VDAT: Virus Data Analysis Tool Documentation

Release 0.9.0.post0

HETDEX Collaboration.

Jun 20, 2018

Contents

Version: 0.9.0.post0

The Virus Data Analysis Tool (VDAT) software is a GUI tool to reduce data from Hobby-Eberly Telescope (HET) observations. The software is being created specifically for HET Dark Energy Experiment (HETDEX), but we try to design it to be easily extended to other user cases.

`vdat` supports python 2.7, 3.4 or later.

This documentation is also available on vdat.readthedocs.io.

CHAPTER 1

User documentation

1.1 Installation

1.1.1 Instructions

The recommended way

The recommended way to install VDAT is using pip:

```
pip install --extra-index-url https://gate.mpe.mpg.de/pypi/simple/ vdat
```

It's possible to set the extra index URL permanently by adding the following lines to the `$HOME/.pip/pip.conf` file:

```
[global]
extra-index-url = https://gate.mpe.mpg.de/pypi/simple
```

or exporting the environment variable:

```
export PIP_EXTRA_INDEX_URL=https://gate.mpe.mpg.de/pypi/simple
```

The list of released versions can be seen on the MPE pypi server. A specific version can be installed using specifiers, e.g. issuing `pip install vdat==0.4`.

`pip` will take care of installing all the *VDAT dependances*, except for the `Qt` bindings. Instructions about the `Qt` binding and how to install them can be found in *Qt bindings*. Also, if you want to run commands in VDAT you will also need to install *Cure*.

If the `pyds9` package, VDAT will be able to send fits files to DS9. See *pyds9* for instructions on how to install the package

We suggest you install VDAT into a virtualenv, in an anaconda/conda or in similar environments.

Of course it is also possible to install VDAT without any of the above with:

```
pip install --user --extra-index-url https://gate.mpe.mpg.de/pypi/simple/ vdat
```

This way the VDAT executables are installed in `$HOME/.local/bin`, so make sure to add this to the environment variable `PATH` to be able to easily use them on the command line. The use of `sudo` when installing with pip is discouraged and potentially harmful.

From the online svn repository

These steps are to be followed if you want to install the latest version.

First get a local copy of `VDAT`, you can checkout the repository with:

```
svn checkout svn://luna.mpe.mpg.de/vdat/trunk vdat
```

Similarly you can check out any branch. Now you can install with:

```
pip install /path/to/vdat
```

or:

```
cd /path/to/vdat
pip install .
```

It's also possible to install vdat directly from the svn repository without checking it out:

```
pip install svn+svn://luna.mpe.mpg.de/vdat/trunk#egg=vdat
```

If necessary replace `trunk` with the desired tag or branch to checkout and install.

1.1.2 Dependances

Mandatory dependences

```
pyhetdex>=0.12
astropy
colorama
numpy
six
peewee
ginga
pyyaml
qimage2ndarray>=1.5.1
qtpy >= 1.1
```

`Qt` bindings

Starting with version 0.9.0, VDAT makes use of the qtpy abstraction layer to communicate with the Qt instead of using directly some of the available python bindings. This allows the user to decide which version of `Qt` to use and which binding you use. `qtpy` automatically detects which `Qt` binding to use and defaults to `PyQt5` if multiple are present. It is possible to select a specific binding using the `QT_API` environment variable; see here for more details.

Available bindings and current status:

- PyQt4, `Qt4` bindings: fully supported. Unfortunately `PyQt4` cannot be installed using pip; also SIP library one of his dependences, cannot be installed with pip for python < 3.5. To our knowledge it is possible to obtain `PyQt4` in one of the following ways:
  - If you feel brave you can try to install them by hand following e.g. these instructions
  - use a package manager; most, if not all, Linux distributions, as well as Mac packet managers like `homebrew` and `macports`, provide `PyQt4`
  - use tools like the anaconda distribution or the conda package manager
- PyQt5, `Qt5` bindings: fully supported. `PyQt5` provides wheels for 32- and 64-bit Windows, 64-bit macOS and 64-bit Linux under python v3.5 or later. Thus it is possible to install `Qt5` bindings using:

  ```
  pip install pyqt5
  ```

  `PyQt5` is also available in `anaconda`/`conda`
- PySide, `Qt4` bindings: not supported. We were not able to run unit test for `PySide` because of problems with the underlying C/C++ library (see issue #2607). Attempts to run VDAT directly with `PySide` cause errors (issue #2598 and issue #2606). `PySide` can be installed using `pip` (see *A note on PySide* if there are problems).
- PySide2, `Qt5` bindings: not supported. It is not yet supported by some dependences (issue #2597). `PySide2` can be installed using pip:

  ```
  pip install --index-url=http://download.qt.io/snapshots/ci/pyside/5.11/latest/␣
  ↪pyside2 --trusted-host download.qt.io
  ```

Optional dependences

`pyds9`

`pyds9` allows to communicate between python and DS9 and is use to send fits and region files from VDAT to DS9. If the package is not installed, VDAT works.

You can install `pyds9` together with the rest of VDAT with:

```
pip install vdat[ds9]
```

or alone with:

```
pip install pyds9
```

`ltl` and `cure`

`cure` is a C++ code that provides utilities that are executed in subprocesses by most of the VDAT commands and depends on `ltl`. However VDAT doesn't depend on `cure`.

Detailed information on how to install `ltl` and `cure` can be found here. Here we report the basic steps for completeness.

Check out and compile `ltl`:

```
cd /path/to/project
svn checkout svn://luna.mpe.mpg.de/ltl/trunk ltl
cd ltl
```

(continues on next page)

(continued from previous page)

```
./bootstrap
./configure
make
cd ..
```

Check out and compile `cure`:

```
svn checkout svn://luna.mpe.mpg.de/cure/trunk cure
cd cure
make install
```

This will install the binaries into `bin` directory. If you don't have OpenGL libraries installed, run:

```
make install HAVE_OPENGL=no
```

as last step. You can use `make -j N` to speed up the build process using N cores.

After it is installed, you might want to:

```
export CUREBIN=/path/to/project/cure/bin
```

Python dependences

- testing:

```
pytest>=3.3
pytest-qt>=2.0
pytest-cov
pytest-xdist
coverage

tox  # for automatizing the tests
```

- documentation:

```
sphinx
numpydoc
alabaster
pyhetdex
```

- automatic documentation build:

```
sphinx-autobuild => 0.5.2
```

1.1.3 Development

If you develop `pyhetdex` and/or `vdat` we suggest you checkout both svn repositories and install them in "editable" mode . We also recommend installing all of the optional dependances:

```
cd /path/to/vdat
pip install -e .
```

It is possible to also install the `pyhetdex` software in editable mode adding `-e` in the `requirements.txt` file before the url or path, e.g.:

```
~> cat requirements.txt
-e /path/to/pyhetdex
```

and adding the options `-r requirements.txt` in the `pip` call.

See *Contribute to VDAT* for more information.

1.1.4 Notes and problems

- It is possible to change the version to install from svn by selecting a specific commit:

  ```
  pip install svn+svn://luna.mpe.mpg.de/vdat/trunk@5#egg=vhc
  ```

  or a different branch/tag:

  ```
  pip install svn+svn://luna.mpe.mpg.de/vdat/tag/v0.0.0#egg=vhc
  ```

- If the installation gets interrupted with an error like:

  ```
  ImportError: No module named 'numpy'
  ```

  run `pip install numpy` and then retry `vdat` installation

1.2 Use VDAT: how to

This document describes how to start using VDAT and introduces the configuration files needed to run the software.

Note: All the executables shipped with VDAT accept the `-h/--help` command line option to show the help text. We suggest our users read them to get acquainted with the available options. Also, this documentation might not directly cover all of them.

1.2.1 Get the configuration files

VDAT needs a handful of configuration files to run. A set of essential files are shipped together with the code and can be recovered with the `vdat_config` executable. Issuing:

```
vdat_config copy
```

copies the following files in the current directory:

```
├── extra_files
│   ├── defaultL.dist
│   ├── defaultR.dist
│   ├── dither_positions.txt
│   ├── IFUcen_HETDEX.txt
│   ├── lines_L.par
│   └── lines_R.par
├── fplane.txt
├── tasks.yml
├── vdat_commands.yml
└── vdat_setting.cfg
```

If any of the files already exists, the user is asked whether she/he wants to overwrite existing files. When issuing the `-f/--force` keyword existing files are overwritten without asking. Alternatively, it is possible to backup existing files before copying with the `-b/--backup` option: this will rename the files adding `.bkp` to the end of their names.

The files are:

- `vdat_setting.cfg`: this is is the main configuration file that contains all of the relevant options to make vdat work. The options in the file are commented and the relevant parts of the file are described in some section of this documentation. VDAT expects to find this file in the directory from which `vdat` is launched; alternatively it can be given with the `-s/--setting` command-line argument. This file is versioned.
- `tasks.yml`: contains the instructions used to build the central panel in the GUI. The entries in the file are described in *Main Panel*. Alternative or multiple file names can be provided either through the configuration option `tasks_config` in the `[general]` section or via the `-t/--tasks-config` option of the `vdat` executable. The order in which options are provided is important: sections in a file might be overridden by sections with the same name in subsequent files. This file is versioned.
- `vdat_commands.yml`: contains the definition of the commands that can be run; see *The Command Interpreter Tool* for more info. Alternative or multiple file names can be provided either through the configuration option `command_config` in the `[general]` section or via the `-c/--command-config` option of the `vdat` executable. The order in which options are provided is important: sections in a file might be overridden by sections with the same name in subsequent files. This file is versioned.
- `fplane.txt`: it tells the GUI where the IFUs are located in the focal plane and connects the various IDs together. The name of the focal plane is passed to VDAT via the `fp_filename` option in the `[fplane]` section of the main configuration files.
- `extra_files/default*.dist`: reference distortion files used by the quick reconstruction. Their names is passed to VDAT via the `default_dist_r` and `default_dist_l` options of the `[reconstruction]` section.
- `extra_files/dither_positions.txt`: list of dither positions per slot position. Used in the `mkdither` command in `vdat_commands.yml`.
- `extra_files/IFUcen_HETDEX.txt`: reference IFU-center files, used both in the quick reconstruction and in the definition of various commands, like `detect`. In the former case, the name is passed to VDAT via the `ifucen` option in the `[general]` section.
- `extra_files/lines_*.par`: reference line files used in the `deformer` command.

1.2.2 Check the configuration files

As the code evolves, so do the configuration files. Some of the changes are essential for the VDAT inner working while some are minor modifications and improvements, we add to the three main configuration files, `vdat_setting.cfg`, `vdat_commands.yml` and `tasks.yml`, a version number. Also to help users track changes and decide how to deal with them, the `vdat_config` command has also `compare` subcommand, that does some basic comparison between the files shipped with VDAT and the ones of the users.

By default the command checks that all the files exist and that the version numbers match. Optionally it is possible to see the line diffs (`-d/--diff`) or to try to load the user files to check for inconsistencies (`-l/--load`). By default those two checks apply only to the *`files`* essential to run VDAT. It is possible to run them on all the files with the `-a/--all` switch.

1.2.3 Launch VDAT

Once you have all the configuration in place, you can run VDAT using the `vdat` executable.

However VDAT needs to know where the raw files are. To do so you can point `vdat` to one or more directories with the `rawdir` option in the `[general]` section of `vdat_setting.cfg` file or with the positional arguments of `vdat`. The directory structure expected is described in *Raw directory structure*. It is also possible to point `vdat` to one or more nights directly: to do so it is also necessary to set the `is_rawdir_night` option in `[general]` section or the `-N/--is-rawdir-night` command line option to `yes`.

At first vdat executes the symlinking as described in *The directory structure and symlinking*, then starts the GUI, the various elements of which are described in *The Graphical User Interface*.

Within the GUI buttons are available that execute tasks in the background, either in single- or multi-processor mode. Multiprocessing and the number of processors can be set using the `use_multiprocessing` and `n_processors` options in the `[general]` section of the `vdat_setting.cfg` file or via the `-m/--use-multiprocessing` and `-n/--n-processors` command line options. We suggest using, at maximum, two fewer processors than available. This way one ensures leaving enough power for the GUI event loop and the secondary threads that we run for some intensive tasks, e.g. when creating thumbnails.

1.2.4 The VDAT database

When symlinking, VDAT in each reduction directory two metadata files, *`SHOT_FILE`* and *`EXPS_FILE`*. They contain information about the type of files in the directory and where they originally came from, as well as allow mapping file names to exposures.

If found, these files are loaded into VDAT and used to rebuild the internal database, without need to redo the symlinking.

Given the importance of these files, we provide an executable to check and if possible repair the metadata files: `vdat_db`. It has two subcommands:

1. `check`: it checks that all the metadata file found have the correct version and can be correctly loaded into the database.
2. `update`: it tries to update the metadata files to the current version and to load them to the database. Before beginning the checks, the old metadata are moved into a backup file and if the update fails they are not moved back. This way the offending directories do not make `vdat` fail on the next run

Success or errors are reported to the console either in a condensed way. More details can be obtained using the `-v\--verbose` option.

1.3 The directory structure and symlinking

The first thing that VDAT performs when launched is to symlink the files from the `raw` directory to the `redux` directory, changing the way the files are organised. First we describe how the files are (should be) organized in the *raw directories* and how VDAT reorganises them in the *redux directories*. Then we will describe how the *symlinking* works and how the user can interact with it.

1.3.1 Raw directory structure

The path to the raw directory can be provided via the configuration option `rawdir` in the `[general]` section of the main VDAT configuration file or via the `vdat` positional command line arguments.

VDAT assumes that the raw directory has the following structure and that `rawdir` points to the directory just containing the dates:

```
├── 20120301                                                # Date of the start of␣
↪the night
│   ├── virus                                               # instrument name
│   │   ├── virus0003000                                    # shot ID
│   │   │   ├── exp01                                       # science dithers 1/3
│   │   │   │   └── virus
│   │   │   │       ├── 20120301T002848_014LL_sci.fits
│   │   │   │       ├── [...]
│   │   │   │       └── 20120301T002848_065RU_sci.fits
│   │   │   ├── [...]
│   │   │   └── exp03                                       # science dithers 3/3
│   │   │       └── virus
│   │   │           ├── 20120301T004248_014LL_sci.fits
│   │   │           ├── [...]
│   │   │           └── 20120301T004248_065RU_sci.fits
│   │   ├── virus0003001                                    # shot ID
│   │   │   ├── exp01                                       # bias frames
│   │   │   │   └── virus
│   │   │   │       ├── 20120301T000000_014LL_zro.fits
│   │   │   │       ├── [...]
│   │   │   │       └── 20120301T000000_065RU_zro.fits
│   │   │   ├── [...]
│   │   │   └── expNN                                       # NN exposures
│   │   │       └── virus
│   │   │           ├── 20120301T001000_014LL_zro.fits
│   │   │           ├── [...]
│   │   │           └── 20120301T001000_065RU_zro.fits
│   │   ├── virus0003002                                    # shot ID
│   │   │   ├── exp01                                       # flat frames
│   │   │   │   └── virus
│   │   │   │       ├── 20120301T001200_014LL_flt.fits
│   │   │   │       ├── [...]
│   │   │   │       └── 20120301T001200_065RU_flt.fits
│   │   │   └── expNN                                       # NN exposures
│   │   │       └── virus
│   │   │           ├── 20120301T002000_014LL_flt.fits
│   │   │           ├── [...]
│   │   │           └── 20120301T002000_065RU_flt.fits
│   │   └── virus0003002                                    # shot ID
│   │       ├── exp01                                       # comp/arc frames
│   │       │   └── virus
│   │       │       ├── 20120301T002200_014LL_cmp.fits
│   │       │       ├── [...]
│   │       │       └── 20120301T002200_065RU_cmp.fits
│   │       ├── [...]
│   │       └── expNN                                       # NN exposures
│   │           └── virus
│   │               ├── 20120301T003000_014LL_cmp.fits
│   │               ├── [...]
│   │               └── 20120301T003000_065RU_cmp.fits
│   ├── gc1                                                 # guide probe data,␣
↪ignored
│   │   └── [...]
│   ├── gc2                                                 # guide probe data,␣
↪ignored
│   │   └── [...]
│   ├── wf1                                                 # wave front sensor,␣
↪ignored
```

(continues on next page)

(continued from previous page)

```
│   │   └── [...]
│   └── wf2                                                    # wave front sensor,␣
↪ignored
│       └── [...]
└── 20120302                                                   # Date of the following␣
↪night
    └── [...]                                                  # as before
```

Early versions of the data might miss the `virus` directory, the instrument name, just below the nights. See *Configuration* for more info.

1.3.2 Redux directory structure

The name of the redux directory is taken from the `redux_dir` option in the `[general]` section of the main VDAT configuration file.

The structure of the redux directory is the following:

```
├── vdat.db
├── 20120301
│   ├── cal
│   │   └── 20151025_122009
│   │       ├── 20120301T001200_014LL_flt.fits
│   │       ├── [...]
│   │       ├── 20120301T001200_065RU_flt.fits
│   │       ├── 20120301T002000_014LL_flt.fits
│   │       ├── [...]
│   │       ├── 20120301T002000_065RU_flt.fits
│   │       ├── 20120301T002200_014LL_cmp.fits
│   │       ├── [...]
│   │       ├── 20120301T002200_065RU_cmp.fits
│   │       ├── 20120301T003000_014LL_cmp.fits
│   │       ├── [...]
│   │       └── 20120301T003000_065RU_cmp.fits
│   ├── sci
│   │   └── M37SIM-00003-obs-1
│   │       ├── 20120301T002848_014LL_sci.fits
│   │       ├── [...]
│   │       ├── 20120301T002848_065RU_sci.fits
│   │       ├── 20120301T004248_014LL_sci.fits
│   │       ├── [...]
│   │       └── 20120301T004248_065RU_sci.fits
│   └── zro
│       └── 20151025_120252
│           ├── 20120301T000000_014LL_zro.fits
│           ├── [...]
│           ├── 20120301T000000_065RU_zro.fits
│           ├── 20120301T001000_014LL_zro.fits
│           ├── [...]
│           └── 20120301T001000_065RU_zro.fits
└── 20120301
```

1.3.3 Symlinking

In this phase the files from *Raw directory structure* are symlinked into *Redux directory structure* and a database, used to drive the GUI, is created.

As in the raw directory, the data are divided per night, but inside each night they are reorganized by their type.

- The science frames belonging to the same virus shot are organized in a subdirectory of the `sci` directory, called as the value of the `OBJECT` header keyword, with white spaces replaced by underscores. If the keyword is not present or holds multiple values for the files belonging to one shot the symlinking will fail. If the `OBJECT` keyword is not populated, the directory name `'no_object'` is used. If multiple shots in the same night have the same `OBJECT`, the directory name is the value of the keyword followed by `"_sym"` and a three digit counter.
- The bias (a.k.a. zero) frames belonging to one shot are grouped together in a subdirectory of `zro`, whose name is the average time stamp of all of the files.
- The calibration frames, namely flats and arcs (a.k.a. comps), are grouped together in pairs in subdirectories of `cal`. The name of each subdirectory is the average time stamp of the flats or arc/comp files it contains. The rationale for the grouping is that the flat and the arcs need to be combined to compute distortion and fiber model.

All the files are symlinked in the above subdirectories without replicating the `exp??/virus` structure.

When symlinking the files, two metadata files are created in each directory: `shot_name.txt` and `exposure_names.txt`. The former one contains information about the type of files and the original shot directories; the latter contains a mapping between exposure number.

If a `redux` directory is found, VDAT uses these files to rebuild the internal database, thus they **should never** be removed without removing the directory too. In this way is possible to re-run VDAT without performing the initial symlink.

After the symlink is finished, VDAT attempts to find a `zro` and a `cal` directory for each `sci` directory. This is used as a source of calibration frames for the reduction. As a default VDAT chooses the closest directories in time, but the `zro` and `cal` directories can also be chosen by the user via the GUI.

Configuration

It's possible for the user to customize some part of the symlinking, e.g. providing new directory name matches, via the main VDAT configuration file, `vdat_setting.cfg`.

In the rest of the section we will describe the name of the available options, divided per section in the file.

`[general]`

- `rawdir`: path to the raw directory/ies or to the night(s) to symlink; it is overridden by the `vdat` positional arguments.
- `redux_dir`: path to the redux directory.
- `is_rawdir_night`: if true `vdat` interprets the `rawdir` entries as being already night directories. If not found defaults to `no`. Can be overridden with the `-N/--is-rawsir-night` command line option.

`[symlink]`

- `night`: generic name of the night directories. If, e.g. the night directory name is a eight digit number, the value would be: `nights = */[0-9]{8}`; ignored if `is_rawdir_night` is given.

- `is_night_regex`: if the value is any of `yes/true/on/1` the value of `night` is interpreted as a regex expression, if it's any of `no/false/off/0` or not present, the `night` is interpreted as described in `fnmatch`.
- `virus_instrument`: name of the instrument; it is the name of the directory between the night and the shot; if not found, defaults to 'virus'. If it's given but empty, the directory structure becomes night/shot. Can be overridden with the `-i/--virus-instrument` command line option.
- `virus_shot`: generic name for the virus shot directories. Very likely will be `virus_shot = *virus[0-9]*`.
- `is_virus_shot_regex`: as `is_night_regex` but for the virus shot entry.
- `virus_fits_files`: name of the fits file to search; e.g.: `*/virus/*.fits`
- `is_virus_fits_files_regex`: as `is_night_regex` but for the virus fits file
- `relative_symlink`: decide whether to do an absolute or a relative symlink
- `replace_symlink`: if a symlink exists, remove and recreate it
- `image_type_pattern`: regex pattern to use to extract the image type from the file name in the form of `pattern, replacement`. For example `.*_(\w{3})\.fits, \1` extract the last three letters before the file
- `datetime_pattern`: regex pattern to use to extract the date and time from the file name in the form of `pattern, replacement`. For example `.*/(\d{8}T\d{6})_.*\.fits, \1` extract eight number, a `T` and other six number at the beginning of the file name
- `datetime_fmt`: format of the date and time extracted wit the above pattern; e.g.: `%Y%m%dT%H%M%S`
- `object_key`: name of the header keyword to use to extract the object for the science frames; default `OBJECT`
- `cal_types`: list of comma separated image types that should be collected together; `cal_types = flt, cmp` collects flat and arc frames, while `cal_types = twi, cmp` collects twilights and arc frames. The option can be overridden with the `--cal-types` command line option.
- `cal_type_header_typ`: when dealing with the calibration type `typ` get the value of the header keyword `cal_type_header_typ` and use it as `object` for that type. This allows the collection multiple shots of the same `typ` in the same calibration directory as long as the header keyword has different values. The `cmp` types have different lamp types, whose names are saved in the `OBJECT` header keyword, so by default VDAT uses `cal_type_header_cmp = OBJECT`
- `max_delta_cal`: a flat and an arc per `lamp_type` shot are collected in the same redux directory if their average time stamps differs less second than:

Unknown or nonstandard types can be also symlinked, as long as the operation can be carried out as if the types were `sci` or `zro`. At the moment there is no support for nonstandard `cal` like types. To declare that an unknown type, e.g. `drk` can be symlinked as if it were a known one, e.g. `zro` add to the `[symlink]` section the following option:

```
drk_symlink_as = zro
```

1.4 The Graphical User Interface

This part of the documentation strives to describe the look and functionality of the various parts that make up VDAT and how to customize them.

