Toolbox.Toolbox* static method), 45  
`getSelectAllStatement()` (*Database.DatabaseStatements* static method), 19  
`getSelectAllStatementWhereEquals()` (*Database.DatabaseStatements* static method), 19  
`getTabIndex()` (*SenderTabElement.SenderTabElement* method), 39  
`getUniqueIDs()` (*PacketsDialog.PacketsDialog* method), 32  
`getUniquePackets()` (*PacketsDialog.PacketsDialog* method), 32  
`getUniqueRawPackets()` (*PacketsDialog.PacketsDialog* static method), 32  
`getUpdateByIDStatement()` (*Database.DatabaseStatements* static method), 20  
`getValue()` (*PacketTableModel.PacketTableModel* method), 34  
`getWorkingDialog()` (*Toolbox.Toolbox* static

*method*), 45  
GITHUB\_URL (in module *Settings*), 41  
Globals (module), 24

## H

handleCellChanged() (*AbstractTab.AbstractTab method*), 48  
handleCopy() (*AbstractTab.AbstractTab method*), 48  
handleCopy() (*ManagerTab.ManagerTab method*), 30  
handleInterfaceSettingsDialog() (*AbstractTab.AbstractTab method*), 48  
handleInterfaceSettingsDialog() (*SenderTab.SenderTab class method*), 38  
handleInterfaceSettingsDialog() (*SnifferTabElement.SnifferTabElement method*), 44  
handleManageIgnoredPacketsDialog() (*SnifferTabElement.SnifferTabElement method*), 44  
handlePaste() (*AbstractTab.AbstractTab method*), 48  
handlePaste() (*ManagerTab.ManagerTab method*), 30  
handleRightClick() (*AbstractTab.AbstractTab method*), 48  
headerData() (*PacketTableModel.PacketTableModel method*), 34

## I

ICON\_PATH (in module *Settings*), 41  
IDMask (*FuzzerTab.FuzzerTab attribute*), 23  
IDMaskChanged() (*FuzzerTab.FuzzerTab method*), 23  
IDMaxValue (*FuzzerTab.FuzzerTab attribute*), 23  
importProject() (*ManagerTab.ManagerTab method*), 30  
indexInMainTabBar (*SenderTab.SenderTab attribute*), 38  
indexInMainTabBar (*SnifferTab.SnifferTab attribute*), 42  
insertRow() (*PacketTableModel.PacketTableModel method*), 35  
interfaceDialogWidget (*Toolbox.Toolbox attribute*), 45  
interfaceSettingsDialog() (*Toolbox.Toolbox static method*), 45  
interfaceSettingsDialogCheckBoxChanged() (*Toolbox.Toolbox static method*), 45  
interfaceSettingsDialogComboBoxChanged() (*Toolbox.Toolbox static method*), 45  
interfaceSettingsDialogFDCheckBoxChanged() (*Toolbox.Toolbox static method*), 45  
isHexString() (*Toolbox.Toolbox static method*), 46  
ItemAdderThread (class in *ItemAdderThread*), 25

itemAdderThread (*FuzzerTab.FuzzerTab attribute*), 23  
ItemAdderThread (module), 25

## K

KnownPacket (class in *KnownPacket*), 25  
KnownPacket (module), 25  
knownPackets (in module *Globals*), 24  
knownPacketTableCANIDColName (*Database.DatabaseStatements attribute*), 20  
knownPacketTableDataColName (*Database.DatabaseStatements attribute*), 20  
knownPacketTableDescriptionColName (*Database.DatabaseStatements attribute*), 20  
knownPacketTableName (*Database.DatabaseStatements attribute*), 20

## L

labelInterfaceValue (*SenderTab.SenderTab attribute*), 38  
lastWorkingChunk (*SearcherTab.SearcherTab attribute*), 37  
lengthStringToInt() (*Packet.Packet method*), 31  
loadingData (*ManagerTab.ManagerTab attribute*), 30  
loadKernelModules() (*MainTab.MainTab static method*), 27  
Logger (class in *Logger*), 26  
logger (*MainTab.MainTab attribute*), 27  
Logger (module), 26  
logger (*SenderTab.SenderTab attribute*), 39  
logger (*SnifferTab.SnifferTab attribute*), 42  
logger (*Toolbox.Toolbox attribute*), 46  
loggerName (*AbstractTab.AbstractTab attribute*), 48  
LogHandler (class in *Logger*), 26  
LOGO\_PATH (in module *Settings*), 41  
LoopSenderThread (class in *SenderThread*), 41  
loopSenderThread (*SenderTabElement.SenderTabElement attribute*), 40