1.4.1 The main window

The main window is split into various parts:

- a menu bar, in the upper part of the VDAT window, or in the menu bar at the top of the screen under OSX;
- the largest part of the VDAT window is occupied by two panels that allow navigation through directories and execution of reduction steps;
- the bottom part is occupied by a log panel and a progress bar used to notify the user about the progress and success/failures of the reductions steps.

> **Warning:** the progress bar is currently under development

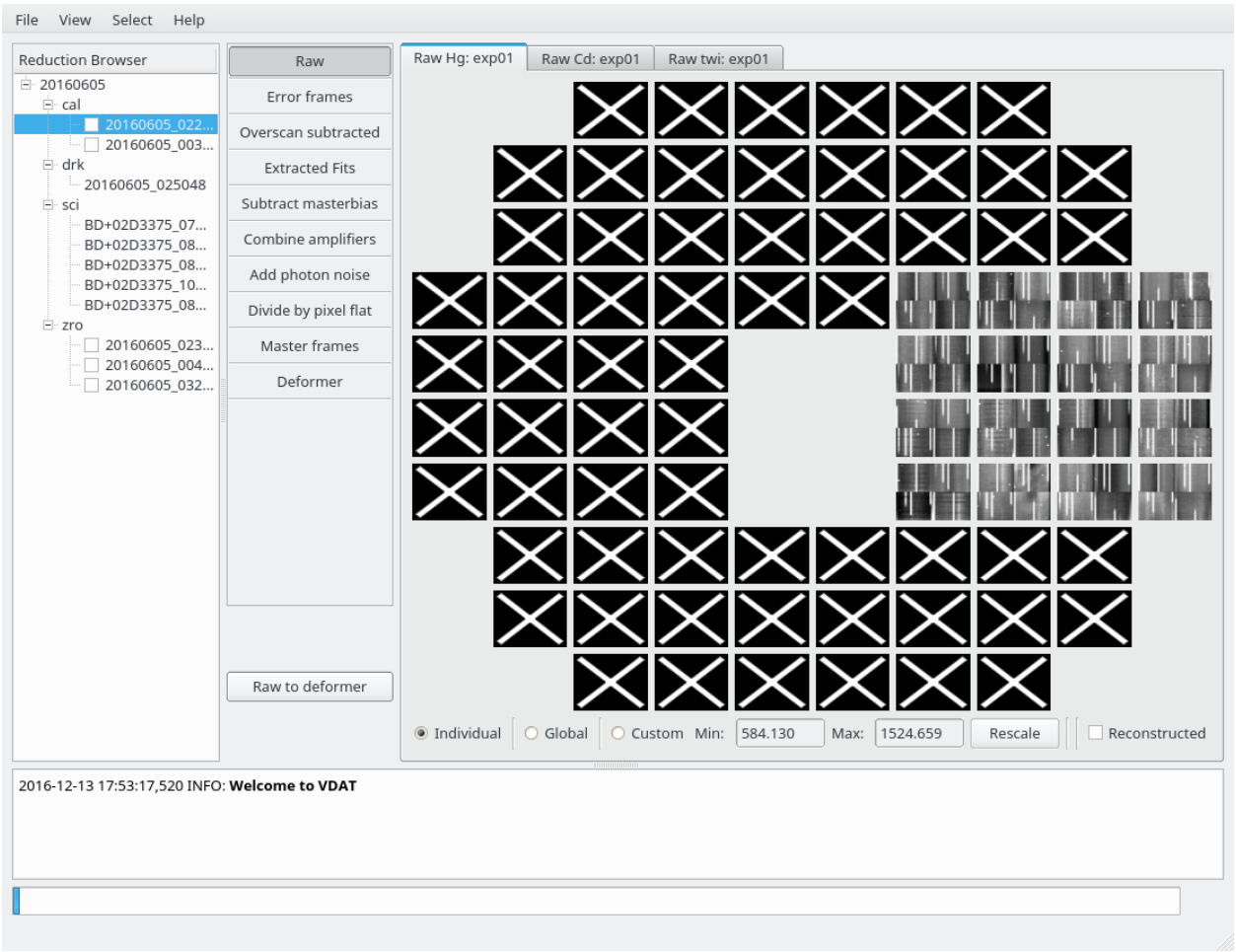


Fig. 1: Screenshot of the main VDAT window, showing all the elements discussed in *The main window*

A more in-depth description of the above elements can be found by following these links:

The menu bar

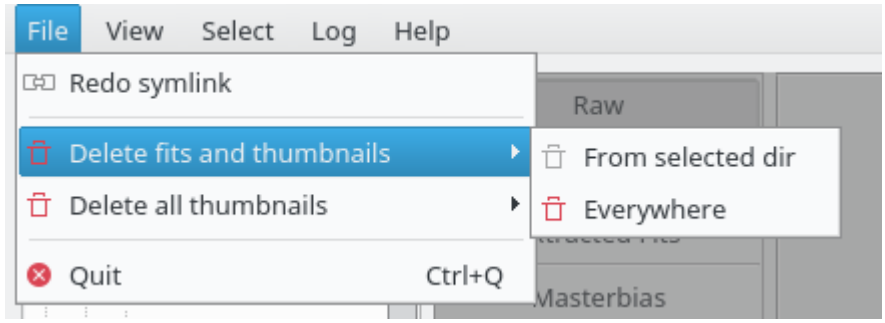
The menu bar, as for every graphical application, allow to perform actions via drop-down menus.

On Mac OSX operation systems, the menu bar can be found in the menu bar at the top of the screen. On all the other operating systems the menu bar is located in the upper part of the VDAT window.

Note: Colors and icons might change depending on the operating system and graphical environment.

`File` menu

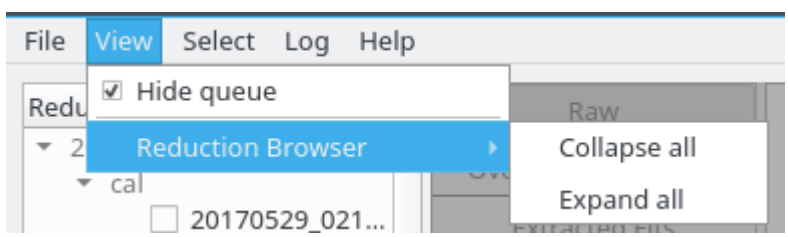
Under this menu there are actions to:



- Redo symlink: if the raw directory/ies provided via the configuration file or the command line arguments exist, this action is enables and it allows to redo the symlinking without restarting VDAT.
- Delete fits and thumbnails: allow to remove all the fits files, the corresponding error files and thumbnails matching the entries from the `intermediate_files` option of the `[general]` section of the main configuration file. It is possible to delete the files from the selected directory or everywhere. The former option is enabled only after selecting one directory.

  It is possible to remove a different set of files for a specific directory type specifying them in the `intermediate_files_typ` option of the `[general]` section. For instance if the type is `cal` the option would be called `intermediate_files_cal`. This works only when removing files from the selected directory.
- Delete all thumbnails: remove all the thumbnails either from the selected directory of everywhere.
- Quit VDAT

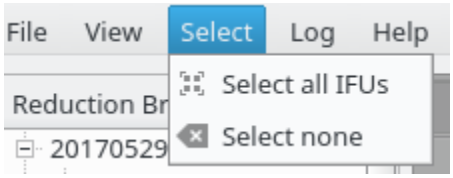
`View` menu



This menu contains two entries:

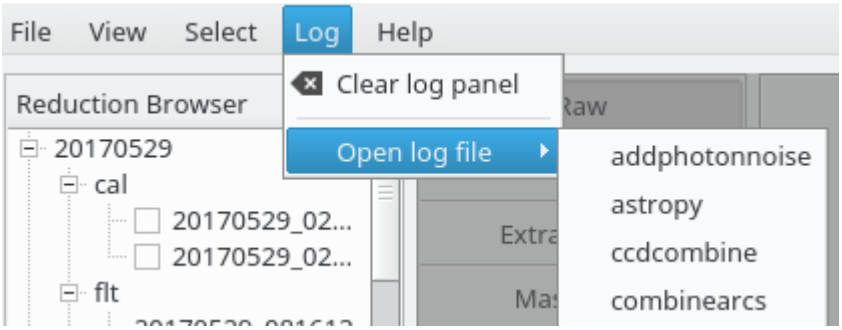
- hide or show the queue window. The queue can also be closed clicking on the corresponding button in the queue window and the change is mirrored into this menu;
- collapse or expand the directory tree shown in the "Reduction Browser".

`Select` menu



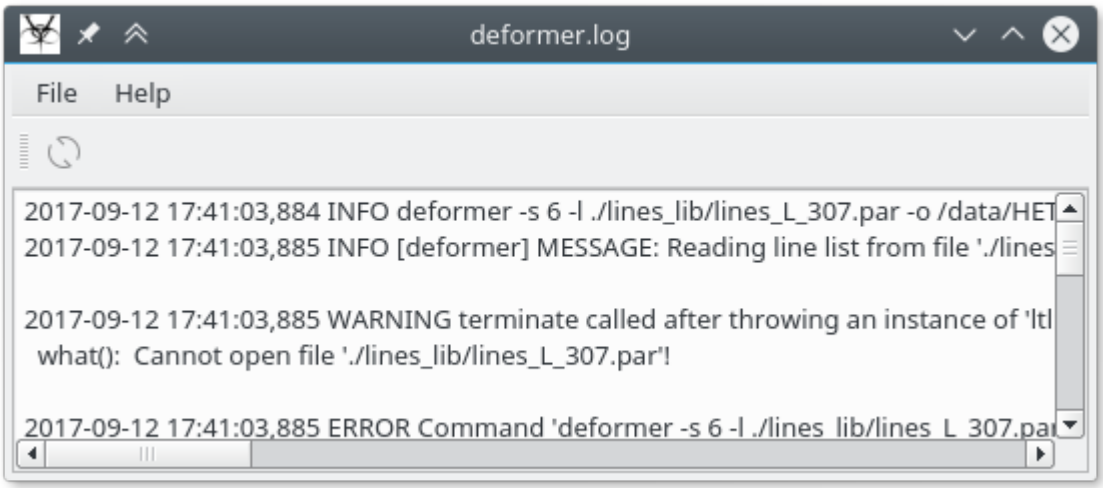
This menu allow to select or deselect all the IFUs at once.

`Log` menu



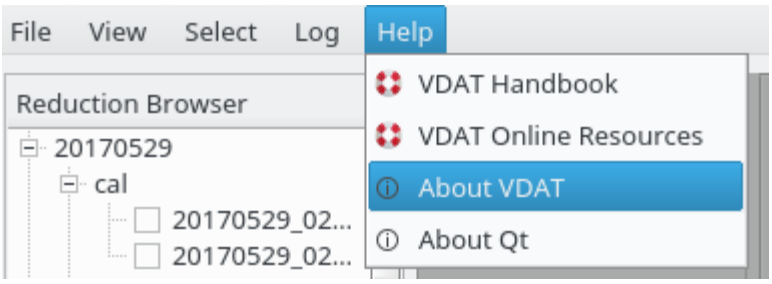
This menu is divided in two parts. The "Clear log panel" button clears the *The logging panel*.

The "Open log file" shows a list of log files: clicking on one of the entries open the corresponding files in a new window, like e.g. in the image below. The window watches the file and if it is modified the "Refresh" action, available in the toolbar and in the "File" menu, is activated and allow to reload the file.



`Help` menu

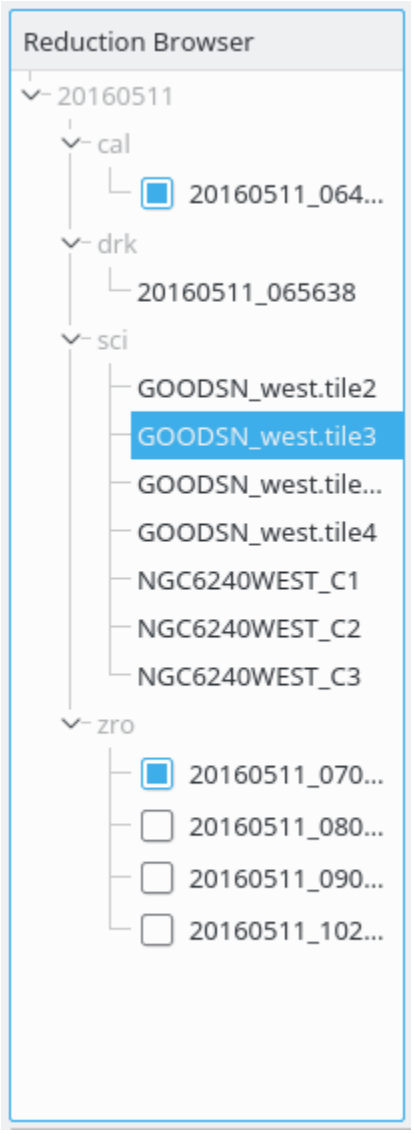
Under this menu there are actions to:



- VDAT Handbook: offline VDAT documentation
- VDAT Online Resources: contains the links to this html documentation
- About VDAT: contains information like the version and the authors
- About Qt: information about the Qt version in use

Reduction browser

The leftmost part of the VDAT window is occupied by the `Reduction browser`. This panel shows the directories that have already been symlinked and reflects the structure described in *Redux directory structure*.



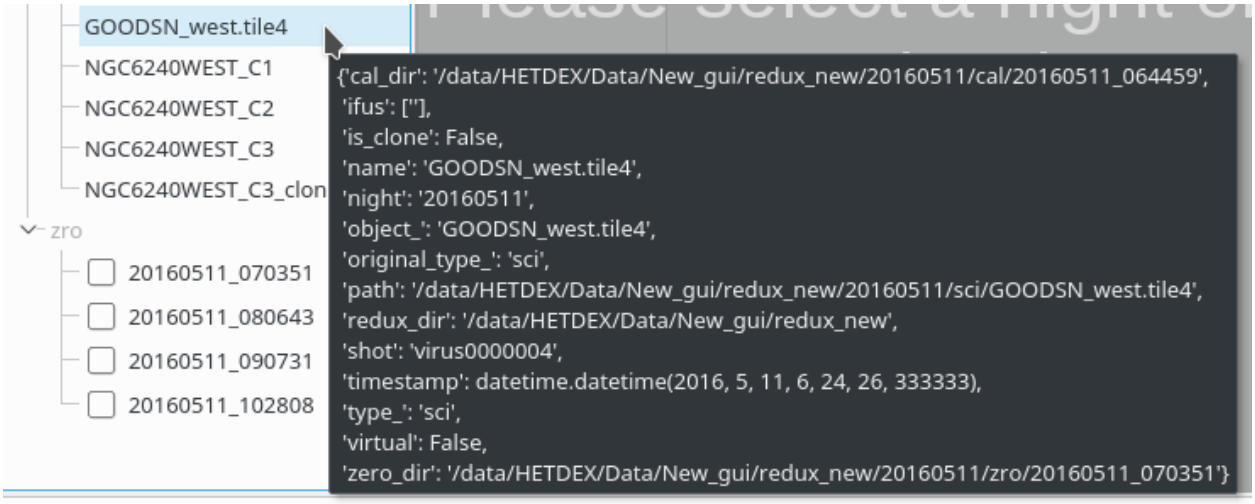
The figure on the right shows what the directory browser looks like. The directories shown with a light colour cannot be selected. Clicking the arrow or double-clicking on the corresponding name will collapse or expand the branch below them.

The directories shown in dark colours can be selected by the user. Upon clicking, the background of the corresponding entry changes colour and the view in the *Main Panel* is updated to show the data stored in the selected directory.

The directories below the `cal` and the `zro` types show a check box on the left of the name. As described at the end of the *Symlinking* section, every directory contains a reference to a `zro` and a `cal` directory. They are passed by default to the command interpreter, as described in *The interpreter*. Using the checkboxes it is possible to override these defaults and use a specific set of `zro` and `cal` directories, even across different nights. Only one directory per type is checkable at any time and, if no directory is checked, the default is used.

If, when first shown, the elements in the reduction browser exceed the visible area, the directory tree is fully collapsed.

During the symlinking we store more information than is shown in the reduction browser. This information is accessible via a tooltip, that is displayed when the mouse hovers for a few moments over a directory name. It appears similar to the one in the following image:

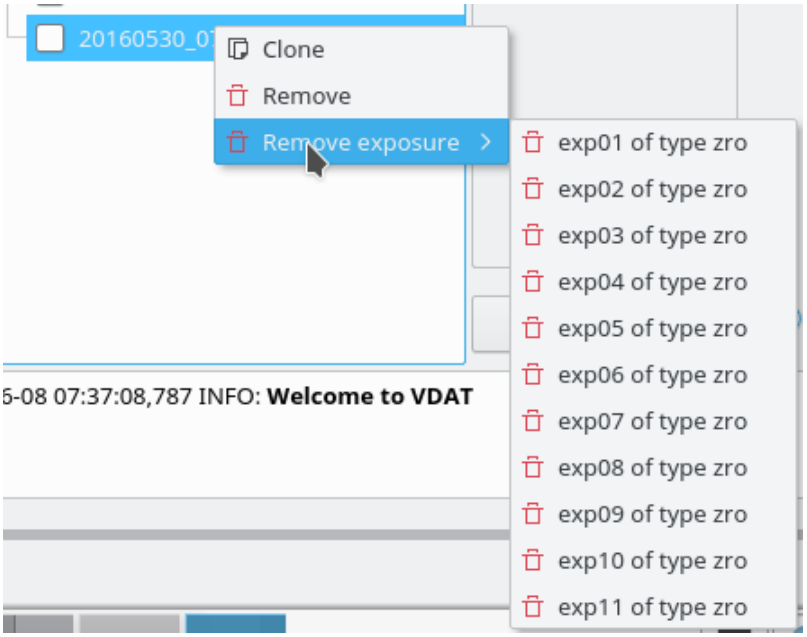


The tooltip will have the following entries:

- `cal_dir`, `zero_dir`: path to the default reference calibration and zero directories, present only if applicable
- `is_clone`: if true, mark the directory as cloned; see below for more details
- `name`: name of the directory
- `night`: night to which it the directory belongs
- `object_`: for `sci` this is the value of the `OBJECT` header keyword; for `cal` it should be a combination of the flat and the name of the arc lamps; for `zro` it is the same as `type_`; for other types it depends on how they have been symlinked.
- `original_type_`: file type as encoded in the fits file names
- `path`: full path to the directory
- `redux_dir`: full path to the redux directory
- `shot`: name of the shot or shots that end up in the directory
- `timestamp`: average time for all the raw files in the directory
- `type_`: name of the type as it appears in the reduction browser

> **Warning:** the following entries will be removed in future releases:
>
> - `ifus`
> - `virtual`
>
> The following entries might be modified in future releases:
>
> - `is_clone`

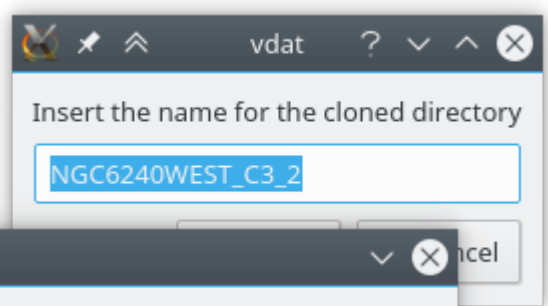
When planning VDAT, we envisioned the possibility for people to branch reduction steps for testing purposes. Therefore we added the option to copy or remove a directory.

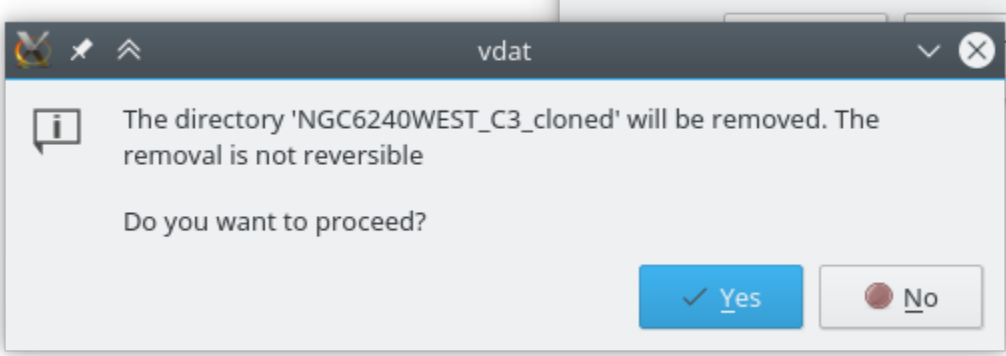


To do it select a directory and right-click on it; this brings up a menu, like the one on the left, showing one or two entries:

- `Clone`: copy the selected directory and all the files in it; this entry is available for every directory.
- `Remove`: remove the selected directory recursively; available only for directories created with the `clone` command.
- `Remove exposure`: allow to remove one exposure from the selected directory; available only for directories created with the `clone` command.

Upon selecting `Clone`, a new window pops up with a proposed name for the new directory (figure on the right). The user can edit the name and proceed to the creation of the directory with the `Clone` button or cancel it with the `Cancel` button. If the named directory already exists, the window appears again, making it impossible to overwrite or mangle existing directories.





When deciding to remove cloned directories, the user will be asked to confirm the decision in order to avoid unwanted deletions. An example of this can be seen in the

above figure.

Main Panel

This is the central part of the VDAT window. It is divided in two parts:

- a column on the left: it shows the reduction steps as horizontal tabs and, on the lower part, buttons to execute one or more commands;
- a big area on the right: typically this part shows fits files for the directory and the step selected.

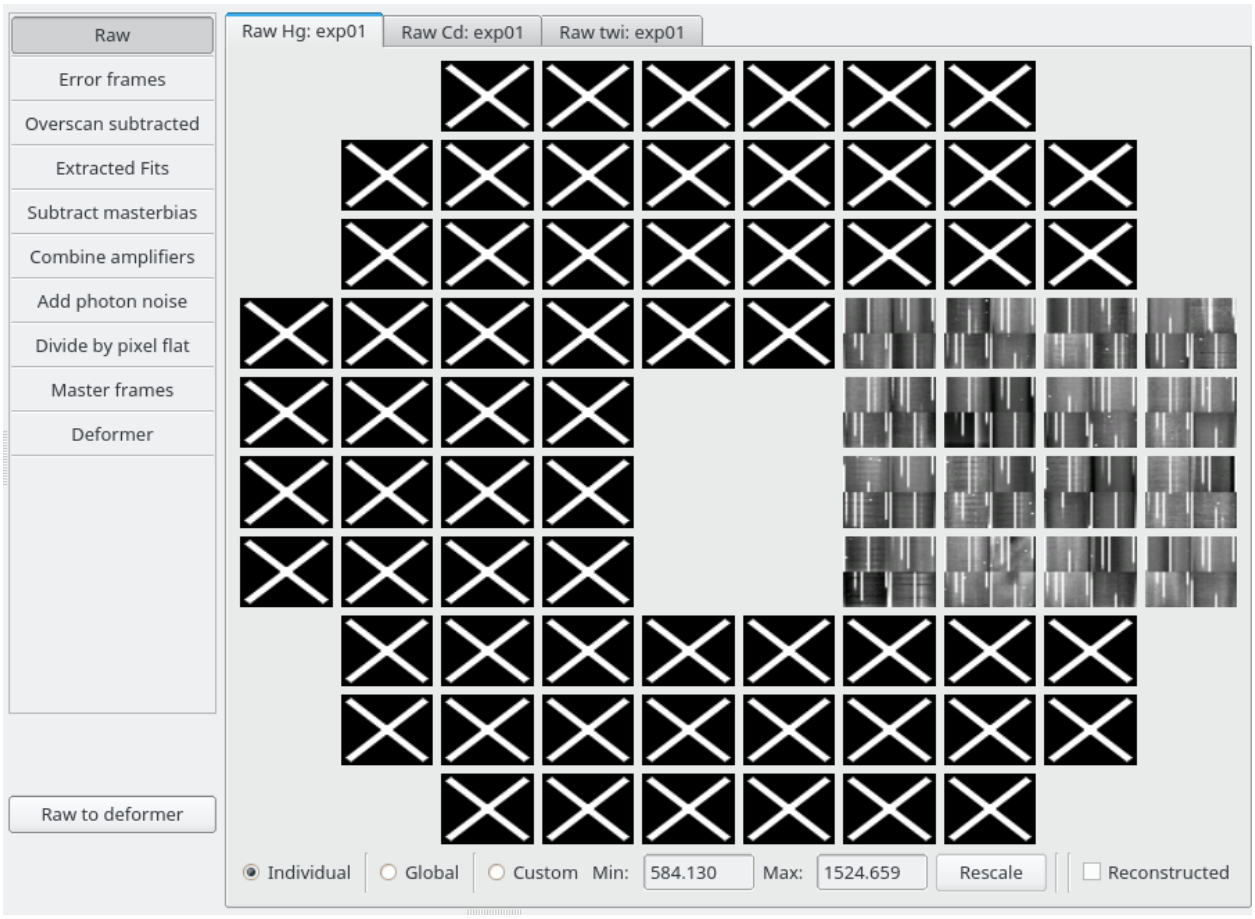


Fig. 2: Screenshot of the central VDAT panel, showing all the elements discussed in this section

Configuration file

The content of the main panel is configurable by the user. The configuration file is written according to the YAML standard. VDAT ships a standard `tasks.yml` file, that can be retrieved using the `vdat_config copy` command. When starting, `vdat` will look for such a file in the current directory. A different file can be provided using the `-t/--tasks-config` command line options.

The configuration file must contain a number of sections, one for each type of files that the user wants to visualize. The name of the sections correspond to the types that are shown in the *Reduction browser*. Each section contains a list

of steps, each one corresponding to one of the horizontal buttons in the upper part of the left panel. The main structure is as in this example:

```
zro:
    # configuration for the zero/bias shots
    - # step 1
      [...]
    - # step n
      [...]
cal:
    # configuration for the flat and arc shots
    - # step 1
      [...]
    - # step n
      [...]
sci:
    # configuation for the science shots
    - # step 1
      [...]
    - # step n
      [...]
```

Each step has three sections, like in this example:

```
- step_name: Raw
  buttons:  # description of the buttons to create
      - # button 1
        [...]
      - # button n
        [...]
  tabs:  # descriptions of the tabs to show
      - # tab 1
        [...]
      - # tab n
        [...]
```

- `step_name` (mandatory): this is the name that appears in the horizontal tab. Typically it represents one step of the reduction process.
- `buttons` (optional): list of buttons to show in the lower part of the left panel. See *The buttons* for further information.
- `tabs` (optional): list of tab types to show in the right panel. See *The tabs* for more info.

The buttons

To each step can be associated one or more button, each of which can run one or more commands. Each command is interpreted and run as described in *The Command Interpreter Tool*.

The buttons are defined as a list and each item is composed by:

- `name` (mandatory): the name that appears in the button.
- `commands` (mandatory): a string or a list of strings. Each string define a command that will pushed into a queue when the button is clicked.
- `tool_tip` (optional): if provided, the text is used as tool tip for the button.

As an example, the following will create two buttons, called `Subtract Overscan` and `Reduce`, associated with one and two commands, respectively. Hovering over the first button the string `Subtract the overscan from the frame` will appear, while the second doesn't show any tooltip.

```
buttons:
  - name: Subtract Overscan
    commands: subtractfits $args -o $biassec $fits
    tool_tip: 'Subtract the overscan from the frame'
  - name: Reduce
    commands:
      - subtractfits $args -o $biassec $fits
      - masterbias $args -o $target_dir/$masterbias_file $fits
```

The tabs

For each step and file type, the user might want to inspect files, logs and have tools to validate or compare the results. To allow this, we reserve the largest portion of the VDAT GUI. In this area multiple tabs can be added by the user via the aforementioned configuration file.

Tabs are defined as a list under the `tabs` section. Each item must have the `tab_type` entry, specifying the type of tab to add, while all the other entries depend on the type of tab. Each tab type can create one or more tabs, according to its specifications. A generic configuration entry looks like the following example:

```
tabs:
    - tab_type: type1
      [all the options for the tab type 1]
    [...]
    - tab_type: typeN
      [all the options for the tab type N]
```

VDAT ships with few tab types, described in *Builtin tab types*. Also it is possible to add extra tabs types using the plugin mechanism described in *New tab types*.

List available tab types

VDAT provides a way to discover which tab types are available on the system, via the `vdat_plugins` executable. The command:

```
vdat_plugins tab_types
```

lists all the available tab types names; the option `-v\--verbose` will show the doc strings for all the tab types. It is also possible to display only one type providing its name in the command line. E.g:

```
vdat_plugins tab_types exp_combined
```

shows only the `exp_combined` type and its doc string (this command ignores the `verbose` settings).

Builtin tab types

In this section are described the builtin types and the arguments required to use them. We will provide more types in the future.

All the builtin tab types, unless otherwise noted, use the focal plane file provided via the `[fplane]` section main vdat configuration file to build the focal plane shown in the tabs. If the input file, given with the option `fp_filename` contains IFUs that you don't want to show, it is possible to list the undesired IFUSLOTs using the `exclude_ifuslot` option. Finally the `respect_empty` option allows to decide whether empty rows and columns in the focal plane file must be shown (`yes`) or can be hidden (`no`). Here is an example of the relevant configuration entries:

```
[fplane]
# Settings related to the focal plane
fp_filename = ${general:config_dir}/fplane.txt
# (optional, default empty) comma separated list of IFUSLOTs that should not be
# shown in the GUI. There is no harm listing IDs even if they are not in the
# fplane file
exclude_ifuslot = 000, 555, 056, 066
# (optional, default: ``yes``) If the options is set to ``no``, the focal plane
# uses the maximum available space and all the empty columns and rows are not
# shown. If ``yes``, shows empty internal rows and columns, but not outside the
# range of existing IFUs.
# Example: the focal plane has the following two IFUSLOT: 073 and 095.
# * if ``no``: the two IFUs are shown at the upper left and lower right corners
#   of a 2x2 square, i.e. the empty column and row between them is not shown
# * if ``yes``: the two IFUs are shown at the upper left and lower right
#   corners of a 3x3 square
respect_empty = yes
```

Note: the same focal plane file is used internally by the VDAT GUI to register the selected IFUs.

`exp_fits` and `exp_combined`

Each shot consists of a number of exposures. Also multiple calibration shots might be combined into a unique directory by VDAT. These tab types allow to show each exposure in one tab. The difference between the two types is that the `exp_fits` type shows the fits files, while the `exp_combined` adds a checkbox that can be used to display to a "quick reconstruction" image.

Both types accept the options described below, together with the formatting names available for building file names and tab titles.

`vdat.gui.tabs.entry_points.`**`exp_fits`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return tabs of type `tab_widget.FitsFplanePanel`. Each tab represent one exposure in one of the types in `target_dir`.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type} {exp}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.

- `header_keys` (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `basename`: date-time part of the file name.
- `col`, `row` (`file_name` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type of the file(s) in the target directory.
- `object`: value of the OBJECT header keyword.
- `exp`: exposure number.

See *`interface.plugin_interface()`* for the signature of this function.

`vdat.gui.tabs.entry_points.`**`exp_combined`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as *`exp_fits()`*, but using tabs of type *`tab_widget.FitsAndReconFplanePanel`*.

As example the following configuration creates the tabs shown in the image below.

```
- &cal_raw_exp_combined
  tab_type: exp_combined
  file_name: '{basename}_{ifuslot}{col}{row}_{orig_type}.fits'
  cols: ['L', 'R']
  rows: ['U', 'L']
  title: '({step}) {object}: {exp}'
  tool_tip: '({step}) {object}, {type}, {orig_type}: {exp}'
  header_keys: ['SPECID', 'IFUID', 'IFUSLOT', 'CCDPOS', 'CCDHALF']
```

The tab types create multiple tabs, one per each exposure in the selected directory. Each of the squares represent one IFU. If one shows a white "X" on black, it means that no file has being found. Otherwise the area is divided in parts, according to the number of `cols` and `rows` in the configuration, and each part is filled with the thumbnail of one file. In the example there are four files per IFU per exposure. Each file is displayed using the zscale scaling algorithm. Since each IFU is represented by a rectangle only a few centimeters in size, we don't display the fits files themselves, but a much smaller thumbnail that is created, updated or removed at need. Users can remove thumbnails from the selected directory or from everywhere using the *File menu*.

If at least one file is shown, the user can change the scaling of the images using the radio buttons below the focal plane. `Global` repaint the thumbnails using the absolute minimum and maximum index from the `zscale` algorithm. The limits are shown in a tool tip that appears when hovering over the button for a few seconds. The `Custom` button allow to set user selected upper and lower limits; the limit submitted via the `Rescale` button are applied on all the IFUs.

The three radio buttons can also be used to repaint the thumbnails. This is useful, e.g., when files has been created or removed.

For the `exp_combined` type, the `Reconstructed` checkbox on the lower right part allows to switch between a view showing the fits files and one showing their combination. This use the `QuickReconstructedIFU` to build an approximated image from the fits files. The image is created only if all the files required exists. The reconstruction algorithm needs some extra information to know where fibers located on the fits files and which part of the sky they see. The information are provided to VDAT via the `vdat_setting.cfg` file:

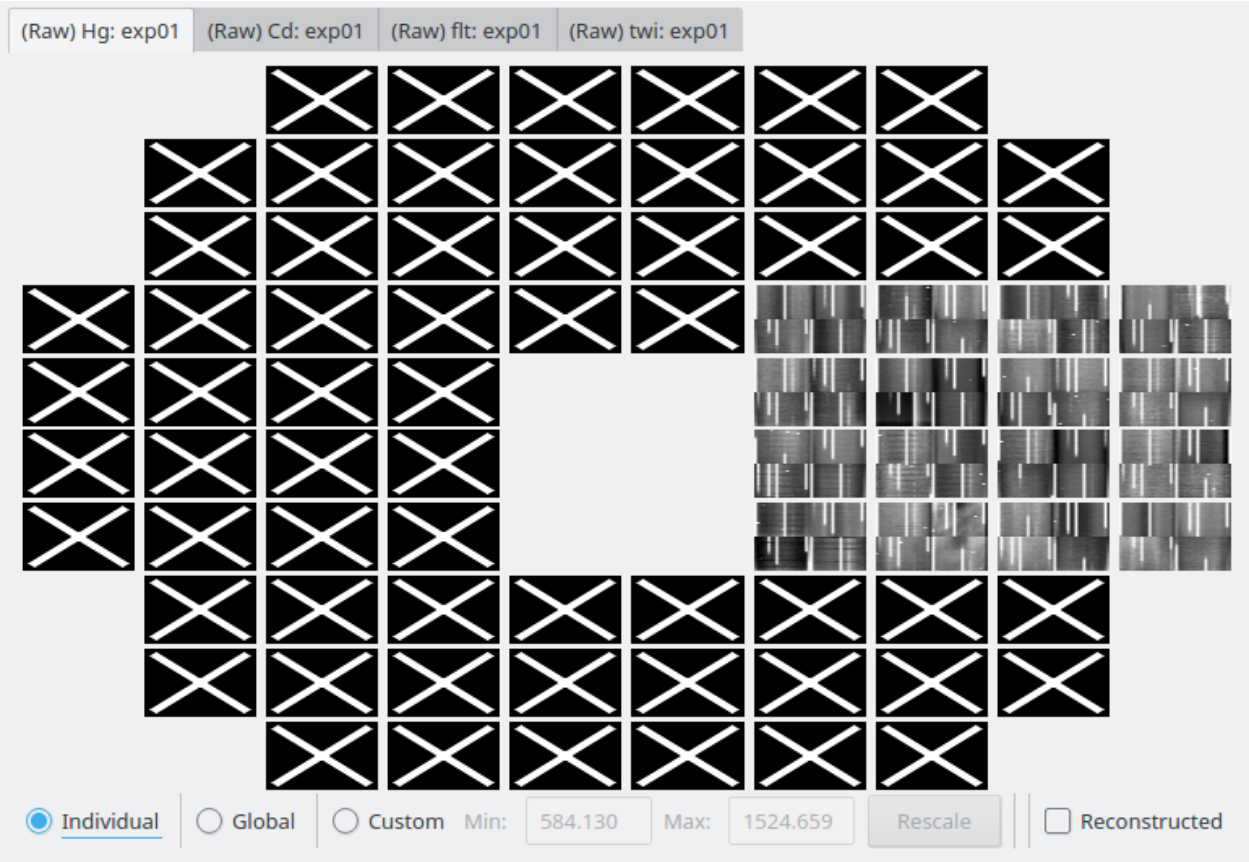


Fig. 3: Screenshot of the tabs created by the `exp_combined` type, when selecting a directory containing two cmp, one flt and one twi shots, each one with one exposure.

```
[reconstruction]
# The name of the IFUCen file
ifucen = ${general:extra_files}/IFUcen_HETDEX.txt
default_dist_r = ${general:extra_files}/defaultR.dist
default_dist_l = ${general:extra_files}/defaultL.dist
pixscale = 0.5
```

VDAT provides a copy of such files and, of course, the user can use other files as fit. The pixel scale gives the size of a pixel: the smaller the number the more pixels will be used in the reconstructed image and the more realistic it will look (with all the caveats given by the fact that the reference files might not be accurate for the files at hand). This comes at the cost of greater time needed to initialize the reconstruction. However we cache the object used to performed the reconstruction, so the impact of the pixel scale will be visible only when creating the first tab that requires it.

In order to be able to run command, at least one IFU must be selected. This can be achieved by clicking on the desired IFUs. When selected, an IFU shows a blue border. Clicking on a selected IFU, will deselect it. It is also possible to select or deselect all the IFUs using *Select menu*.

The tab type also offers the possibility to inspect both the fits files and the reconstructed image in a dedicated window. Double clicking will bring up a window with as many tabs as there are fits files in the IFU or, if clicking on the reconstructed image, with only one tab. The window shows both the fits files data and their header. Via the `header_keys` option in the yaml configuration, it is possible to enhance some header keywords by putting them on top of the list. The window is described with more details in *The FITS viewer*.

`fits` and `fits_combined`

When creating master frames, the user combines multiple exposures into one single fits file. Thus the previous tab type is not suited to display these new files and the `fits` or `fits_combined` type can be used:

`vdat.gui.tabs.entry_points.`**`fits`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one or more tabs of type `tab_widget.FitsFplanePanel`. Each tab represents one file type chosen by the user.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `file_types` (optional): can be `'type'`, `'orig_type'`, `'object'` or a user defined string. This entry is used in the following way:
  - If the value of `file_types` is any of `'type'`, `'orig_type'`, `'object'`: the corresponding information is extracted from the internal database and interpreted as a list. If e.g. the `'orig_type'` is used and the database contains for the entry the following `'cmp, flt'` value, two tabs will be created and the `{orig_type}` formatting key will be replaced with `'cmp'` in one tab and `'flt'` in the other.
  - If the value of `file_types` is any of `'type'`, `'orig_type'`, `'object'` **and** if it present as a keyword in the tab configuration: the content of the keyword is used, instead of the internal information. The value of the keyword must be a list. If e.g. the `'orig_type'` is used and there is one keyword whose value is `['flat', 'arc']`, two tabs will be created and the `{orig_type}` formatting key will be replaced with `'flat'` in one tab and `'arc'` in the other.
  - If the value is a custom string: this string must be present in the tab configuration and its value must be a list. It is also added in the list of available formatting names to allow substitutions. If e.g. the `'custom'` string is used, there must be one such key. If its value is `['my', 'tab']`, two tabs

will be created and the `{custom}` formatting key will be replaced with `'my'` in one tab and `'tab'` in the other.

Default: `'orig_type'`.

- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.
- `header_keys` (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `col`, `row` (`file_name` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type of the file(s) in the target directory.
- `object`: value of the OBJECT header keyword.

See *`interface.plugin_interface()`* for the signature of this function.

`vdat.gui.tabs.entry_points.`**`fits_combined`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as *`fits()`*, but using tabs of type *`tab_widget.FitsAndReconFplanePanel`*.

These types produce one or more tabs that look and behave like the `exp_combined` ones. E.g.

```
zro:
    step_name: Masterbias
    tabs:
        - &masterbias_tab
        tab_type: fits_combined
        file_name: 'masterbias_{ifuslot}_{col}{row}.fits'
        cols: ['L', 'R']
        rows: ['U', 'L']
        title: '{step}'
        tool_tip: 'Reduction {step}. Masterbias'

cal:
    step_name: Masterframes
    tabs:
        - <<: *masterbias_tab
          orig_type: ['flat', 'arc', 'twi']
          file_name: 'master{orig_type}_{ifuslot}_{col}.fits'
          rows: [''',]
          title: 'Master{orig_type}'
```

Will create one tab (`Masterbias`) for a `zro` directory and three (`Masterflat`, `Masterarc` and `Mastertwi`) for a `cal` directory. An example of the latter is shown in the figure below.

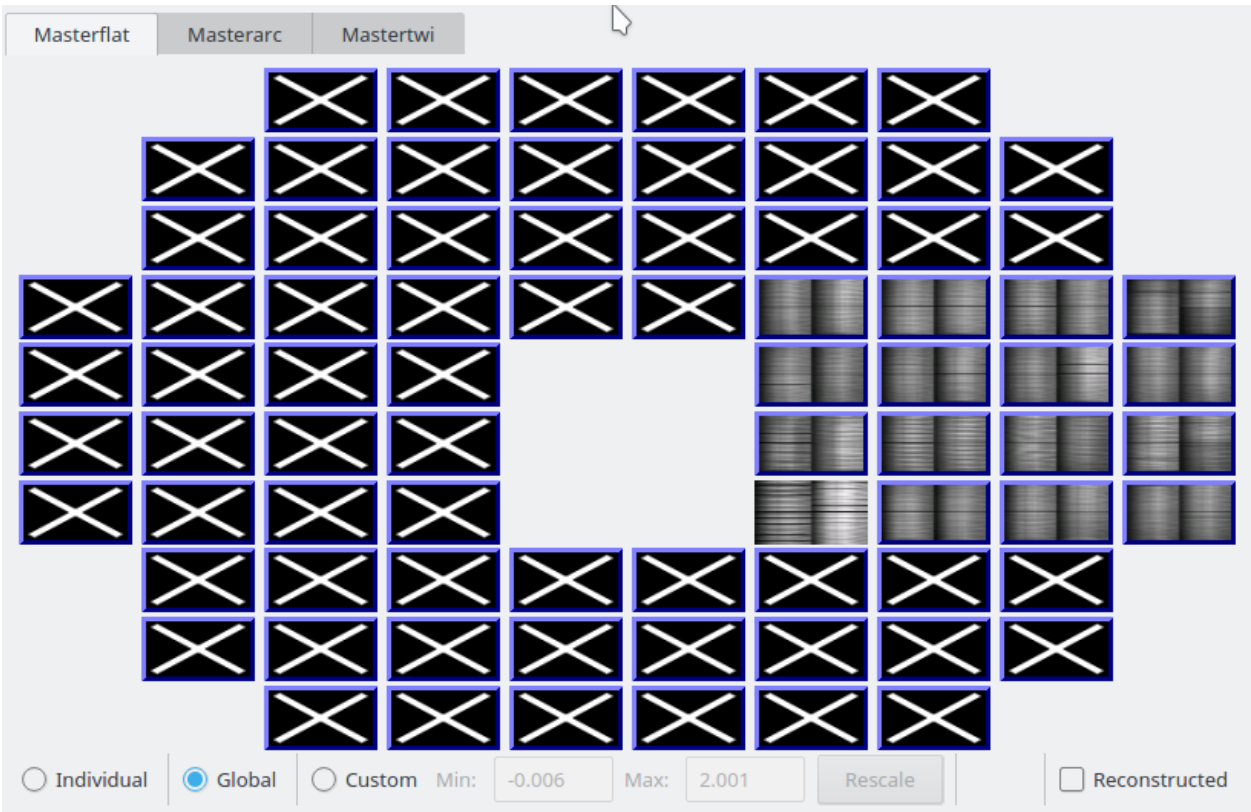


Fig. 4: Screenshot of the tabs created by the `fits_combined` type, when selecting a directory containing a master frame for arcs, flats and twilights.

Without providing the `orig_type` keyword, VDAT would have used the information from the internal database and would have created three tabs with `{orig_type}`, both in the file name and the tab title, replaced by `cmp`, `flt`, `twi`.