## M

MainTab (class in *MainTab*), 27  
MainTab (module), 27  
mainWindow (module), 47  
ManagerTab (class in *ManagerTab*), 29  
ManagerTab (module), 29  
manualAddPacket() (*AbstractTab.AbstractTab method*), 48  
minLogLevel (*Logger.Logger attribute*), 26  
mp3Processes (*Toolbox.Toolbox attribute*), 46



## N

noiseData (*FilterTab.FilterTab* attribute), 22

## O

open () (*PacketsDialog.PacketsDialog* method), 32

outputRemainingPacket ()  
(*SearcherTab.SearcherTab* method), 37

outputRemainingPackets () (*FilterTab.FilterTab*  
method), 22

outputRemainingPackets ()  
(*SearcherTab.SearcherTab* method), 37

## P

Packet (class in *Packet*), 31

Packet (module), 31

packetBuildErrorCount (*FuzzerTab.FuzzerTab*  
attribute), 23

PacketsDialog (class in *PacketsDialog*), 31

PacketsDialog (module), 31

PacketSet (class in *PacketSet*), 32

PacketSet (module), 32

packetSetTableName  
(*Database.DatabaseStatements* attribute),  
20

packetTableCANIDColName  
(*Database.DatabaseStatements* attribute),  
20

packetTableDataColName  
(*Database.DatabaseStatements* attribute),  
20

packetTableModel (*AbstractTab.AbstractTab* at-  
tribute), 48

PacketTableModel (class in *PacketTableModel*), 33

PacketTableModel (module), 33

packetTableName (*Database.DatabaseStatements*  
attribute), 20

playMP3 () (*Toolbox.Toolbox* static method), 46

populateInterfaceComboBox () (*Tool-  
box.Toolbox* static method), 46

populateKnownPacketEditLineEdits ()  
(*ManagerTab.ManagerTab* method), 30

populateKnownPackets () (*Man-  
agerTab.ManagerTab* method), 30

populatePacketSets () (*ManagerTab.ManagerTab*  
method), 30

populateProjectEditLineEdits () (*Man-  
agerTab.ManagerTab* method), 30

populateProjects () (*MainTab.MainTab* static  
method), 27

populateProjects () (*ManagerTab.ManagerTab*  
method), 30

prepareUI () (*AboutTab>AboutTab* static method), 13

prepareUI () (*AbstractTab.AbstractTab* method), 48

prepareUI () (*FuzzerTab.FuzzerTab* method), 24

prepareUI () (*MainTab.MainTab* static method), 27

prepareUI () (*ManagerTab.ManagerTab* method), 30

prepareUI () (*PacketsDialog.PacketsDialog* method),  
32

prepareUI () (*SenderTab.SenderTab* static method),  
39

prepareUI () (*SenderTabElement.SenderTabElement*  
method), 40

prepareUI () (*SnifferTab.SnifferTab* static method), 42

preselectUseBitrateCheckBox ()  
(*MainTab.MainTab* static method), 28

Project (class in *Project*), 36

project (in module *Globals*), 25

Project (module), 36

projectTableDescriptionColName  
(*Database.DatabaseStatements* attribute),  
20

projectTableName (*Database.DatabaseStatements*  
attribute), 20

projectTableNameColName  
(*Database.DatabaseStatements* attribute),  
20

## R

rawData (*AbstractTab.AbstractTab* attribute), 48

readOnlyCols (*AbstractTab.AbstractTab* attribute),  
48

removeApplicationStatus () (*MainTab.MainTab*  
static method), 28

removeKnownPacket () (*ManagerTab.ManagerTab*  
method), 30

removeRow () (*PacketTableModel.PacketTableModel*  
method), 35

removeRows () (*PacketTableModel.PacketTableModel*  
method), 35

removeSelectedPackets () (*Abstract-  
Tab.AbstractTab* method), 48

removeSelectedPackets () (*Man-  
agerTab.ManagerTab* method), 30

removeSender () (*SenderTab.SenderTab* static  
method), 39

removeSender () (*SenderTabEle-  
ment.SenderTabElement* method), 40

removeSniffer () (*SnifferTab.SnifferTab* static  
method), 42

removeSniffer () (*SnifferTabEle-  
ment.SnifferTabElement* method), 44

removeVCANInterface () (*MainTab.MainTab* static  
method), 28

rowCount () (*PacketTableModel.PacketTableModel*  
method), 35

run () (*FilterTab.DataAdderThread* method), 21

run () (*ItemAdderThread.ItemAdderThread* method), 25

run () (*SenderThread.FuzzSenderThread* method), 40

`run()` (*SenderThread.LoopSenderThread method*), 41

`run()` (*SnifferProcess.SnifferProcess method*), 42

## S

`saveKnownPacket()` (*Database.Database method*), 16

`savePacket()` (*Database.Database method*), 16

`savePacketsBatch()` (*Database.Database method*), 16

`savePacketSet()` (*Database.Database method*), 16

`savePacketSetWithData()` (*Database.Database method*), 16

`saveProject()` (*Database.Database method*), 17

`saveToFile()` (*ManagerTab.ManagerTab method*), 30

`SearcherTab` (*class in SearcherTab*), 36

`SearcherTab` (*module*), 36

`searchPackets()` (*SearcherTab.SearcherTab method*), 37

`sendAll()` (*SenderTabElement.SenderTabElement method*), 40

`sendAndSearch()` (*SearcherTab.SearcherTab method*), 37

`sendButtonList` (*SenderTabElement.SenderTabElement attribute*), 40

`SenderTab` (*class in SenderTab*), 38

`SenderTab` (*module*), 38

`SenderTabElement` (*class in SenderTabElement*), 39

`SenderTabElement` (*module*), 39

`senderTabs` (*SenderTab.SenderTab attribute*), 39

`SenderThread` (*module*), 40

`sendSinglePacket()` (*SenderTab.SenderTab static method*), 39

`setData()` (*PacketTableModel.PacketTableModel method*), 35

`setGlobalInterfaceStatus()` (*MainTab.MainTab static method*), 28

`setInitialCANData()` (*AbstractTab.AbstractTab method*), 49

`setInitialCANData()` (*SenderTab.SenderTab class method*), 39

`setPacketSet1()` (*ComparerTab.ComparerTab method*), 14

`setPacketSet2()` (*ComparerTab.ComparerTab method*), 14

`setProject()` (*MainTab.MainTab static method*), 28

`setProjectStatus()` (*MainTab.MainTab static method*), 28

`setRowCount()` (*PacketTableModel.PacketTableModel method*), 35

`setSendButtonState()` (*SenderTabElement.SenderTabElement method*), 40

`setText()` (*PacketTableModel.PacketTableModel method*), 35

`Settings` (*module*), 41

`setupStatusBar()` (*MainTab.MainTab static method*), 28

`sharedDataAdderEnabledFlag` (*FilterTab.FilterTab attribute*), 22

`sharedSnifferEnabledFlag` (*FilterTab.FilterTab attribute*), 22

`signalSniffedPacket` (*FilterTab.DataAdderThread attribute*), 21

`sliderChanged()` (*FuzzerTab.FuzzerTab method*), 24

`SnifferProcess` (*class in SnifferProcess*), 42

`SnifferProcess` (*module*), 42

`SnifferTab` (*class in SnifferTab*), 42

`SnifferTab` (*module*), 42

`SnifferTabElement` (*class in SnifferTabElement*), 43

`SnifferTabElement` (*module*), 43

`snifferTabs` (*SnifferTab.SnifferTab attribute*), 43

`sort()` (*PacketTableModel.PacketTableModel method*), 35

`splitLists()` (*SearcherTab.SearcherTab method*), 38

`startFilter()` (*FilterTab.FilterTab method*), 22

`startSnifferAndAdder()` (*FilterTab.FilterTab method*), 22

`staticMetaObject` (*FilterTab.DataAdderThread attribute*), 21

`staticMetaObject` (*ItemAdderThread.ItemAdderThread attribute*), 25

`staticMetaObject` (*PacketTableModel.PacketTableModel attribute*), 36

`staticMetaObject` (*SenderThread.FuzzSenderThread attribute*), 41

`staticMetaObject` (*SenderThread.LoopSenderThread attribute*), 41

`statusBarActiveStatuses` (*MainTab.MainTab attribute*), 28

`statusBarApplicationStatus` (*MainTab.MainTab attribute*), 28