If for some reason `orig_type` from the example above does not suffice, it is possible to specify a custom list of file types. The following example will create three tabs called `cmp, flt, twi round`, `cmp, flt, twi bow` and `cmp, flt, twi 42`.

```
cal:
    tabs:
        - <<: *masterbias_tab
          file_types: custom_name
          custom_name: ['round', 'bow', '42']
          file_name: 'master{custom_name}_{ifuslot}_{col}.fits'
          rows: ['',]
          title: '{orig_type} {custom_name}'
```

Clicking and double clicking on each IFU behaves as described at the end of *exp_fits and exp_combined*.

fits_cube

This tab type is almost identical to the `fits` one, except for the fact that is specialised for displaying data cubes. It accepts only one extra configuration option:

`vdat.gui.tabs.entry_points.`**`fits_cube`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as `fits()`, but using tabs of type `tab_widget.CubeFplanePanel`.

On top of the configuration options described in `fits()`, this tab type accepts the following options:

- `z_indx` (optional): before creating the thumbnail for the data cubes, the image is compressed along the z-dimension using the median; if `z_indx` is not given or is `[null, null]`, it uses the whole range, otherwise it uses only the part of the cube in the range [z_indx[0], z_indx[1])

This type produce one or more tabs that look and behave like the `fits` ones. As example take the following configuration entries:

```
- &sci_cube_tab
  tab_type: fits_cube
  title: 'Cube 20'
  file_name: 'CuFeSpdsses_{ifuslot}.fits'
  z_indx: [null, 40]
  tool_tip: 'Compressed cube slice'
-
  <<: *sci_cube_tab
  title: 'Cube 600'
  z_indx: [550, 650]
```

This creates two tabs, called `Cube 20` and `Cube 600`, each showing a compressed slice of the data cube; the former tab shows the data for z index in the range [1, 40] while the latter for [551, 650]. The compression is done using `numpy.nanmedian()`. If `z_indx` is not given or its value is `[null, null]` the full cube is compressed. The following image shows an example of the focal plane created by the above configuration

Clicking and double clicking on each IFU behaves as described at the end of *exp_fits and exp_combined*. In addition in this case the *The FITS viewer* also uses the `z_indx` option.

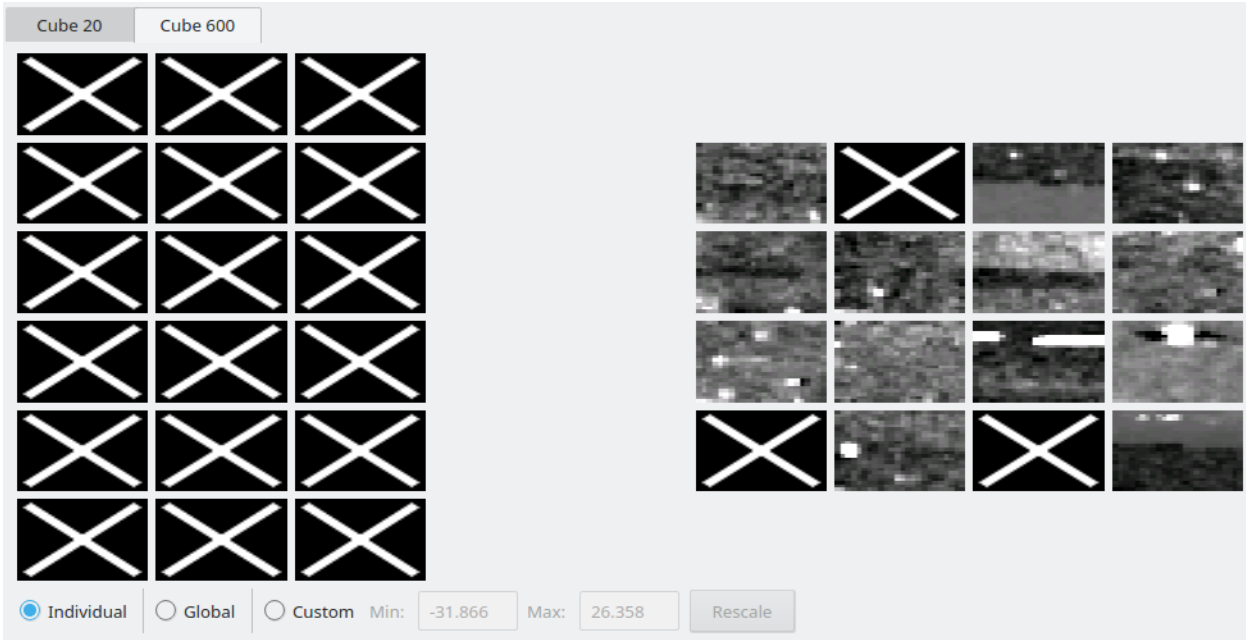


Fig. 5: Screenshot of the tabs created by the `fits_cube` type

fits_multiext

This tab type is almost identical to the `fits` one, except for the fact that it display the desired extensions from multi extensions fits files. It accepts one extra configuration option and provide one more formatting name, described in:

`vdat.gui.tabs.entry_points.`**`fits_multiext`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as *`fits()`*, but using tabs of type *`tab_widget.MultiExtFplanePanel`*.

On top of the configuration options described in *`fits()`*, this tab type accepts the following options:

- `extensions` (list of ints or strings): indices or names of the fits extensions to display in the IFU.

On top of the formatting names described in *`fits()`*, this tab type has this additional formatting name:

- `ext`: index or name of the extension displayed

This type produce one or more tabs that look and behave like the `fits` ones. As example the following configuration:

```
- &cal_fmod
  tab_type: fits_multiext
  orig_type: ['twi', ]
  file_name: 'master{orig_type}_{ifuslot}_{col}.fmod'
  rows: ['',]
  cols: ['L', 'R']
  title: 'mastertwi fmod {ext}'
  extensions: ['AMPLITUDE', 'Y', 'SIGMA', 'H2', 'H3', 'EXP']
```

creates six tabs, called `mastertwi fmod AMPLITUDE`, `mastertwi fmod Y` and so on. Each tab shows the one extension of the fits file called `mastertwi_{ifuslot}_{col}.fmod`. The following configuration:

```
- <<: *cal_fmod
  extensions: [1, 2, 3, 4, 5, 6]
```

creates the same tabs, but with titles like `mastertwi fmod 1` and possibly with a different order. An example of the tabs can be seen in the image below:

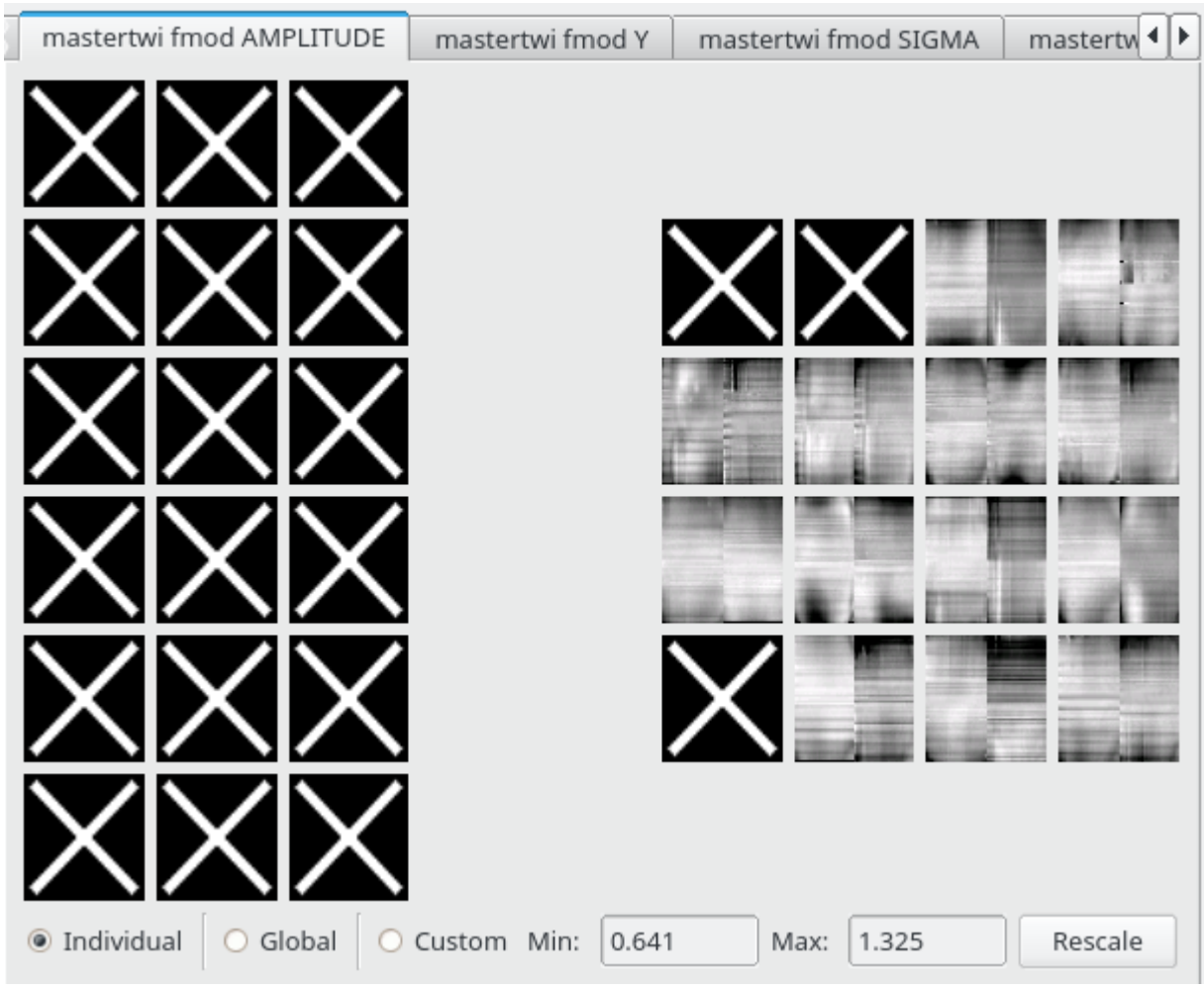


Fig. 6: Screenshot of the tabs created by the `fits_multiext` type

Clicking and double clicking on each IFU behaves as described at the end of *exp_fits and exp_combined*. In addition in this case the *The FITS viewer* also uses the `ext` option.

reconstruct

Science shots usually consist of multiple dithered exposures, typically three for HETDEX observations. The `reconstruct` tab type allows to merge all the exposures of one shot into one image. The type creates always only one tab. The configuration options and formatting names are very similar to the previous two tab types:

`vdat.gui.tabs.entry_points.`**`reconstruct`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return a tab of type `tab_widget.QuickReconFplanePanel`. It collects all the exposures for the `target_dir` and combine all of them in a single reconstructed image.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.

- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.
- `header_keys` (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `col`, `row` (`file_name` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `basename` (`file_name` only): date-time part of the file name.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type(s) of the file(s) in the target directory.
- `object`: value(s) of the OBJECT header keyword.

See *`interface.plugin_interface()`* for the signature of this function.

Reusing the configuration entries shown for `exp_combined`, we can easily create one tab the exposure and add a `reconstruct` tab as shown here:

```
  tabs:
      - *cal_raw_exp_combined
      - <<: *cal_raw_exp_combined
        tab_type: reconstruct
        title: '{step} reconstructed'
        tool_tip: 'Reduction {step}. Object: {object}; type: {type}; original type:
↪{orig_type}'
```

The figure below shows the result for the `reconstruct` type. The tab behaves and feels like in the previous cases, with the exception that there is no `"Reconstructed"` button to switch between the fits files and the reconstructed image.

Clicking and double clicking on each IFU behaves as described at the end of the `exp_combined` section. The

text_file

Some reduction steps create text files and this tab type allow to show them. The number of lines is shown in the focal plane. The configuration options and formatting names are very similar to the previous tab types:

`vdat.gui.tabs.entry_points.`**`text_file`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one tab of type `tab_widget.TextFileWidget`.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.

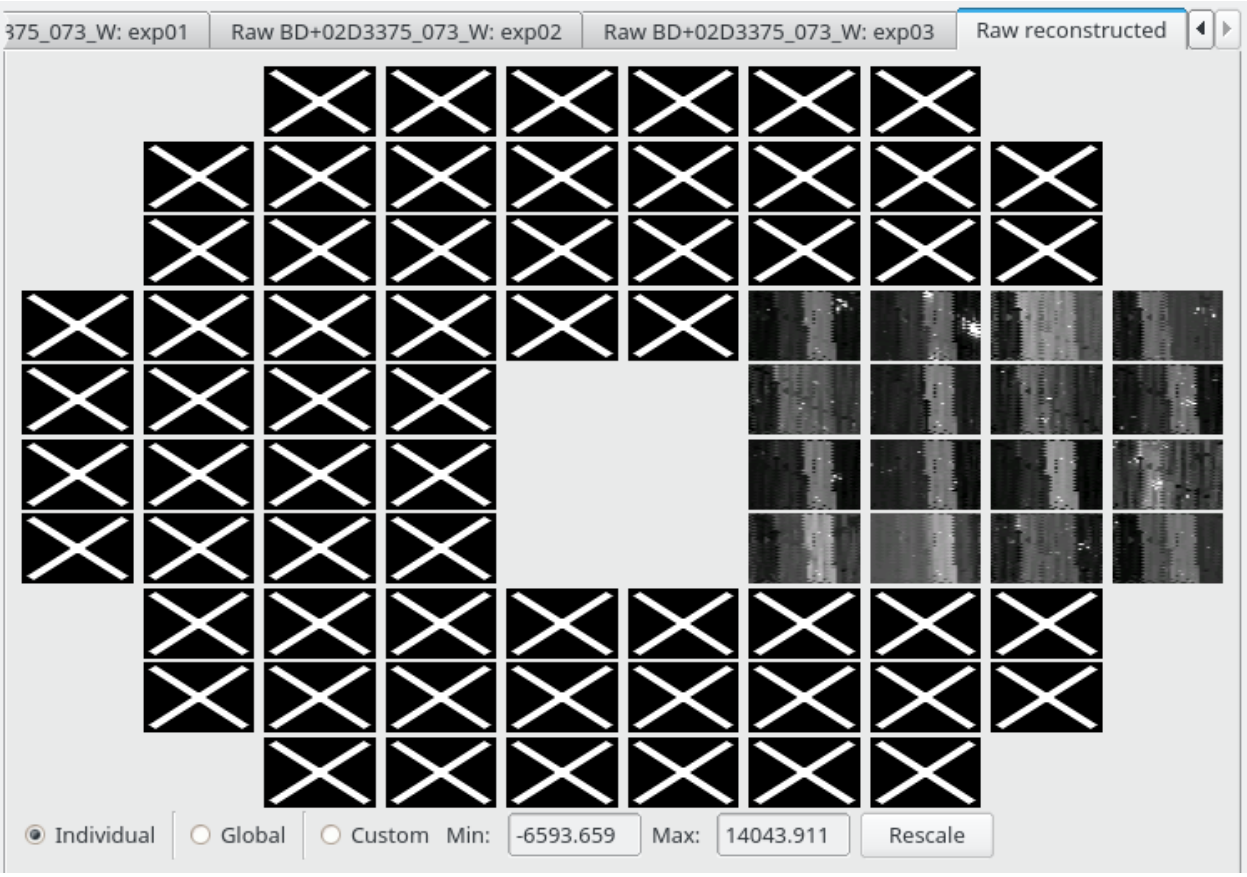


Fig. 7: Screenshot of the tabs created by the `reconstruct` type

- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `title` (optional): title to use for the tab; if absent `'{step}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type(s) of the file(s) in the target directory.
- `object`: value(s) of the OBJECT header keyword.

See *`interface.plugin_interface()`* for the signature of this function.

An example of a configuration entry for the `text_file` tab is shown here:

```
tabs:
    - tab_type: text_file
      file_name: 'dither_{ifuslot}.txt'
      title: 'Dither file'
      tool_tip: 'Reduction {step}. Show the dither file'
```

The figure below shows a tab created with the above configuration. The tab behaves and feels like in the previous cases. However the content is very different: the upper line shows the total number of lines, while the one lower one shows within parenthesis the number of non comment lines. A comment line starts with `#`.

Clicking on each IFU behaves as described at the end of the `exp_combined` section. On double click, a *The text file viewer* window is opened, if the file exists.

dist

One of the products of the `deformer` step are distortion solutions. This tab type shows if distortion files have been produced (i.e. if the `deformer` step worked) and allow to display them. The configuration options and formatting names are :

`vdat.gui.tabs.entry_points.`**`dist`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one tabs of type *`tab_widget.DistPanel`*.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the distortion file(s) to show. It is possible to format the file name using the python formatting syntax.
- `fits_names` (mandatory): list of names of the fits files to use then displaying the distortion in DS9. If the list is empty, it is not possible to display the data on DS9.
- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type}'` is used. It is possible to format the title similarly to the `file_name`.

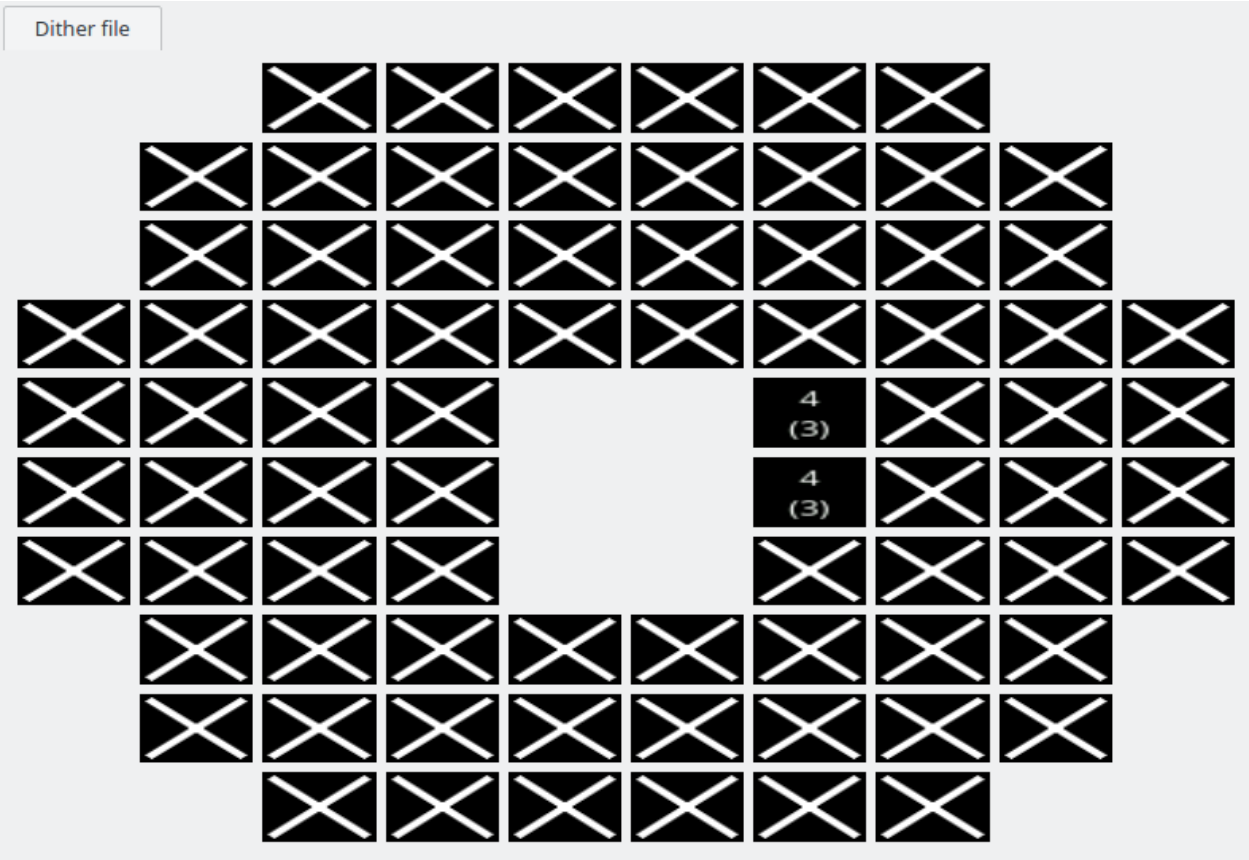


Fig. 8: Screenshot of the tab created by the `text_file` type

- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` and `fits_names` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `col`, `row` (`file_name` and `fits_names` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type of the file(s) in the target directory.
- `object`: value of the OBJECT header keyword.

See *`interface.plugin_interface()`* for the signature of this function.

An example of a configuration entry for the `dist` tab is shown here:

```
tabs:
    - tab_type: dist
      file_name: 'mastertwi_{ifuslot}_{col}.dist'
      fits_names: ['mastertwi_{ifuslot}_{col}.fits', 'masterarc_{ifuslot}_{col}.fits']
      rows: ['',]
      cols: ['L', 'R']
      title: 'mastertwi distortion'
```

The figure below shows a tab created with the above configuration. The tab behaves and feels like in the previous cases. However the content shows the version of the distortion file, in the image `D: 17`, and whether the region file has been created (`R: yes` or `R: no`).

Clicking on each IFU behaves as described at the end of the `exp_combined` section. On double click, a *The distortion file viewer* window is opened, if the at least one of the region files exist.

The logging panel

Todo: describe this

The progress and status bar

VDAT uses the progress bar and the status bar in the lower part of the main VDAT window to communicate the status of the command execution.

A command in VDAT is split up into a series of jobs, each representing a single subprocess, e.g. a Cure command running on a single IFU. When a command is running, the progress bar, as expected, shows the fraction of finished jobs for the command at hand. On Linux and Windows systems it also shows the number of finished jobs, the total number of jobs and the percentage that are finished. The status bar, below it, shows the expanded command of the most recently completed job, truncated if the corresponding string is too long.

The image below shows an example of it. Note that the aspect of the bar might change depending on the operating system and the desktop environment, as described here.