`statusBarInterface` (*MainTab.MainTab attribute*), 28

`statusBarProject` (*MainTab.MainTab attribute*), 28

`stopMP3()` (*Toolbox.Toolbox static method*), 46

`stopSending()` (*SenderTabElement.SenderTabElement method*), 40

`stopSnifferAndAdder()` (*FilterTab.FilterTab method*), 22

`Strings` (*module*), 44

## T

`tabIndex()` (*SnifferTabElement.SnifferTabElement method*), 44

`tableCount` (*Database.DatabaseStatements attribute*), 20

`tableExtractAllRowData()` (*Toolbox.Toolbox static method*), 46  
`tableExtractSelectedRowData()` (*Toolbox.Toolbox static method*), 46  
`tabWidget` (*AbstractTab.AbstractTab attribute*), 49  
`terminateThreads()` (*SnifferTabElement.SnifferTabElement method*), 44  
`textBrowserLogs` (*in module Globals*), 25  
`toComboBoxString()` (*KnownPacket.KnownPacket method*), 26  
`toComboBoxString()` (*PacketSet.PacketSet method*), 33  
`toComboBoxString()` (*Project.Project method*), 36  
`toggleActive()` (*SenderTab.SenderTab static method*), 39  
`toggleActive()` (*SnifferTab.SnifferTab static method*), 43  
`toggleActive()` (*SnifferTabElement.SnifferTabElement method*), 44  
`toggleDisabledProjectGUIElements()` (*Toolbox.Toolbox static method*), 46  
`toggleDisabledSenderGUIElements()` (*Toolbox.Toolbox static method*), 46  
`toggleFuzzing()` (*FuzzerTab.FuzzerTab method*), 24  
`toggleGUIElements()` (*AbstractTab.AbstractTab method*), 49  
`toggleGUIElements()` (*FilterTab.FilterTab method*), 22  
`toggleGUIElements()` (*FuzzerTab.FuzzerTab method*), 24  
`toggleGUIElements()` (*SearcherTab.SearcherTab method*), 38  
`toggleGUIElements()` (*SenderTab.SenderTab static method*), 39  
`toggleGUIElements()` (*SenderTabElement.SenderTabElement method*), 40  
`toggleLoopActive()` (*FuzzerTab.FuzzerTab method*), 24  
`toggleLoopActive()` (*SenderTabElement.SenderTabElement method*), 40  
`toggleSniffing()` (*SnifferTabElement.SnifferTabElement method*), 44  
`toJSON()` (*KnownPacket.KnownPacket method*), 26  
`toJSON()` (*Packet.Packet method*), 31  
`toJSON()` (*PacketSet.PacketSet method*), 33  
`toJSON()` (*Project.Project method*), 36  
`Toolbox` (*class in Toolbox*), 44  
`Toolbox` (*module*), 44  
`updateCANDataInstance()` (*SenderTab.SenderTab class method*), 39  
`updateCANDataInstances()` (*Toolbox.Toolbox static method*), 46  
`updateDump()` (*ManagerTab.ManagerTab method*), 30  
`updateInterfaceLabel()` (*AbstractTab.AbstractTab method*), 49  
`updateInterfaceLabel()` (*SenderTab.SenderTab class method*), 39  
`updateInterfaceLabel()` (*SnifferTab.SnifferTab class method*), 43  
`updateInterfaceLabels()` (*Toolbox.Toolbox static method*), 46  
`updateKnownPacket()` (*Database.Database method*), 17  
`updateNoiseCollectProgress()` (*FilterTab.FilterTab method*), 22  
`updatePackets()` (*Database.Database method*), 17  
`updateProject()` (*Database.Database method*), 17  
`updateStatusBar()` (*SenderTabElement.SenderTabElement method*), 40  
`updateStatusBar()` (*SnifferTabElement.SnifferTabElement method*), 44  
`updateVCANButtons()` (*MainTab.MainTab static method*), 28

## V

`validateDataMaskInput()` (*FuzzerTab.FuzzerTab method*), 24  
`validateIDMaskInput()` (*FuzzerTab.FuzzerTab method*), 24  
`VCANCheckboxChanged()` (*MainTab.MainTab static method*), 27

## W

`widgetFromUIFile()` (*Toolbox.Toolbox static method*), 46

## Y

`yesNoBox()` (*Toolbox.Toolbox static method*), 47

## U

`ui` (*in module Globals*), 25  
`updateCANDataInstance()` (*AbstractTab.AbstractTab method*), 49