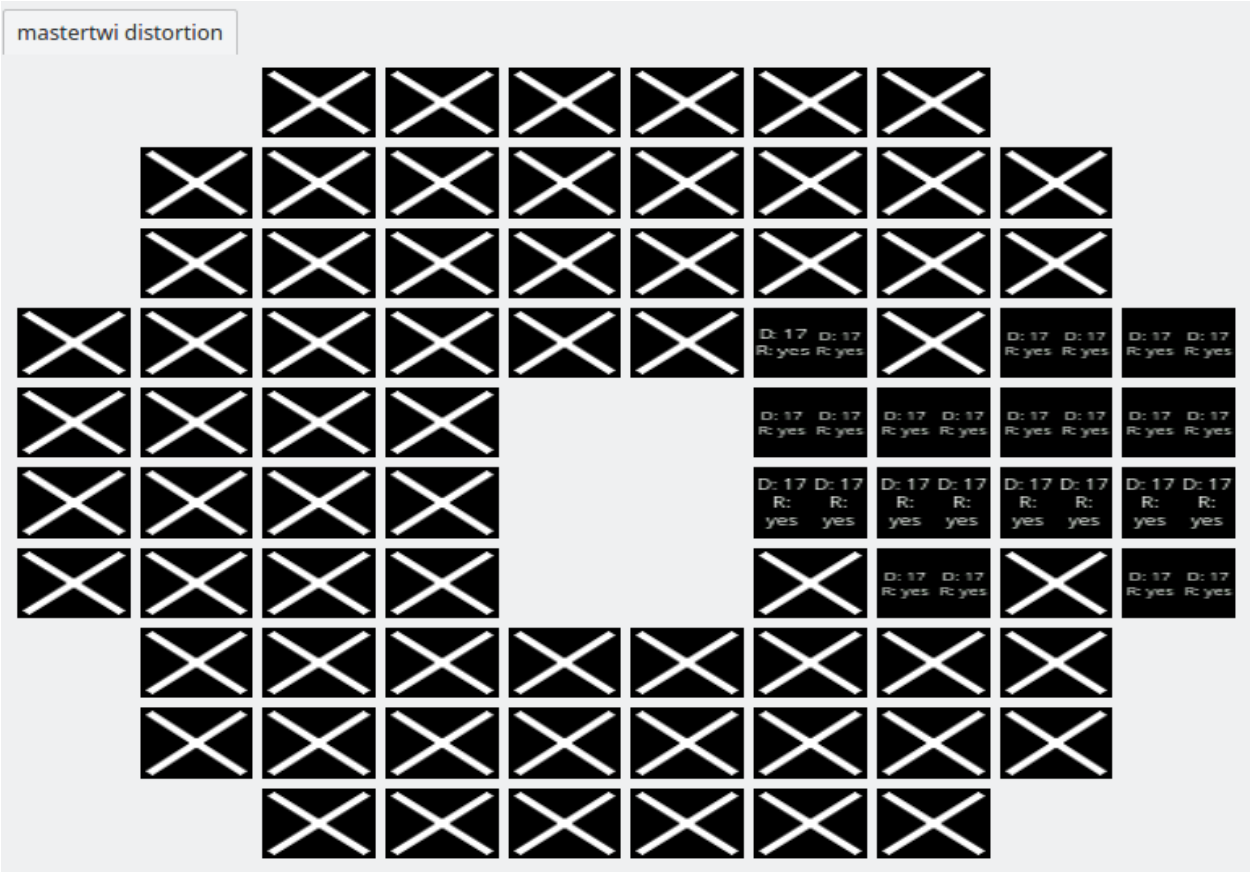
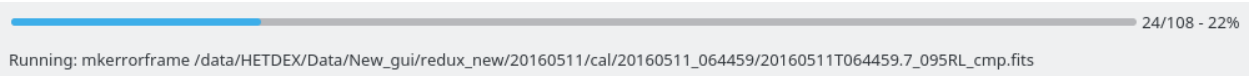
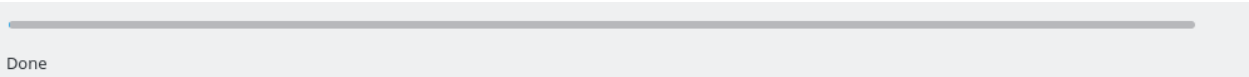


Fig. 9: Screenshot of the tab created by the `dist` type



When a command finishes running, the progress bar gets reset and the status bar flashes the message `Done` for a few seconds, as shown below.



1.4.2 IFU viewer

IFU Viewer

All of the tabs described in *Builtin tab types* allow to open a window to better inspect the content of one IFU by double clicking on it.

Here we describe the various IFU Viewer that are shipped with VDAT.

The FITS viewer

The fits viewer allows to display the content of one or more fits files in separate tabs together with their header information. The figure below show an example of such a window:

The window consist of multiple tabs, one per file and each tab is divided in two parts:

- The upper part contains a widget based on the Ginga fits viewer that allow to explore a fits file in details through, e.g., zooming, panning, changing scale method and color map. All the available Ginga key bindings, that allow to change things like color map, are enabled. The `Help` menu contains a link to the documentation of the available key bindings.

  If the input fits file is a data cube, it shows a compressed splice along the `z` direction. Custom z indices can be given using the `z_indx` configuration option, as described in *fits_cube*.

  If the configuration dictionary passed to the FITS viewer contains the `ext` option, the corresponding extension of the fits file will be displayed. If no custom title is used (as e.g. in the `exp_*` and `fits*` tab types), the extension name or index, i.e. the value of `ext`, is appended within square brackets to the tab name, as shown in this figure:

- The lower part shows the full file header. By default header keys are shown in the same order as they appear in the file. However it is possible for the user to have one or more header keywords to be shown on top of all the others using the `header_keys` option in the tab configuration entries for the tabs supporting it. `header_keys` expects a list of string. When provided, the headers keywords specified are searched and put on top of all the others in the same order as they are given. Non existent keywords are ignored. If at least one keyword is found, a separator is added. The figure above was created using

  ```
  header_keys: ['SPECID', 'IFUID', 'IFUSLOT', 'CCDPOS', 'CCDHALF']
  ```

By default, the title and the tool tip of each tab are the file name and its absolute path. However those can be customized. E.g. when double-clicking on one IFU reconstructed image, the only tab shown has `'Recontructed'` for title and the list of all files used to create the image for tool tip message.

If the Ginga viewer is not enough for the user needs, it is also possible to send the files to DS9, if available in the system. The `ds9` menu allows to send the file shown in the current tab to a new or an existing DS9 session. The figure shows the menu as it appears if one DS9 window is already open. In case of data cubes, the full underlying fits file is sent to DS9, not only the visible slice. For multi extension files, only the current extension is sent to DS9.

Fig. 10: Screenshot of the fits viewer windows.

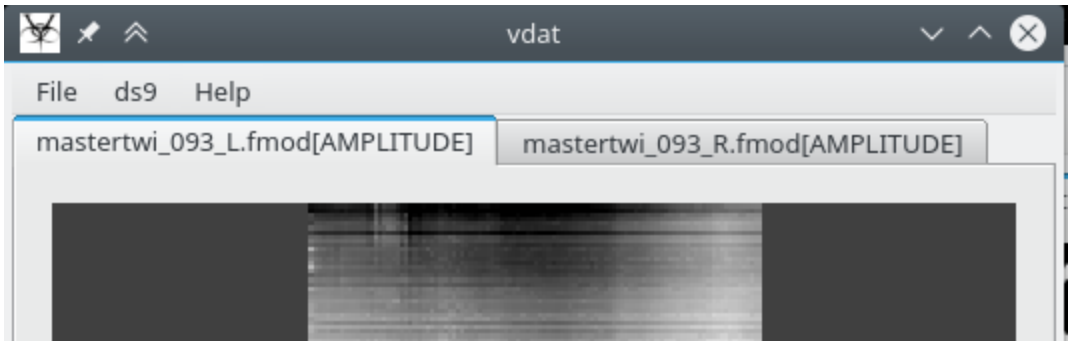


Fig. 11: Screenshot of the upper part of the fits viewer windows when a fits extension is explicitly chosen.

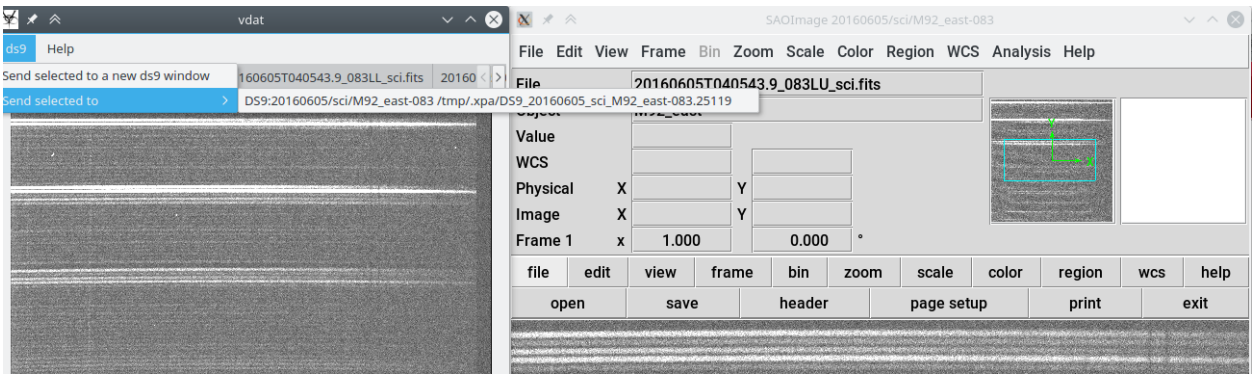


Fig. 12: Example of the `ds9` menu used to send the current immage to a DS9 instance

The text file viewer

The text file viewer displays the content of a file as plain text and allows basic editing functionalities. The following image shows the window displaying a typical dither file:

It is possible to edit the file and save it, also to a different name. Default functionality, like undo/redo, selection and navigation are available via the key bindings listed below and described in the QTextEdit documentation:

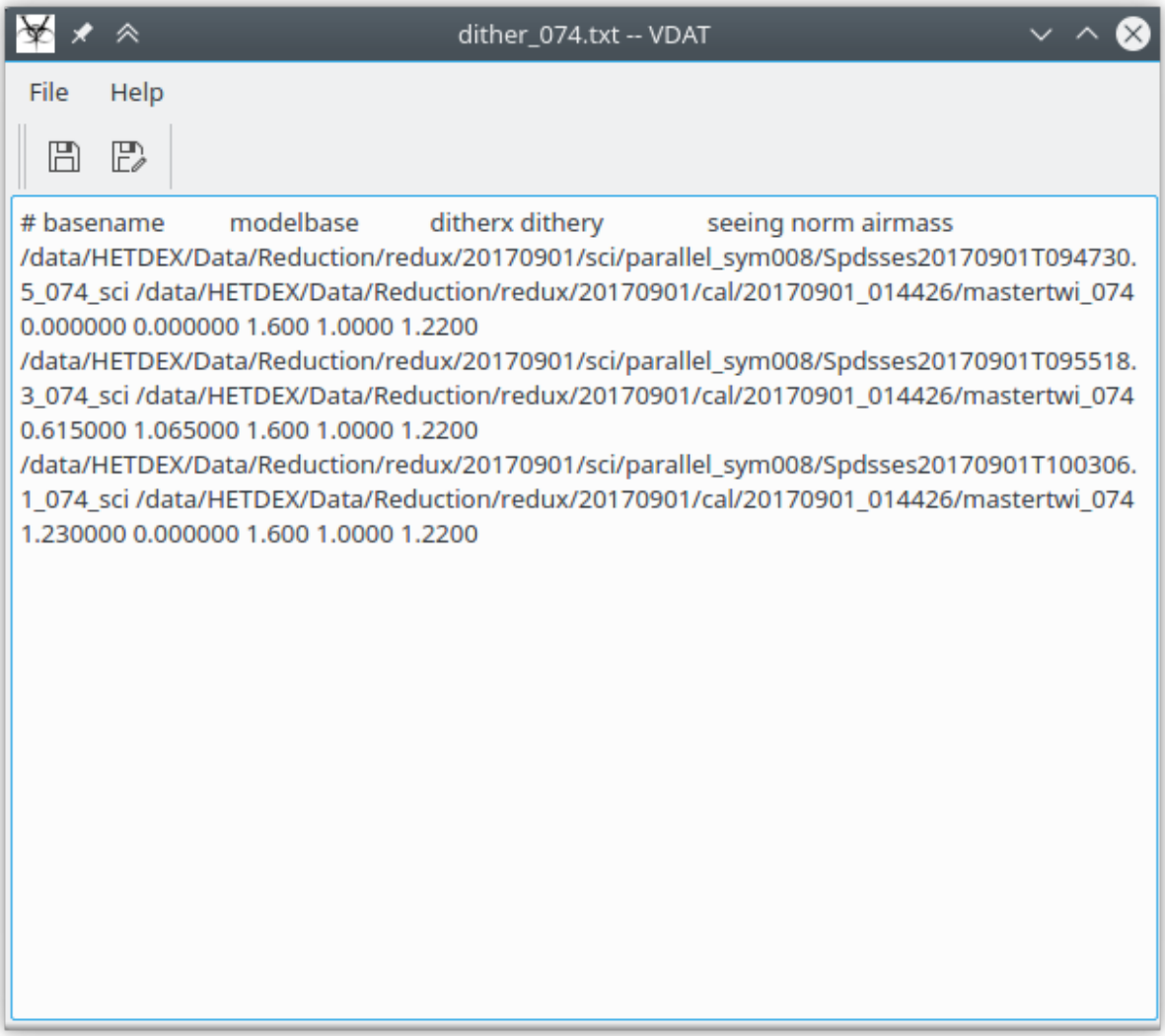


Fig. 13: Screenshot of the text file viewer window.

Keypresses	Action
Backspace	Deletes the character to the left of the cursor.
Delete	Deletes the character to the right of the cursor.
Ctrl+C	Copy the selected text to the clipboard.
Ctrl+Insert	Copy the selected text to the clipboard.
Ctrl+K	Deletes to the end of the line.
Ctrl+V	Pastes the clipboard text into text edit.
Shift+Insert	Pastes the clipboard text into text edit.
Ctrl+X	Deletes the selected text and copies it to the clipboard.
Shift+Delete	Deletes the selected text and copies it to the clipboard.
Ctrl+Z	Undoes the last operation.
Ctrl+Y	Redoes the last operation.
LeftArrow	Moves the cursor one character to the left.
Ctrl+LeftArrow	Moves the cursor one word to the left.
RightArrow	Moves the cursor one character to the right.
Ctrl+RightArrow	Moves the cursor one word to the right.
UpArrow	Moves the cursor one line up.
Ctrl+UpArrow	Moves the cursor one word up.
DownArrow	Moves the cursor one line down.
Ctrl+Down	Arrow Moves the cursor one word down.
PageUp	Moves the cursor one page up.
PageDown	Moves the cursor one page down.
Home	Moves the cursor to the beginning of the line.
Ctrl+Home	Moves the cursor to the beginning of the text.
End	Moves the cursor to the end of the line.
Ctrl+End	Moves the cursor to the end of the text.
Alt+Wheel	Scrolls the page horizontally (the Wheel is the mouse wheel).
Ctrl+Wheel	Zooms the text.

The distortion file viewer

This window displays the content of one or more distortion files. More importantly, it also provides a few menus that sends fits files and distortion solutions to DS9. Under the `ds9` menu there are the following sub menus:

- `Distortion and fits`: send the `fits_names` (from the *dist tab type configuration*) either to a new or an existing DS9 window (as done in *The FITS viewer*) and overlay the distortion solutions to all the above fits files;
- `Distortion only`: overlay the distortion solution to a file in an existing DS9 window; this menu entry is useful to overlay a distortion solution to e.g. a science fits file.

In both cases only the distortion in the currently visible tab and the related fits files are sent to DS9.

The following image shows an example the distortion window and of the DS9 window with two fits files sent (`mastertwi` and `masterarc`) with the distortion solution overlaid in to them in green:

1.4.3 Queue window

The Command Queue Window

When the user presses a command button, the commands associated with that button are added to a queue. Commands are then extracted from the queue one at a time and executed.

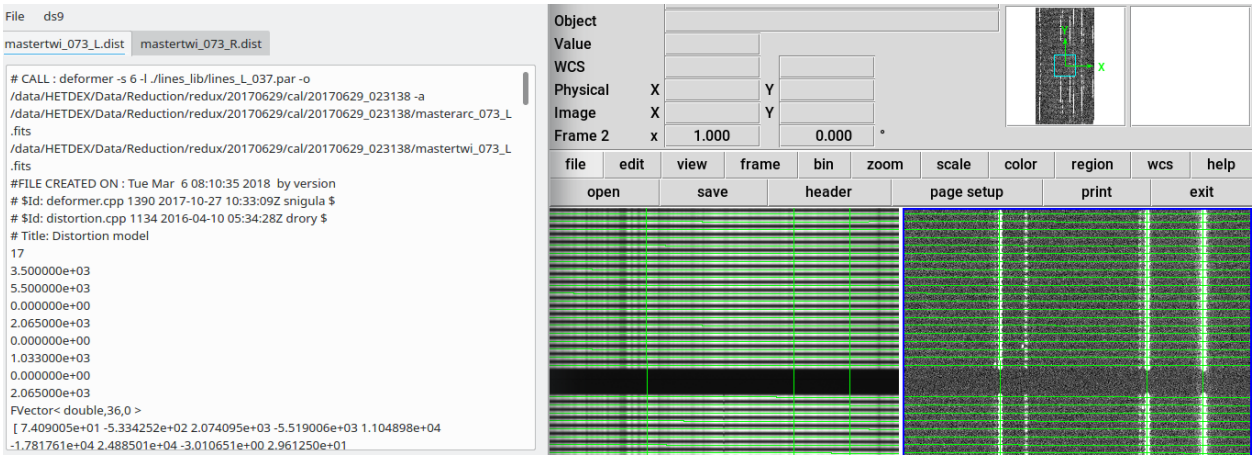
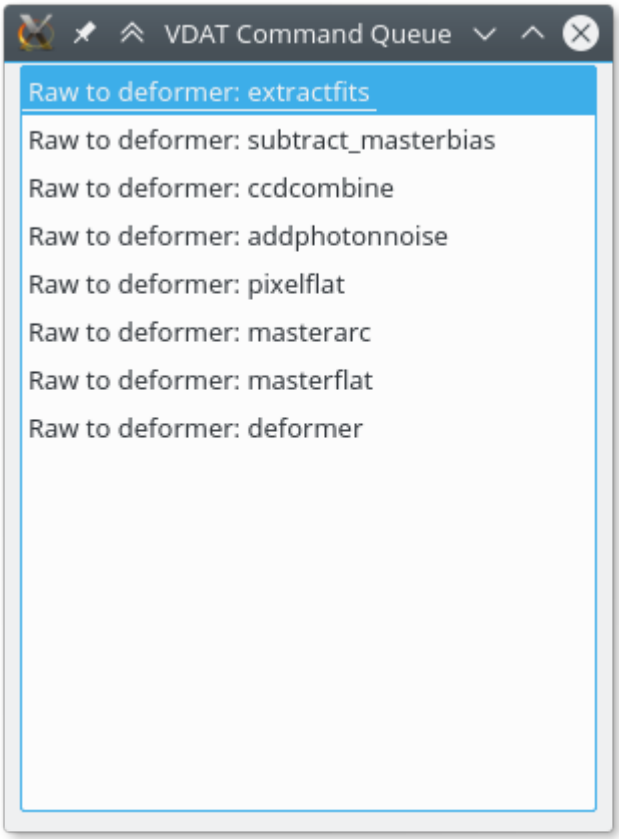
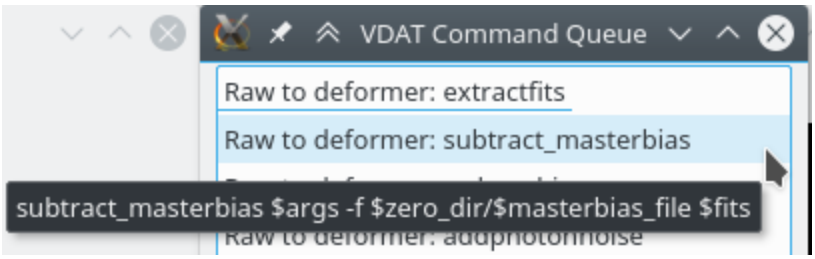


Fig. 14: Screenshot of the distortion file viewer window and of a ds9 window



The VDAT command queue window, shown in the right image, allows the user to monitor the status of the queue. Each entry in the queue is shown with a name composed by the name of the button pressed and of the command. E.g. the selected item has been inserted pressing a button called "Raw to deformer" and will execute the command "extractfits". Hovering over one item will show a tooltip with the full command string, as shown in the image below. See *The Command Interpreter Tool* for details on the structure of the command string.

Commands can be removed from the queue by selecting them and hitting the `Delete` key. Apple computer users might need to use `fn + Delete`.



1.5 The Command Interpreter Tool

This chapter describes the part of `VDAT` responsible for the reduction of the data.

1.5.1 Introduction

The main scope of the `VDAT` GUI is to allow users to select, visualize and reduce `VIRUS` data. `VDAT` relies mostly on `cure` to execute the reduction steps. `cure` is a C++ library that provides a number of executables that operate on a single or group of fits files.

For each of the reduction steps, `VDAT` must collect (i.e. generate a list of) the input files and command line options according to the directories and IFUs selected by the user and run the appropriate `cure` tool.

Although `cure` is the main library of tools to use, some of the steps of the reduction are not implemented there. We also want to allow users to execute generic commands without any prior knowledge of the signature and name of the files.

We have solved those requirements by designing a command line tool based on these two building blocks:

1. an interpreter that parses an input *command string*, containing placeholders, and executes the command in a loop replacing the placeholders with the correct values; we use the standard python string Template to define placeholders;
2. one or more yaml configuration files to instruct the interpreter on how to expand the placeholders for any provided command. In this documentation we'll refer to entries in the configuration as `keys`.

1.5.2 The interpreter

The public interface of the interpreter is defined by the constructor of the class `CommandInterpreter` and its method `run()`.

Constructor

The constructor has the following signature

class `vdat.command_interpreter.`**`CommandInterpreter`**(*command*, *command_config*, *selected=None*, *multiprocessing=False*, *processes=None*)

Interpret and execute the command.

See *The interpreter* section in the documentation for more details

All the custom errors are defined in `vdat.command_interpreter.exceptions`. The ones raised in the constructor are derived from from `CIValidationError`,

Parameters

command [string] command to parse

command_config [dict] dictionary containing the instructions to execute the `command`. A deep copy is executed

selected [list-like, optional] None or a list of the selected items; if `None` no filtering of the primary files is done; otherwise must be an object supporting the membership test operator `in`.

multiprocessing [bool] run the command using multiprocessing

processes [int] number of processors to use

Raises

CINoExeError if the executable does not exists

CIParseError if there is some error when extracting the keywords

CIKeywordError for missing keywords or for keywords of wrong type

CIKeywordTypeError if the type of the keywords is not among the known ones

1. `command` is a string with the command to execute. The command contains what we will refer to as `fields` that will be substituted. For example, the `fields` in the command string below are `args`, `biasec` and `fits`

```
subtractfits $args -o $biassec $fits
```

2. `command_config`: the relevant part of the parsed `yaml` configuration file containing the instructions on how to expand fields like `args`, `biassec` and `fits` while running the command `subtractfits`. The part of the configuration file necessary to run the above command is

```
subtractfits:
    # mandatory fields
    mandatory: [fits, ]

    # primary key: the interpreter collects files according to
    # the instructions in the `fits` key, then loops over them,
    # replacing all the fields and executing the command
    primary: fits

    # looks for all the files matching the pattern in the `selected_dir`
    fits: '[0-9]*.fits'

    # Get the `BIASSECT` value from the header of every file
    # and from it extract the part within square brackets
    biassec:
        type: header
        keyword: BIASSEC
        extract:
            - \[(.*)\]
            - \1

    args: '-s -a -k 2.8 -t -z'
```

The syntax for the expression given for the `fits` and `extract` keys will be explained later.

Both the GUI and the command line interface inject into the `command_config` the following keys:

- `target_dir`: is the directory selected by the user; in the above examples, the `fits` files are searched in this directory
- `cal_dir`: the reference calibration directory

- `zro_dir`: the reference bias directory

If no directory `cal` or `zro` has been explicitly selected in the GUI, the default ones are added.

> **Warning:** If any of these entries is already in the configuration file, they will be overwritten

3. `selected`: list of selected items or `None`, for selecting all. It tells the interpreter which of the `primary` elements must be run. E.g. the `VDAT` GUI passes as `selected` the list of IFUs selected by the user. The instructions on how to extract the information to match against `selected` from the files while running the command is defined in the *command configuration file*.

   > **Note:** VDAT passes the IFU head mount plate IDs (ihmpid) to the command interpreter. This id is a 3 digit number stored in the file headers under the IFUSLOT key.

In the constructor the following steps are performed:

1. the configuration object is copied and saved in local variables: this allows the user to enqueue multiple commands;
2. **validations:**
   (a) the command executable, e.g. `subtractfits`, is searched in the path to check if it exists
   (b) check that all the mandatory `fields` are present in the command (see the `mandatory` key in the `command_config` above)
   (c) check that all the required `keys` are present in the configuration
   (d) check that all the required `keys` are of known type
   (e) map all the types to the functions implementing them

1.5.3 The configuration file

To allow for flexibility and extendability, the instructions on how to expand `fields` come from one or more configuration files, written using the `yaml` standard.

When validating the `command` string, the `fields` and the name of the command are extracted and corresponding `keys` are searched for in the configuration, under the section specific to that command. The value of a `key` can be either a string or a python dictionary. If it's a string, like `'-a -b'`, it is converted into a `key` of type `plain`: `{'type': 'plain'; 'value': '-a -b'}`. If it is a dictionary, it must contain a Python dictionary entry called `type`, whose value defines the type of the key.

Non-substituted keys

These are `keys` that are understood and used by the interpreter, but do not represent `fields` that will be expanded/substituted in the command line calls.

`is_alias_of`

If it exists, its value is the real name of the executable. This allows the creation of multiple commands using the same underlying executable. If e.g. the command is:

```
do_something $args -o $ofile $ifiles
```

and the configuration file contains

```
is_alias_of: an_executable
args: "-a -b"
ofile: outfile
ifiles: file[1-9].txt
primary: ifiles
```

then the interpreter will loop through all the files matching the `ifiles` pattern in `target_dir`. For the first file, it will execute:

```
an_executable -a -b -o outfile file1.txt
```

mandatory

List of mandatory `fields`; field names defined under `mandatory` must exist in the provided command. Do not provide this key or leave it empty to disable these checks.

```
mandatory: [ifiles]
# or equivalently
mandatory:
    - field1
    - field2
```

primary

Name of the `field` to use as primary. A primary `field` has a special status: files are collected from the `target_dir` according to the type of the underlying `key`, then they are looped over and for each step the command string is created and executed. If the value of any other `key` or `field` needs to be built at run time, it will use the `primary` files to do it. `VDAT` is shipped with few *primary types*.

This key can have either a single value or a list of values. If it has a single value, the corresponding `field` must be present in the command. If it is a list of values, **one and only one** of the `fields` must be present in the command. Multiple primary `fields` are not allowed.

```
# single primary
primary: fits
# mutiple primaries
primary: [fits1, fits2]
```

filter_selected

Tells the interpreter how to filter the list of primary files. If this option is not found in the configuration or the `selected` keyword in `CommandInterpreter` is `None`, no filtering is performed. Otherwise, for each element in the primary list:

- uses the instructions from the value of `filter_selected` to extract a string
- check if the string is in `selected`.

The value of `filter_selected` can be any available `key` type, e.g. *the built-in ones* described below.

With the following settings:

```
# Use the value of the header keyword ``IFUSLOT`` to decide whether to
# keep the primary field or not
filter_selected:
    type: header
    keyword: IFUSLOT
```

the content of the fits header keyword `IFUSLOT` is extracted and compared with the list provided with the `selected` options in `CommandInterpreter`

execute

For each iteration of the `primary`, it tells the interpreter whether to run the command or not. If the option is not found, no filtering will be performed. `VDAT` ships some *execute types*.

If the handling of this `key` raises an exception, it is logged as a warning and the command is executed as if the `key` returned `True`.

Built-in primary key/field types

plain

Search for files matching the given pattern in the target directory. If the value of a `key` is a string, it is interpreted as a `plain` type. These three definitions are equivalent:

```
keyword: 20*.fits
---
keyword: &plain
    type: plain
    value: 20*.fits
---
keyword: {type: plain, value: 20*.fits}
```

By default, the keyword values are interpreted as shell-style wildcards. As in the fnmatch the only special characters are:

Pattern	Meaning
`*`	matches everything
`?`	matches any single character
`[seq]`	matches any character in *seq*
`[!seq]`	matches any character not in *seq*

If you need more complex matches, it's possible to use python regular expressions. To make the interpreter aware of it you can add the optional key `is_regex` and set it to `True`. For example:

```
keyword:
    type: plain
    value: '(?:e.)?jpes[0-9].*fits'
    is_regex: True
```

will get all the files in the `target_dir` whose name matches `e.jpes[0-9]*fits` or `jpes[0-9]*fits`, but not, e.g., `FEjpes[0-9]*fits`,

If rather than returning the filename we just one to extract some part of it, e.g. just the time stamp , we can add the `returns` option with the corresponding instructions. The content of returns can be any available secondary keyword:

```
keyword:
    <<: *plain
    returns:
        type: regex
        match: '.*(\d{8}T\d{6}).*'
        replace: \1
```

here the `\1` refers to the first `regex` group returned from the expression in `match`.

loop

This is designed to loop over, for example, IFUs, channels and/or amplifiers.

1. collects the `keys` which have been stored under a `yaml` key called (a little confusingly) `keys`
2. cycles through all the possible combinations of them
3. for each combination replaces the corresponding entries in `value` (see example below) using the standard python format string syntax
4. look for all the files matching the resulting strings
5. if any files are found, construct a string. If multiple files are found, construct a single string with the different files separated by a space.
6. if the `returns` option is given, uses it to manipulate the string with the file names (as explained above)
7. yields the string

The entries stored under the `keys` key are maps between the names of the entries, e.g. `ifu` and the values that they can have in the loop described in step (2) above. Their value can be either a list or three comma separated numbers: `start, stop, step`. The latter case is converted into a list of numbers from `start` to `stop` excluded every `step`.

The following configuration:

```
keyword:  &loop
    type: loop
    value: 's[0-9]*{ifu:03d}{channel}{amp}_*.fits'
    keys:   # dictionary of keys to expand in ``value``
        ifu: 1, 100, 1      # start, stop, step values of a slice
        channel: [L, R]     # a list of possible values
        amp:                # alternative syntax for the list
            - L
            - U
```

cycles through all the possible combinations of the three lists: `[1, 2, .., 99]`, `['L', 'R']` and `['L', 'R']`. For the first combination we get: `ifu`: 1, `channel`: L, `amp`: L and `value` becomes `s[0-9]*001LL_*.fit`. Then all the files matching this pattern are collected.

As with the *plain primary keyword*, it's possible to interpret the strings resulting from filling in the fields in `value` as regexes by providing the optional key `is_regex`. As before, one can also extract some part from the file name(s) with the `returns` key.

groupby

1. collects all the files matching `value` and loops through them
2. for each of the files replace `match` with all the values in `replace` using the *regex* secondary keyword implementation.

The following configuration:

```
keyword:
    type: groupby
    value: 'p*[0-9][LR]L_*.fits'
    match: (.*p.*\d[LR])L(_.*\.fits)
    replace:
        - \1U\2
```

cycles through all the files matching `value` in the `target_dir`, e.g. "p2LL_sci.fits", and for each of them creates a new file name replacing the last "L" with "U", e.g. "p2LU_sci.fits". The two files are then returned.

To create multiple files out of the first one, it's enough to provide other entries to `replace`. E.g.:

```
replace: [\1U\2, \1A\2, \2_\1]
```

will create three new files: "p2LU_sci.fits", "p2LA_sci.fits" and "_sci.fits_p2L"

As with the *plain primary keyword*, it's possible to interpret the `value` as a regex providing the optional key `is_regex`. All the keywords recognised by *regex* secondary keyword are also supported.

all_files

This primary type has the same interface of the *plain primary keyword*. The behaviour is however different: while the `plain` primary keyword return an iterator (or list) of file names or strings, `all_files` returns a list containing a single string of space separated file names or, when using the `returns` option, values.

The following configuration collect all the files matching `value` as explained in *plain* and returns a list with a single element:

```
keyword: &all_files
    type: all_files
    value: 20*.fits
```

If e.g. there are four files matching the pattern, the type returns something like:

```
['/path/to/20180219T071318.8_073LL_sci.fits /path/to/20180219T071318.8_073LU_sci.fits␣
↪/path/to/20180219T072418.2_106RL_sci.fits /path/to/20180219T072418.2_106RU_sci.fits
↪']
```

For comparison, the `plain` primary type would return:

```
['/path/to/20180219T071318.8_073LL_sci.fits',
 '/path/to/20180219T071318.8_073LU_sci.fits',
 '/path/to/20180219T072418.2_106RL_sci.fits',
 '/path/to/20180219T072418.2_106RU_sci.fits']
```

The `regex` and `returns` options are interpreted as described in *plain*

> **Warning:** The *filter_selected* option is used to select which of the elements returned by the primary key are to be used. They are not used to filter substrings of the elements returned by the primary key. So using `filter_selected` with `all_files` might lead to unexpected results and we suggest to avoid the option

Build-in keyword types ——————-

`plain` ^^^^^^^^^

A static string. These three definitions are equivalent:

```
keyword: '-a -b --long option' --- keyword: type: plain value: '-a -b
--long option' --- keyword: {type: plain, value: '-a -b --long option'}
```

regex

Returns a string obtained from primary replacing `match` with `replace`. It uses `re.subn()` to do the substitution. If e.g. the primary is `file_001_LL.fits file_001_RL.fits`, the following entry returns `L001`

```
keyword:
    type: regex
    match: \S*?_(\d{3})_([LR]).*?\.fits
    replace: \2\1
```

If the substitution fails because of a regex mismatch or because more than one substitution is performed, a *`CIKeywordError`* is raised. It is possible to declare the expected number of substitutions or to disable the check altogether via the optional `n_subs` key:

- if not present, defaults to one, if `do_split` is `True`, or to the number of input primary files, otherwise;
- if a negative number: the check is disabled;
- positive integer: exactly `n_subs` must be performed. E.g:

  ```
  keyword:
      type: regex
      match: \S*?_(\d{3})_([LR]).*?\.fits
      replace: \2\1
      n_subs: 2
  ```

  will fail because it requires **two** substitutions;
- list of integers: the number of substitutions must be one of the list entries:

  ```
  keyword:
      type: regex
      match: \S*?_(\d{3})_([LR]).*?\.fits
      replace: \2\1
      n_subs: [1,2]
  ```

  will accept either one or two substitutions;
- string: interpreted as a slice `[start]:[stop][:step]` or a comma separated list of `[start],[stop][,step]`. The string is used to initialize a *`SliceLike`* instance and then to check if the number of substitutions is within the allowed range as defined in the class documentation. E.g the following will succeed:

```
keyword:
    type: regex
    match: \S*?_(\d{3})_([LR]).*?\.fits
    replace: \2\1
    n_subs: 1:10:2
```

but using `n_subs: 0:10:2` with raise an error.

Finally the `do_split` optional key will instruct the function whether to split the primary on white spaces or not. E.g.:

```
keyword:
    type: regex
    match: \S*?_(\d{3})_([LR]).*?\.fits
    replace: \2\1
    do_split: False
```

with return `L001 R001` from the files `Sfile_001_L.fits Sfile_001_R.fits` as a single string. If not provided, it defaults to `True`.

Examples and more information about the python regex syntax can be found in the official python documentation

header

Extract and manipulate the fits header keyword named in `value` from the primary files.

If the optional keyword `do_split` is `True` (the default) it splits the primary on white-spaces and gets `value` only from the first file. Otherwise `value` is extracted from every file, converted to a string and concatenated with white spaces. Assuming that the primary consists of two files containing `BIASSEC = [1:32,1:1032]` in the header, the following instruction:

```
keyword: &header
    type: header
    value: BIASSEC
```

will return `[1:32,1:1032]`.

By default the value of the header keyword is cast to a string. However sometimes it is desirable or necessary to format it, e.g. padding an integer with zeros. Via the `formatter` key, it is possible to format the header keyword value according to standard python format string syntax. E.g. it is possible to convert the integer header keyword `IFUSLOT` (`42`) to a zero padded-three digit string (`042`) with the following definition:

```
keyword:
    type: header
    value: IFUSLOT
    formatter: '{:03d}'  # or '{0:03d}'
```

> **Warning:** the `:` or `0:` part is mandatory, otherwise a `KeyError` will be raised. If the formatting code is wrong for the type a `ValueError` is raised with a message similar to "Unknown format code 'd' for object of type 'str'"

It is also possible to manipulate the return value using the *regex* secondary keyword. To do this, add an `extract` keyword, whose value is a two element list containing the regex pattern to match and the desired return value which can reference the matched regex groups. E.g.:

```
keyword:
    <<: *header
    extract:
        - \[(.*?)\]
        - \1
```

will return `1:32,1:1032` as `\1` will return the first regex group, i.e. whatever is contained within the round brackets.

If the above instructions contained `do_split: False`, the return values would have been `[1:32,1:1032] [1:32,1:1032]` and `1:32,1:1032 1:32,1:1032` respectively.

format

Creates a new string formatting `value` using the `keys`. They can be of any secondary type known to VDAT at loading time, except `format` to avoid circular recursion. Assuming you have a fits file called `file_001_LL.fits`, with a header keyword `DATE-OBS = 2013-01-01`, the following configuration instructs the interpreters to extract the `id` key, a three digit number, from the file name and the `DATE-OBS` fits header value.

```
keyword:
    type: format
    value: file_{id}_{sec}.fits
    keys:
        id:
            type: regex
            match: .*_(\d{3}).*\.fits
            replace: \1
        date:
            type: header
            value: DATE-OBS
```

The resulting value is the string `file_001_2013-01-01.fits`. If the types for the keys do not exist, a `CIKeywordTypeError` will be raised at run time. If one of the keys has a string as value, it will be interpreted as of type `plain`.

As in the previous cases, if `do_split` is present and `False`, the formatting is applied to all the elements in the primary and a concatenated string of white-space separated results is returned.

fplane_map

This type allows to maps from one type of ID to an other using the fplane file. The following code shows all the mandatory keys; their explanation can be found below.

```
keyword:  &fplane_map
    type: fplane_map
    fplane_file: /path/to/fplane.txt
    in_id:
        type: regex
        match: '.*?/dither_(\d{3})\.txt'
        replace: \1
    in_id_type: ifuslot
    out_id_type: ifuid
```

where:

- `fplane_file` points to the fplane file

- `in_id` can be any of the available keyword types and is used to extract the ID from the primaries.
- `in_id_type` is the type of ID returned by `in_id` and can be any of the values supported by `pyhetdex.het.fplane.FPlane.by_id()`.
- `out_id_type` is the type of ID to return and can be any of the ones supported by `pyhetdex.het.fplane.IFU`.

If the primary is `/path/to/dither_073.txt` and the fplane file contains the following IFU:

```
# IFUSLOT X_FP   Y_FP   SPECID SPECSLOT IFUID IFUROT PLATESC
073 150.0   150.0   04  136 023 0.0 1.0
```

the above configuration returns the value `'023'`

Similarly to the `header` keyword, by default the id is cast to a string. The `formatter` keyword can be used to the formatting of the output id. In the following example:

```
keyword:
    <<: *fplane_map
    out_id_type: specid
    formatter: '{:03d}'  # or '{0:03d}'
```

the return value is `'004'`. Without the `formatter` keyword the output would be `'4'`.

As in the previous cases, if `do_split` is present and `False`, the ids are extracted from all the primaries and converted; the resulting IDs are concatenated.

For information about the fplane parser, follow this link.

Built-in execute types

new_file

Following the instructions provided, this type builds a string and checks whether the file referenced by that string exists on the filesytem.

The only mandatory option is `value`: it is used to build the string from the primary and can be any of the available keyword types. E.g. given a primary like `/path/to/123T456_001LL_sci.fits`, the following instruction will create the string `/path/to/masterbias_001LL.fits` and check if it exists.

```
execute:  &execute
    type: new_file
    value:
        type: regex
        match: (.*?)/\d*?T\d*?_(\d{3}[LR][LU])_.*\.fits
        replace: \1/masterbias_\2.fits
```

In some cases it might not be possible to build the path of the output file directly from the primary files. In this case you can provide to the type definition the `path` keyword, whose return value is joined together with the name of the file returned by `value`. `path` can be either of the following:

- any of the available keyword types: the path is then constructed in the same way as `value`. E.g.:

  ```
  execute:
      <<: *execute
      path: /other/path
  ```

- a string like `$key`: this will get the value of path from the `key` keyword from the *command configuration*. The following behaves in the same way as the above example:

```
other_dir: /other/path
execute:
    <<: *execute
    path: $other_dir
```

1.5.4 Add new types

To any type, be it primary or not, there is a corresponding function that implements how to handle it.

All the types are implemented as plugins, discovered and dynamically loaded at run time.

The command interpreter looks for two entry points:

- `vdat.cit.primary`: for the definition of primary types
- `vdat.cit.keyword`: for the definition of other types
- `vdat.cit.execute`: for the definition of types to decide whether to execute or not the command

Each entry point is defined as a string, like:

```
type = package.module:func
```

where `type` is the name of the type and `func` is the function handling the keyword of `type`; `func` is implemented in the `module` module of the package `package`.

The functions implementing primary and secondary keywords have the following signature:

`vdat.command_interpreter.types.`**`primary_template`**(*target_dir*, *key_val*)

Template for a function that deals with a primary keyword.

It collects the files from the `target_dir` according to the instructions in `key_val`, if any and either `yield` a value or return an iterable.

Parameters

target_dir [string] directory in which the files must be collected

key_val [dictionary] configuration for the key handle

Yields

yield a string or iterable of strings

Raises

CIPrimaryError if something goes wrong when handling the primary key

`vdat.command_interpreter.types.`**`keyword_template`**(*primary*, *key_val*)

Template for a function that deals with a non-primary keyword.

A keyword has a value either statically stored in `key_val` or its value need to be extracted from the value of the primary file(s).

Parameters

primary [string] the value of one of the items returned by *`primary_template()`*

key_val [dictionary] configuration for the key handle

Returns

string value to associate to the keyword

Raises

CIKeywordError if something goes wrong when handling the key

`vdat.command_interpreter.types.execute_template`(*primary*, *config*)

For each of the primary entry, this function is called to decide whether to execute or skip the command.

Parameters

primary [string] the value of one of the items returned by `primary_template()`

config [dictionary] configuration for the command

Returns

bool `True`: the command is executed; `False`: the command is skipped

The `run` method

Invoking

`CommandInterpreter.run`()

Collect the files, expand and run the required command

All the custom errors raised here derive from `CIRunError`.

Raises

CICommandFmtError if the substitution of the command doesn't work

will:

1. collect all the `primary` files
2. filter them according to the list of selected items
3. loop over the `primary` files
4. check whether the step must be executed or not
5. for each step in the loop replace the relevant `fields` in the input command according to the instructions from the configuration
6. execute the command
7. report execution progress
8. collect and send out execution results

The execution of each step in the loop is done using the worker-based interface provided by `pyhetdex.tools.processes`. Within the command interpreter only the worker named `command_interpreter` is used. The multiprocessing is enabled, if the `multiprocessing` keyword argument is given.

1.5.5 Communication

The command interpreter communicates with the rest of the world through different channels.

- Upon errors directly handled by the interpreter, one of the errors defined in `vdat.command_interpreter.exceptions` is raised. Most of those errors are notified to the user via pop-up windows. Check the documentation of `CommandInterpreter` for more details.

- During normal execution of the command, the resolved command string, standard output, error and any exception raised while executing the code are logged to a logger with the name of the executable. In `VDAT`, these loggers are set to write to files located in the directory defined in the `VDAT` configuration file; the name of those files are the executable name with a `.log` extension. These loggers are set in the main `VDAT` code, not in the command interpreter sub-package.

  Warning: in a future release also the logging will be performed via the signal mechanism

- *`CommandInterpreter`* uses PyQt like signals to communicate with the external word.

  The names of the `emit` methods arguments are the type of the parameter followed by an underscore and optionally by an explanatory name.

  Available signals are:

  - `command_string`: accept an int and a string;

    `CICommandString.`**`emit`**(*int_*, *string_*)

    Parameters
    int_ [int] loop number
    string_ [string] string of the command

  - `progress`: accept four numbers, the total expected number, the number of done, of skipped and of failures;

    `CIProgress.`**`emit`**(*int_tot*, *int_done*, *int_skipped*, *int_fail*)

    Parameters
    int_tot [int] total number of jobs
    int_done [int] number of finished jobs; the number of successful jobs is `int_done - int_skipped - int_fail`
    int_skipped [int] number of skipped jobs
    int_fail [int] number of failed jobs

  - `command_done`: accept a boolean: `True` for the end of the command interpreter, `False` for the end of one single command;

    `CICommandDone.`**`emit`**(*bool_global*)

    Parameters
    bool_global [boolean] if `True`, the command interpreter is done, if `False` a single command is done

  - `global_logger`: accept an integer and a string

    `CIGlobalLogger.`**`emit`**(*int_level*, *string_msg*)

    Parameters
    int_level [integer] logging level; see the logging documentation for more information
    string_msg [string] string to log

  - `n_primaries`: accept an integer

    `CINPrimaries.`**`emit`**(*int_*)

    Parameters
    int_ [integer] number of primary files

  - `commands`: accept six strings and a dictionary. The first string is the primary value; the second the command with all the substitutions in place, if the execution finished, or with the placeholder, if some exception has been raised; the third and fourth are the stdout and stderr of the executed command; the

fifth is non empty if the command has a non-null return code; the sixth is non empty is the execution of the command crashes for some reason; the seventh is the configuration dictionary passed to the *CommandInterpreter*

CINPrimaries.**emit**(*int_*)

Parameters

int_ [integer] number of primary files

The list of available signal names can be retrieved with *vdat.command_interpreter.signals.get_signal_names()*, while the signals can be accessed with *vdat.command_interpreter.signals.get_signal()*. Callbacks can be connected to or disconnected from each signal using the methods *connect()* or *disconnect()*

1.6 Frequently asked questions

Q: I start `vdat` and I get an error finishing with:

```
vdat.config.core.ConfigurationError: The file 'vdat_setting.cfg' does not exist
```

A: VDAT needs some configuration file to run. You can retrieve them with the `vdat_config copy` command. See *Use VDAT: how to* for more information

---

Q: When running `vdat` I see the following warning:

```
UserWarning: 'CUREBIN' environment variable is not set. If the 'curebin'
option in the configuration file is set to interpolate from 'curebin_env',
all the vdat commands requiring cure binaries will fail.
```

What is `CUREBIN/cure` and why do I need it?

A: `cure` is a library of programs used to reduce HETDEX data. Most of the commands shipped with VDAT use some `cure` executable. CUREBIN is an environment variable pointing to the `cure` binaries. See *ltl and cure* for instructions.

Q: It tells me there's no raw or redux directory.

A: you probably refers to this error:

```
2016-06-01 11:47:39,101 CRITICAL Neither the raw directory '['/path/to/raw']' nor the␣
↪redux directory '/path/to/redux' exist. This is a problem. Aborting.
Traceback (most recent call last):
File "/[...]/bin/vdat", line 9, in <module>
    load_entry_point('vdat', 'gui_scripts', 'vdat')()
File "/[...]/vdat/libvdat/vdat.py", line 115, in main
    vdatlink.symlink("start symlinking the redux directory")
File "/[...]/vdat/libvdat/symlink.py", line 71, in symlink
    raise vutil.VDATDirError("The raw and redux directories cannot be"
vdat.utilities.VDATDirError: The raw and redux directories cannot be found: VDAT␣
↪cannot run.
```

VDAT needs data to run. If it's the first time you run it you need to specify one or more directories containing the raw data, as explained in *Use VDAT: how to*. Once you have ingested the data, VDAT reloads them and it is not necessary anymore to provide the raw directory.

Q Why I get "vdat: cannot connect to X server" or "DISPLAY NOT SET" when I run VDAT?

A: VDAT has a GUI based on Qt technologies and requires a working graphical environment. If you are running it on a remote computer via `ssh`, make sure that you enable X11 forwarding with the `-X` option.

Todo: Add some instruction about the loggers settings

CHAPTER 2

Developer documentation

2.1 Contribute to VDAT

2.1.1 How To

The suggested workflow for implementing bug fixes and/or new features is the following:

- Identify or, if necessary, add to our redmine issue tracker one or more issues to tackle. Multiple issues can be addressed together if they belong together. Assign the issues to yourself.
- Create a new branch from the trunk with a name either referring to the topic or the issue to solve. E.g. if you need to add a new executable, tracked by issue #1111 `do_something`:

```
svn cp ^/trunk ^/branches/do_something_1111\
-m 'create branch to solve issue #1111'
```

- Switch to the branch:

```
svn switch ^/branches/do_something_1111
```

- Implement the required changes and don't forget to track your progress on redmine. If the feature/bug fix requires a large amount of time, we suggest, when possible, to avoid one big commit at the end in favour of smaller commits. In this way, in case of breakages, is easier to traverse the branch history and find the offending code. For each commit you should add an entry in the `Changelog` file.

  If you work on multiple issues on the same branch, close one issue before proceeding to the next. When closing one issue is good habit to add in the description on the redmine the revision that resolves it.

- Every function or class added or modified should be adequately documented as described in *Coding style*.

  Documentation is essential both for users and for your fellow developers to understand the scope and signature of functions and classes. If a new module is added, it should be also added to the documentation in the appropriate place. See the existing documentation for examples.

  Each executable should be documented and its description should contain enough information and examples to allow users to easily run it.

- Every functionality should be thoroughly tested for python 2.7 and 3.4 or 3.5 in order to ensure that the code behaves as expected and that future modifications will not break existing functionalities. When fixing bugs, add tests to ensure that the bug will not repeat. For more information see *Testing*.
- Once the issue(s) are solved and the branch is ready, merge any pending change **from** the trunk:

  ```
  svn merge ^/trunk
  ```

  While doing the merge, you might be asked to manually resolve one or more conflicts. Once all the conflicts have been solved, commit the changes with a meaningful commit message, e.g.: `merge ^/trunk into ^/branches/do_something_1111`. Then rerun the test suite to make sure your changes do not break functionalities implemented while you were working on your branch.
- Then contact the maintainer of `vdat` and ask to merge your branch **back to the trunk**.

Information about branching and merging can be found in the svn book. For any questions or if you need support do not hesitate to contact the maintainer or the other developers.

2.1.2 Coding style

All the code should be compliant with the official python style guidelines described in **PEP 8**. To help you keep the code in spec, we suggest to install plugins that check the code for you, like Synstastic for vim or flycheck for Emacs.

The code should also be thoroughly documented using the numpy style. See the existing documentation for examples.

2.1.3 Testing

Note: Every part of the code should be tested and should run at least under python 2.7 and the latest two or three python 3.x releases (e.g. 3.4, 3.5 and 3.6)

`vdat` uses the testing framework provided by the pytest package. The tests should cover every aspect of a function or method. If exceptions are explicitly raised, this should also tested to ensure that the implementation behaves as expected.

The preferred way to run the tests is using tox, an automatised test help package. If you have installed tox, with e.g. `pip install tox`, you can run it by typing:

```
tox
```

It will take care of creating virtual environments for every supported version of python, if it exists on the system, install `vdat`, its dependences and the packages necessary to run the tests and runs `py.test`

You can run the tests for a specific python version using:

```
py.test
```

or:

```
python setup.py test
```

The latter command fetches all the needed dependences, among others `pytest` itself, will be fetched and installed in a `.eggs` directory. Then it will run `py.test`. This command might fail when running in a virtual environment. If you get `ImportError: No module named 'numpy'` while installing `scipy`, install numpy by hand `pip install [--user] numpy` and rerun it again. Use the option `--addopts` to pass additional options to `py.test`.

The VDAT test suites depends on some large file that is not shipped with the main repository. You can get the file running the following command in the parent directory of the `vdat` root:

```
svn checkout svn://luna.mpe.mpg.de/vdat_test_data/trunk vdat_test_data
```

If for any reason you have the test data somewhere else, you can tell `pytest` where they are using the `--extra-data-dir` option. If the directory is not found, `pytest` will abort with a useful error message.

You can run specific tests providing the file name(s) and, optionally the name of a test. E.g.:

```
py.test tests/test_libvdat/test_symlink.py  # runs only the tests in one file
py.test tests/test_libvdat/test_symlink.py::test_no_raw  # runs only one test function
```

Relevant command line options:

```
-v                    verbose output: print the names and parameters of the
                      tests
-s                    capture standard output: can cause weird interactions
                      with the logging module
--noremove            Do not remove output created output files when
                      tearing down the tests. Useful to inspect the log
                      files created by vhc when updating the tests
--extra-data-dir=EXTRA_DATA_DIR
                      Directory containing extra test data.
```

A code coverage report is also created thanks to the pytest-cov plugin and can be visualized opening into a browser `cover/index.html`. If you want a recap of the coverage directly in the terminal you can provide one of the following options when running `py.test`:

```
--cov-report term
--cov-report term-missing
```

Besides running the tests, the `tox` command also builds, by default, the documentation and collates the coverage tests from the various python interpreters and can copy then to some directory. To do the latter create, if necessary, the configuration file `~/.config/little_deploy.cfg` and add to it a section called `vdat` with either one or both of the following options:

```
[vdat]
# if given the deploys the documentation to the given dir
doc = /path/to/dir
# if given the deploys the coverage report to the given dir
cover = /path/to/other/dir

# it's also possible to insert the project name and the type of the document
# to deploy using the {project} and {type_} placeholders. E.g
# cover = /path/to/dir/{project}_{type_}
# will be expanded to /path/to/dir/vdat_cover
```

For more information about the configuration file check little_deploy.

For other command line arguments type:

```
py.test -h
```

For a list of available fixtures type:

```
py.test --fixtures tests/
```

Qt bindings

With version 0.9.0 and the use of `qtpy`, VDAT can be run against different `Qt` bindings. Ideally we would like to run VDAT against all the possible. When writing (2018/06/15) the `tox` configuration file contains hints about this ideal; I would have liked to run the following tests:

- py27: `PyQt4`
- py34: `PySide`
- py35: `PySide2`
- py36: `PyQt5`

Unfortunately this didn't work out so the actual test matrix is:

- py27: `PyQt4`
- py34: `PyQt4`
- py35: `PyQt5`
- py36: `PyQt5`

If any of `PySide`/`PySide2` becomes supported, then the test matrix should be reverted to

A note on `PySide`

`PySide` can be installed using `pip`, however it needs python to be compiled with the `-fPIC` flag. If this is not the case, you might get the following error while installing `PySide`:

```
/usr/bin/ld:
/home/montefra/.pyenv/versions/3.4.6/lib/libpython3.4m.a(abstract.o):
relocation R_X86_64_32S against `_Py_NotImplementedStruct' can not be used
when making a shared object; recompile with -fPIC
/home/montefra/.pyenv/versions/3.4.6/lib/libpython3.4m.a: error adding
symbols: Bad value
collect2: error: ld returned 1 exit status
libshiboken/CMakeFiles/libshiboken.dir/build.make:381: recipe for
target 'libshiboken/libshiboken.cpython-34m.so.1.2.4' failed
make[2]: *** [libshiboken/libshiboken.cpython-34m.so.1.2.4]
Error 1
CMakeFiles/Makefile2:204: recipe for target
'libshiboken/CMakeFiles/libshiboken.dir/all' failed
make[1]: ***
[libshiboken/CMakeFiles/libshiboken.dir/all] Error 2
Makefile:127: recipe for target 'all' failed
make: *** [all] Error 2
error: Error compiling shiboken
```

If you are using `pyenv` to handle your python versions, you can get a new python version with the correct flag using:

> PYTHON_CFLAGS=-fPIC pyenv install -v 3.4.8

2.1.4 Documentation

To build the documentation you need the additional dependences described in *Optional dependences*. They can be installed by hand or during `vdat` installation by executing one of the following commands on a local copy:

```
pip install /path/to/vdat[qt5,doc]
pip install /path/to/vdat[qt5,livedoc]
```

The first install `sip`, `pyqt5`, `sphinx`, the `alabaster` theme and the `numpydoc` extension; the second also installs `sphinx-autobuild`.

To build the documentation in html format go to the `doc` directory and run:

```
make html
```

The output is saved in `_doc/build/html`. For the full list of available targets type `make help`.

If you are updating the documentation and want avoid the `edit-compile-browser refresh` cycle, and you have installed `sphinx-autobuild`, type:

```
make livehtml
```

This command compiles the documentation and serves it on http://127.0.0.1:{port}, where {port} is an available port, and open the page on your default browser. The html documentation is automatically rebuilt after every change of the source and the browser reloaded.

Please make sure that every module in `vdat` is present in the *Code Documentation*.

Qt Documenation

We distribute this documentation also with VDAT, to allow consulting it offline.

The Qt documentation can be built by sphinx itself using the following command:

```
make qthelp
```

This command creates the documentation into the `_build/qthelp` directory together with these extra files:

- Qt Help Collection Project (`.qhcp`)
- Qt Help Project (`.qhp`)
- Qt Compressed Help (`.qch`)
- Qt Help Collection (`.qhc`)

See the Qt Help documentation for more information about those files.

All those files can also be created running `tox`, or `tox -e qt-doc`. In this case the files can be found in `.tox/qt-doc/tmp/qthelp`

To update the documentation copy the `.qch` and `.qhc` files into `vdat/gui/static`. If the files have a different names from the old ones, remove the latter from the repository with `svn rm` and add the new ones with `svn add`. Don't forget to commit to propagate the changes.

2.2 Code Documentation

2.2.1 `command_interpreter` – The command line interpreter

`command_interpreter.core` – The core functionality

This module implements the core of the command interpreter and any part essential for running it

class vdat.command_interpreter.core.**CommandInterpreter**(*command*, *command_config*, *selected=None*, *multiprocessing=False*, *processes=None*)

Interpret and execute the command.

See *The interpreter* section in the documentation for more details

All the custom errors are defined in `vdat.command_interpreter.exceptions`. The ones raised in the constructor are derived from from `CIValidationError`,

Parameters

command [string] command to parse

command_config [dict] dictionary containing the instructions to execute the `command`. A deep copy is executed

selected [list-like, optional] None or a list of the selected items; if `None` no filtering of the primary files is done; otherwise must be an object supporting the membership test operator `in`.

multiprocessing [bool] run the command using multiprocessing

processes [int] number of processors to use

Raises

CINoExeError if the executable does not exists

CIParseError if there is some error when extracting the keywords

CIKeywordError for missing keywords or for keywords of wrong type

CIKeywordTypeError if the type of the keywords is not among the known ones

make_signals()

Get the signals from the `signals` and save them in attributes with the same names. Reimplement this method to use custom signals. Refers to the `signals` documentation for the list and names of signals to implement

run()

Collect the files, expand and run the required command

All the custom errors raised here derive from `CIRunError`.

Raises

CICommandFmtError if the substitution of the command doesn't work

_replace_alias()

If the command configuration has the `is_alias_of` replace the executable name

_check_exe()

Check that the executable can be found and replace it with the full path returned by `distutils.spawn.find_executable()`

_get_keys(*command*)

Extract the keywords from the command

_validate_mandatory()

Check that all the mandatory keywords are provided.

If `mandatory` is not found, return without doing anything

_validate_primary()

Check that the primary key is present and that, if it has more that one value, only one primary key is present among the command keys

_validate_keywords()

Check that all the requested keywords are in the configuration

_replace_primary()

If the value of `primary` is a string, does nothing. Otherwise find which of the primaries is used in the command and replace "self.config['primary'] with it.

Since *_validate_primary* already checks the primaries, this function can assume that the primary is used only once in the command.

_key_types(*keys*)

Scan the keys and check that the interpreter knows how to deal with them.

Parameters

keys [list of strings] keys extracted from the command

Returns

primary_func [callable] function to call to get the list of primary files

keyword_map [dictionary] map non-primary keywords to the function used to handle them

Raises

CIKeywordTypeError if the type of the keyword is not known

_get_value_as_dict(*key*)

Get the value of `key` from the configuration. If it's a string, convert it to a dictionary with two entries:

- type: `plain`
- value: `value`

and re-add it in the configuration dictionary

Parameters

key [string] key to get

Returns

value [dictionary] dictionary defining the type

Raises

CIKeywordError if `value` is not a dictionary or a string

_filter_selected()

Look for the existence of the `filter_selected` option and check that it is of the correct type and that `selected` is of the correct type

Returns

filter_func [function] function that accepts one string (one element of the primary list) and returns `True` or `False` if that element is valid or not

_execute()

Look for the existence of the `execute` option in the configuration and check that it is of the correct type.

Returns

execute_func [function] function that accepts one string (one element of the primary list) and returns `True` or `False` if that element must be run or not

true(*, **__)
returns always true

command_interpreter.types – The keyword types

Define enumerate-like classes that allows to map from keys to key types and to the functions that needs to be called to deal with any of them.

It uses pkg_resources and entry points to make the framework extendible

vdat.command_interpreter.types.**_load_entrypoints**(*group*)
Get all the entry points for the `group` and load them.

Parameters

group [string] name of the group to load

Returns

entry_points [dictionary] key: name; value: callable loaded from the entry point

class vdat.command_interpreter.types.**_Types**
Bases: `object`

ABC class for the types.

If a type `loop` exists, it can be accessed as `instance.loop` or `instance['loop']`

Attributes

known_types [list of strings] list of known types

entry_point_group [string] Abstract property with the name of the group to load

entry_point_group
Abstract property with the name of the group to load

known_types
list of known types

class vdat.command_interpreter.types.**PrimaryTypes**
Bases: *vdat.command_interpreter.types._Types*

Fill the type<–>function mapping using the `vdat.cit.primary` entry point.

entry_point_group
Abstract property with the name of the group to load

class vdat.command_interpreter.types.**KeywordTypes**
Bases: *vdat.command_interpreter.types._Types*

Fill the type<–>function mapping using the `vdat.cit.keyword` entry point.

entry_point_group
Abstract property with the name of the group to load

class vdat.command_interpreter.types.**ExecuteTypes**
Bases: *vdat.command_interpreter.types._Types*

Fill the type<–>function mapping using the `vdat.cit.execute` entry point.

entry_point_group
Abstract property with the name of the group to load

`vdat.command_interpreter.types.`**`primary_template`**(*target_dir*, *key_val*)
Template for a function that deals with a primary keyword.

It collects the files from the `target_dir` according to the instructions in `key_val`, if any and either `yield` a value or return an iterable.

Parameters

target_dir [string] directory in which the files must be collected

key_val [dictionary] configuration for the key handle

Yields

yield a string or iterable of strings

Raises

CIPrimaryError if something goes wrong when handling the primary key

`vdat.command_interpreter.types.`**`keyword_template`**(*primary*, *key_val*)
Template for a function that deals with a non-primary keyword.

A keyword has a value either statically stored in `key_val` or its value need to be extracted from the value of the primary file(s).

Parameters

primary [string] the value of one of the items returned by *`primary_template()`*

key_val [dictionary] configuration for the key handle

Returns

string value to associate to the keyword

Raises

CIKeywordError if something goes wrong when handling the key

`vdat.command_interpreter.types.`**`execute_template`**(*primary*, *config*)
For each of the primary entry, this function is called to decide whether to execute or skip the command.

Parameters

primary [string] the value of one of the items returned by *`primary_template()`*

config [dictionary] configuration for the command

Returns

bool `True`: the command is executed; `False`: the command is skipped

`vdat.command_interpreter.types.`**`primary_plain`**(*target_dir*, *key_val*)
Get all the files in `target_dir` matching the string in `key_val['value']`

Parameters

target_dir [string] directory in which the files must be collected

key_val [dictionary] configuration for the key handle

Returns

iterator yields file names matching the value recursively

`vdat.command_interpreter.types.`**`primary_loop`**(*target_dir*, *key_val*)

Make a nested loop over the set of given keys, in each step of the loop construct the value using python format string syntax and then get all the files matching it.

If the key `returns` is found, the output string is manipulated according to the instruction in the value of `returns`. The type of returns can be any available keyword type.

If any of the steps doesn't produce any file, no value is yielded.

Parameters

target_dir [string] directory in which the files must be collected

key_val [dictionary] configuration for the key handle

Yields

string of space separated file names

`vdat.command_interpreter.types.`**`primary_groupby`**(*target_dir*, *key_val*)

Loop over all the files matching the `value` entry. For each one, create a list of file names replacing the regex in `pattern` with the elements of `replace`.

Parameters

target_dir [string] directory in which the files must be collected

key_val [dictionary] configuration for the key handle

Yields

string of space separated file names

`vdat.command_interpreter.types.`**`primary_all_files`**(*target_dir*, *key_val*)

Get all the files in `target_dir` matching the string in `key_val['value']` and returns all the files as a single string, so that they can be used all at once in a command.

This primary type relies on *`primary_plain()`* to collect all the files or values.

Parameters

target_dir [string] directory in which the files must be collected

key_val [dictionary] configuration for the key handle

Returns

files [list of one element] space separated list of file names or return values.

`vdat.command_interpreter.types.`**`keyword_plain`**(_, *key_val*)

Returns the value contained in the keyword

Parameters

primary [string] ignored

key_val [dictionary] configuration for the key handle

Returns

string value to associate to the keyword

`vdat.command_interpreter.types.`**`keyword_regex`**(*primary*, *key_val*)

Extract a string from the primary using regular expression substitution. If `do_split` is False (default `True`), do not split the primary on white spaces and use only the first entry.

After performing the substitution, it checks that the expected number of substitutions is performed; the number is given by the option `n_subs` (default 1), with the following meaning:

- negative: no check performed
- positive integer: exactly `n_subs` must be performed
- list of integers: the number of substitutions must be in `n_subs`
- string: interpreted a `[start]:[stop][:step]` or `[start],[stop][,step]` and is used to initialise *`vdat.command_interpreter.utils.SliceLike`*; the number of substitutions must be in `n_subs`, as defined by the above class.

Parameters

primary [string] primary file name(s)

key_val [dictionary] configuration for the key handle

Returns

string string built from the primary file name

Raises

CIKeywordError if the number of substitutions is not the expected or the value of the `n_subs` key is not correct

`vdat.command_interpreter.types.`**`keyword_header`**(*primary*, *key_val*)

Extract and parse an fits header keyword from the first file.

Extract the `value` keyword from the header.

If `formatter` is not given, cast the value to a string, otherwise use convert it to a string using the give formatter; e.g. `"{0:03d}"` assumes that the value is an integer and converts it into a zero padded-three digits string; see format string syntax

If `extract` is in the configuration, it instruct how to build a variable out of the extracted header value using the machinery from *`keyword_regex()`*.

If `do_split` keyword is given and is `False`, the `value` is extracted from the header of every file, converted to a string and all the values are concatenated with white spaces.

Parameters

primary [string] primary file name(s)

key_val [dictionary] configuration for the key handle

Returns

string value to associate to the keyword

`vdat.command_interpreter.types.`**`keyword_format`**(*primary*, *key_val*)

Create a new string formatting `value` according to the provided `keys`.

The keys are substituted using format string syntax.

The value of `keys` is a map between values to substitute in `value` and keyword types used to extract them from the primary file name. Strings are interpreted as of type `plain`.

Parameters

primary [string] primary file name(s)

key_val [dictionary] configuration for the key handle

Returns

string value to associate to the keyword

`vdat.command_interpreter.types.`**`keyword_fplane_map`**(*primary*, *key_val*)

Create a new ID from the one extracted from `primary` using the fplane file for the mapping.

For informations about the fplane file parser and the internals, specifically the type of the IDs, see `pyhetdex.het.fplane.FPlane` and `pyhetdex.het.fplane.IFU`.

> **Warning:** currently the `fplane_file` value is a string. If we need more flexibility, we can very easily modify it to act like the `in_id` key.

Parameters

primary [string] primary file name(s)

key_val [dictionary] configuration for the key handle

Returns

string value to associate to the keyword

`vdat.command_interpreter.types.`**`execute_new_file`**(*primary*, *config*)

Given the instructions, the `new_file` type constructs, for each primary entry, a string and check if it exists on the file system as a file. If the string is a new file, returns `True`.

The instruction on how to build the string are encoded in the mandatory key `value`, whose value can be any of the available keyword types.

If the path to the file cannot be easily extracted from the primary itself, it is possible to build it using the `path` optional key. If `path` is present, the value of `path` and the basename from `value` are joined. `path` can be either one of the available keyword types or a `$identifier`, where `identifier` is an other key in the command configuration (**not** the `execute` configuration).

Parameters

primary [string] the value of one of the items returned by *`primary_template()`*

config [dictionary] configuration for the command (not for the type)

Returns

bool `True`: if the output of the keyword handling does not exist

`vdat.command_interpreter.types.`**`value_to_dict`**(*value*)

If it's a string, convert it to a dictionary with two entries:

- type: `plain`
- value: `value`

And also check that the `type` entry is in value

Parameters

value [string or dict] value to check

Returns

value [dictionary] dictionary defining the type

Raises

CIKeywordError if `value` is not a dictionary or a string

command_interpreter.utils – Utilities

Utility functionalities

vdat.command_interpreter.utils.**id_**(**args*)

Identity function: returns the input unchanged

If the input is one element only, extract it from `args`

Examples

```
>>> id_()
()
>>> id_(5)
5
>>> id_(3, 4)
(3, 4)
>>> a, b, c = id_(2, 3, 4)
>>> print(a, b, c)
2 3 4
```

vdat.command_interpreter.utils.**flip**(*func*)

Flip the order of the arguments in the function

Can also be used a decorator

Parameters

func [callable] function to replace

Returns

callable

vdat.command_interpreter.utils.**multiwrap**(**decorators*)

Create a decorator combining the given decorators

class vdat.command_interpreter.utils.**SliceLike**(*str_or_int*, **args*)

A slice like object, but it's not usable as `slice`.

If the input argument is a string of `:` or `,` separated integers parse it assuming `[start]:[stop][:step]`. `start`, `stop`, and `step` must be integers or `None`; the latter is optional and cannot be zero.

It's possible to use the `in` statement to check if a value is in the slice with the following logic:

- if `start` is not given, it checks that the value is less than `stop`
- if `stop` is not given, it checks that the value is greater or equal than `start`
- if neither `start` not `stop` are given and `step` is not given or one, returns `True`
- if `step` is not 1, it checks that the value is equal to `start + n*step` with integer `n`.
- if `start` is not given and `step` is not 1, returns `False`; undefined behaviour.

Parameters

str_or_int, args [string or one/two/three integers] if the first argument is a string is parsed as if it's a `:` or `,` separated one and all the elements are converted to None (if empty) or to integers. In this case no other arguments are allowed. If the arguments are integers are interpreted as:

```
SliceLike(stop)
SliceLike(start, stop[, step])
```

Raises

CISliceError if the initialisation fails

Attributes

start Start of the slice (read only)

stop Stop of the slice (read only)

step Step of the slice (read only)

start
Start of the slice (read only)

stop
Stop of the slice (read only)

step
Step of the slice (read only)

range()
Create and returns a range using the elements of the slice.

If *start* is `None`, set it to 0, if *stop* is `None`, it fails and when *step* is `None` set it to 1.

Returns

a range or a xrange object (python 3 / python 2)

vdat.command_interpreter.utils.**abstractproperty**(*func*)

- Python 3: decorator from combining `property` and `abc.abstractmethod()`
- Python 2: alias of `abc.abstractproperty()`

command_interpreter.exceptions – Custom exceptions

Command interpreter exceptions

exception vdat.command_interpreter.exceptions.**CIError**
Bases: `Exception`

Generic exception. It's the parent of all the other exceptions defined here

exception vdat.command_interpreter.exceptions.**CIValidationError**
Bases: *vdat.command_interpreter.exceptions.CIError*

Exception raised when validating the command in the constructor

exception vdat.command_interpreter.exceptions.**CINoExeError**(*name*)
Bases: *vdat.command_interpreter.exceptions.CIValidationError*

Raised when the executable `name` is not found

exception vdat.command_interpreter.exceptions.**CIParseError**
Bases: *vdat.command_interpreter.exceptions.CIValidationError*

Failed parsing of the command

exception vdat.command_interpreter.exceptions.**CIKeywordValidationError**
Bases: *vdat.command_interpreter.exceptions.CIValidationError*

Raised when the keyword validation fails

exception vdat.command_interpreter.exceptions.**CIKeywordTypeError**
Bases: *vdat.command_interpreter.exceptions.CIKeywordValidationError*

Raised when the keyword doesn't have a `type` key or its type is not known

exception vdat.command_interpreter.exceptions.**CIRunError**
Bases: *vdat.command_interpreter.exceptions.CIError*

Raised when running the command

exception vdat.command_interpreter.exceptions.**CIPrimaryError**
Bases: *vdat.command_interpreter.exceptions.CIRunError*

Raised if something bad happens when handling a primary keyword

exception vdat.command_interpreter.exceptions.**CIKeywordError**
Bases: *vdat.command_interpreter.exceptions.CIRunError*

Raised if something bad happens when handling a secondary keyword

exception vdat.command_interpreter.exceptions.**CICommandFmtError**
Bases: *vdat.command_interpreter.exceptions.CIRunError*

Raised when the replacement of the keyword fails

exception vdat.command_interpreter.exceptions.**CISubprocessError**
Bases: *vdat.command_interpreter.exceptions.CIRunError*

Error raised when the start of the subprocess or the communication with it fails. It's not related with the execution of the underlying command

exception vdat.command_interpreter.exceptions.**CISliceError**
Bases: *vdat.command_interpreter.exceptions.CIError*, `ValueError`

Errors raised while initialising the command_interpreter.utils.SliceLike class

The interpreter communication channels

`command_interpreter.signals` – Signals

`CommandInterpreter` uses PyQt like signals to communicate with the external word.

The names of the `emit` methods arguments are the type of the parameter followed by an underscore and optionally by an explanatory name.

Available signals are:

- `command_string`: accept an int and a string;

  `CICommandString.`**`emit`**(*int_*, *string_*)

  Parameters

  int_ [int] loop number

  string_ [string] string of the command

- `progress`: accept four numbers, the total expected number, the number of done, of skipped and of failures;

  `CIProgress.`**`emit`**(*int_tot*, *int_done*, *int_skipped*, *int_fail*)

Parameters

int_tot [int] total number of jobs

int_done [int] number of finished jobs; the number of successful jobs is `int_done - int_skipped - int_fail`

int_skipped [int] number of skipped jobs

int_fail [int] number of failed jobs

- `command_done`: accept a boolean: `True` for the end of the command interpreter, `False` for the end of one single command;

  `CICommandDone.`**`emit`**(*bool_global*)

  Parameters

  bool_global [boolean] if `True`, the command interpreter is done, if `False` a single command is done

- `global_logger`: accept an integer and a string

  `CIGlobalLogger.`**`emit`**(*int_level*, *string_msg*)

  Parameters

  int_level [integer] logging level; see the logging documentation for more information

  string_msg [string] string to log

- `n_primaries`: accept an integer

  `CINPrimaries.`**`emit`**(*int_*)

  Parameters

  int_ [integer] number of primary files

- `commands`: accept six strings and a dictionary. The first string is the primary value; the second the command with all the substitutions in place, if the execution finished, or with the placeholder, if some exception has been raised; the third and fourth are the stdout and stderr of the executed command; the fifth is non empty if the command has a non-null return code; the sixth is non empty is the execution of the command crashes for some reason; the seventh is the configuration dictionary passed to the `CommandInterpreter`

  `CINPrimaries.`**`emit`**(*int_*)

  Parameters

  int_ [integer] number of primary files

`vdat.command_interpreter.signals.`**`register`**(*name*)

Initialise the decorated class and save it into the signals dictionary with `name` as key

Parameters

name [string] name under which the class or instance must be registered

Returns

decorator [function] class decorator

`class` `vdat.command_interpreter.signals.`**`BaseCISignal`**

Bases: `object`

Base implementation for the signals used in the `CommandInterpreter`

Attributes

connected [list] list of the connected callables

connected
list of the connected callables

connect(*callable_*)
Connect the callable with the signal

Parameters

callable_ : function/method/callable instance to connect with the signal

disconnect(*callable_*)
Remove the first instance of callable from the signal

Parameters

callable_ : function/method/callable instance to disconnect from the signal

Raises

ValueError if the callable is not already connected

clear()
Remove all the associated signals

emit(**args*, ***kwargs*)
Abstract method that emit the signal, executing all the registered callables.

The default implementation loop through the *connected* and call them. Reimplement to give it a proper signature.

class vdat.command_interpreter.signals.**CICommandString**
Bases: *vdat.command_interpreter.signals.BaseCISignal*

Emit the string of the command to execute after substituting the keywords

emit(*int_*, *string_*)

Parameters

int_ [int] loop number

string_ [string] string of the command

class vdat.command_interpreter.signals.**CIProgress**
Bases: *vdat.command_interpreter.signals.BaseCISignal*

Emit informations about execution progress

emit(*int_tot*, *int_done*, *int_skipped*, *int_fail*)

Parameters

int_tot [int] total number of jobs

int_done [int] number of finished jobs; the number of successful jobs is `int_done - int_skipped - int_fail`

int_skipped [int] number of skipped jobs

int_fail [int] number of failed jobs

class vdat.command_interpreter.signals.**CICommandDone**
Bases: *vdat.command_interpreter.signals.BaseCISignal*

Signal emitted when a command has been executed or the run of the interpreter is done

emit(*bool_global*)

Parameters

bool_global [boolean] if `True`, the command interpreter is done, if `False` a single command is done

class vdat.command_interpreter.signals.**CIGlobalLogger**

Bases: *vdat.command_interpreter.signals.BaseCISignal*

Log information about the command interpreter execution. This channel is not used to log the execution of the commands.

emit(*int_level*, *string_msg*)

Parameters

int_level [integer] logging level; see the logging documentation for more information

string_msg [string] string to log

class vdat.command_interpreter.signals.**CINPrimaries**

Bases: *vdat.command_interpreter.signals.BaseCISignal*

Emit the number of primary files collected

emit(*int_*)

Parameters

int_ [integer] number of primary files

class vdat.command_interpreter.signals.**CICommandLogger**

Bases: *vdat.command_interpreter.signals.BaseCISignal*

For each primary emits the primary name(s), the actual command string executed in the subprocess, the log messages and the full configuration dictionary.

emit(*string_primary*, *string_command*, *string_info*, *string_warning*, *string_error*, *string_critical*, *dict_config*)

Parameters

int_ [integer] number of primary files

vdat.command_interpreter.signals.**get_signal**(*name*)

Get the signal associated with `name`

Parameters

name [string] name of the requested relay

Returns

one of the registered signal instances

vdat.command_interpreter.signals.**get_signal_names**()

Get the full list of signals

`command_interpreter.helpers` – Helper functionalities

Helper function and classes.

These functionalities are not essential for the interpreter, but can help the to setup things.

`vdat.command_interpreter.helpers.print_first`(*int_*, *string_*)
Print `string_` when `int_` is zero.

Can be connected to *vdat.command_interpreter.signals.CICommandString*

`vdat.command_interpreter.helpers.print_progress`(*int_tot*, *int_done*, *int_skipped*, *int_fail*)
Print the percentages of finished, successful and failed jobs, overwriting the line. Skipped are counted as successful.

Can be connected to *vdat.command_interpreter.signals.CIProgress*

`vdat.command_interpreter.helpers.print_done`(*bool_cidone*)
Function that prints `command done` when `bool_cidone == False` and `execution done` otherwise

Can be connected to *vdat.command_interpreter.signals.CICommandDone*

`vdat.command_interpreter.helpers.print_global_logger`(*int_level*, *string_msg*)
Function that prints `levelname: string_msg`. The conversion is done

Can be connected to *vdat.command_interpreter.signals.CIGlobalLogger*

`vdat.command_interpreter.helpers.print_n_primaries`(*int_*)
Function that prints `Number of primaries: int_`.

Can be connected to *vdat.command_interpreter.signals.CINPrimaries*

`vdat.command_interpreter.helpers.log_command_logger`(*string_primary*, *string_command*, *string_info*, *string_warning*, *string_error*, *string_critical*, *dict_config*)
Function that log the command, and the strings_{info,warning,error,critical} to a logger called with the name of the executable, from the first element of the string command.

Can be connected to *vdat.command_interpreter.signals.CICommandLogger*

2.2.2 `vdat.libvdat` – Stuff that isn't GUI

Symlinking

Symlink raw files into a redux directory

Multiprocessing is disabled for the following reasons:

- Calibration and science frames need locking to correctly deal with grouping and renaming if multiple shots have the same name from same objects;
- a lot of `peewee.OperationalError: database is locked` errors are raised
- symlinking is going to run only now and then and is always going to be much faster that any of the reduction steps
- if the symlink is run from the gui, there is no risk that interferes with reduction steps running

`vdat.libvdat.symlink.FMT_DATE_DIR = '%Y%m%d_%H%M%S'`
Format for converting a `datetime` instance into as string used as directory name

`vdat.libvdat.symlink.symlink`(*log_message*)
Symlink shots in the directory 'path' to an output redux directory. Format of redux directory follows *issue 820 <https://luna.mpe.mpg.de/redmine/issues/820>*

Parameters

log_message [string] Message to log as info before starting

`vdat.libvdat.symlink.`**`_scan_dirs`**(*redux_dir*)

Scan the redux directories and fill the database with entries from existing 'shot_name.txt' and 'exposure_names.txt' files. It also updates the 'shot_name.txt' file if the redux directory has changed.

Parameters

redux_dir [string] name of the redux directory

Returns

int, int Number of VDATDir and VDATExposures entries added to the database

`vdat.libvdat.symlink.`**`do_symlink`**(*raw_dir*, *redux_dir*)

Run the symlinking from the raw to the redux directory

Parameters

raw_dir [list of strings] name of the raw or the night directories

redux_dir [string] name of the raw and the redux directory

`vdat.libvdat.symlink.`**`_symlink_shot`**(*shot_dir*, *redux_dir*)

Create the redux directory for the shot and symlink all the files from the shot directory

Parameters

shot_dir [string] name of the shot directory

redux_dir [string] name of the directory where the new directory and symlink must go

Raises

:class:'~vdat.utilities.VDATFitsTypeError' if the image type or object are not consistent or unknown

`vdat.libvdat.symlink.`**`_save_exposures`**(*shot_dir*, *fits_files*, *vdat_dir*)

Give the list of files belonging to `shot` and the VDATDir database entry `vdat_dir`, takes the first file in each exposure and create the necessary `VDATExposures` entries.

Parameters

shot_dir [string] name of the shot directory

fits_files [list of strings] symlinked files

vdat_dir [`vdat.database.VDATDir` instance] database entry related to the input files

`vdat.libvdat.symlink.`**`_get_imagetype_datetime`**(*conf*, *log*, *shot_dir*, *fnames*)

Extract the image type and the date from the files

Parameters

conf: configuration object can be either a `ConfigParser` or a dictionary(-like) object

log [logging object]

shot_dir [string] name of the shot directory

fnames [list] file names to scan

Returns

image_type [string] type of the images

avg_date [`datetime`] average timestamp

avg_date_string [string] string representation of the average time stamp

Raises

VDATFitsParseError if it fails parsing the file names

VDATFitsTypeError if there are more than one image types

`vdat.libvdat.symlink.`**`_save_and_symlink`**(*log*, *fits_files*, *vdat_dir*, *write_shot_file*, *append_to_shot_file*, *vdat_dir_shot_file=None*)

Fail safe symlinking:

- add `vdat_dir` to the database; if it fails, abort
- symlink the `fits_files` to `vdat_dir.path`; if it fails removes the symlinked files and `vdat_dir` from the database
- add `vdat_dir` to the shot file; if it fails undo the previous steps

Parameters

log [logging object]

fits_files [list of strings] as in the input

vdat_dir [`vdat.database.VDATDir` instance] database entry to save

write_shot_file, append_to_shot_file [bool] whether the entry needs to be written into the shot file and, in such case, whether it must be appended

vdat_dir_shot_file [`vdat.database.VDATDir` instance] if given this entry, not `vdat_dir`, is added to the shot file

`vdat.libvdat.symlink.`**`_get_unique_keyword`**(*fits_files*, *keyword*, *shot*)

Get the header `keyword` from all the fits files and check that only one exists.

Parameters

fits_files [list of strings] names of the fits files

keyword [string] name of the keyword to extract

shot [string] name of the shot

Returns

string value of the header keyword

Raises

VDATFitsParseError if the header keyword does not exist

VDATFitsTypeError if there are more than one values for the header keywords

`vdat.libvdat.symlink.`**`_symlink_sci`**(*fits_files*, *vdat_dir*)

Symlink the science shots

Parameters

fits_files [list of strings] names of the fits files

vdat_dir [`vdat.database.VDATDir` instance] contains all the relevant information for the symlinking

Returns

out_vdat_dir [`vdat.database.VDATDir` instance] database entry to representing the current data

Raises

VDATSymlinkError if there is a mismatch between objects and shots (e.g. the same shot in the same night has a different object name)

`vdat.libvdat.symlink.`**`_symlink_zro`**(*fits_files*, *vdat_dir*)

Symlink the bias shots.

Parameters

fits_files [list of strings] names of the fits files

vdat_dir [`vdat.database.VDATDir` instance] contains all the relevant information for the symlinking

Returns

out_vdat_dir [`vdat.database.VDATDir` instance] database entry to representing the current data

`vdat.libvdat.symlink.`**`_symlink_cal`**(*fits_files*, *vdat_dir*)

Symlink the calibration, flat and arc, shots.

Flats and arcs taken together goes into the same directory.

1. If the shot is already symlinked, reuse the directory
2. **If not look for directories with different original type or object,** order them at increasing time distance and take the nearest one: if it's within a maximum time distance, symlink into that directory
3. Otherwise create a new directory and symlink into it

Parameters

fits_files [list of strings] names of the fits files

vdat_dir [`vdat.database.VDATDir` instance] contains all the relevant information for the symlinking

Returns

out_vdat_dir [`vdat.database.VDATDir` instance] database entry to representing the current data

`vdat.libvdat.symlink.`**`_mkdir`**(*dirname*, *log*, *failsafe=True*)

Create the directory.

If it exists, log it as error and, if `failsafe` is False, re-raise the exception

Parameters

dirname [string] name of the directory to create

log [`logging.Logging` instance] log messages to this logger

safe [bool, optional] if true silently ignores `OSError` due to existing directories

Raises

:class:'~vdat.utilities.VDATDirError' if the creation fails with a

:class:'OSError' and ''failsafe'' is False

`vdat.libvdat.symlink.`**`_symlink_file`**(*file_list*, *target_dir*, *log*, *failsafe=True*)

Symlink the files into the target directory.

If it exists, log it as error and, if `failsafe` is False, re-raise the exception

Parameters

file_list [list of strings] names of the fits files to symlink

target_dir [string] name of the directory where to do the symlink

log [`logging.Logging` instance] log messages to this logger

safe [bool, optional] if true ignores `OSError` due to existing files

Returns

symlinked_list [list of strings] list of files in the target directory successfully symlinked

Raises

:class:'~vdat.utilities.VDATSymlinkError' if the symlink creation fails

with a :class:'OSError' and ''failsafe'' is False

`vdat.libvdat.symlink.`**`_average_timestamps`**(*dates*, *infmt*, *outfmt='%Y%m%d_%H%M%S'*)

Average the list of timestamps.

Parameters

dates [list of strings] strings containing timestamps

infmt [strings] format of `dates`

outfmt [string, optional] format of the output time stamp

Returns

avg_timestamp [`datetime.datetime` instance] average time

string `avg_timestamp` formatted according to `outfmt`

Raises

:class:'~vdat.utilities.VDATDateError' if it fails to parse dates from the fits headers

`vdat.libvdat.symlink.`**`_find_nearest`**(*q*, *timestamp*, *n_nearest=1*, *nearest_then=None*)

Go through the list of query results, order them according to the absolute distance from `timestamp` and return the `n_nearest`.

Parameters

q [`peewee.SelectQuery`] query to use

timestamp [`datetime` instance] timestamp to use as reference

n_nearest [int, optional] maximum number of directories returned; set it to negative to return all

nearest_then [`timedelta` instance] if not None, don't consider any directory whose delta time is larger than `nearest_then`; applied after n_nearest

Returns

sorted_q [list of query results] ordered with respect to the timestamp

`vdat.libvdat.symlink.`**`db_create_references`**()

search reference zero and calibration directories and add them to the database

Logging

Deal with loggers

`vdat.libvdat.loggers.`**`setup_main_logger`**(*conf*, *name='logger'*)

Setup the main vdat logger.

Add to the logger called `name` a standard output and a file logger

Parameters

conf [`ConfigParser` instance] configuration options,

`vdat.libvdat.loggers.`**`make_logger`**(*logger*, *conf*, *section=None*)

Create a handler using the instructions in the configuration `section` and add it to the `logger`.

It looks for the following options:

- `remove_old`: if True remove all the previous handlers before adding the new one, default `False`
- `remove_std`: if True remove all the previous `StreamHandler`\`, default `False`
- the ones used by *`make_handler()`*

Parameters

logger [`Logger` instance] logger

conf [`ConfigParser` instance] configuration options

section [string, default] name of the section containing the information to build the logger; defaults to `logging.name`, where `name` is the logger name

`vdat.libvdat.loggers.`**`astropy_handlers`**(*conf*)

Tweak the astropy logger removing the existing (stream) handlers and setting a new one if required

Parameters

conf [`ConfigParser` instance] configuration options,

`vdat.libvdat.loggers.`**`ginga_handlers`**(*conf*)

Create/update the ginga logger removing existing handlers setting a new one if required

Parameters

conf [`ConfigParser` instance] configuration options,

`vdat.libvdat.loggers.`**`setup_command_loggers`**(*main_config*)

Set up the loggers for the commands connected or connectible to buttons

Parameters

main_config [`ConfigParser` instance] configuration options,

`vdat.libvdat.loggers.`**`add_parent`**(*conf, section, from_parent=['logdir', 'logger_level']*)

Extract the `section` and, if `from_parent` is not an empty list, extend the section with those entry from the parent, without overwriting existing entries in `section`

The parent section of e.g. `section=logging.main.stdout` is `logging`.

Parameters

conf [`ConfigParser` instance] configuration options,

section [string] name of the section in the configuration where to find all the info

Returns

:class:'~pyhetdex.tools.configuration.ConfigParser' section

`vdat.libvdat.loggers.`**`make_handler`**(*conf*)

Create a handler following the instructions contained in `section`

It looks for the following options:

- enabled: if False, doesn't create a handler and returns None; default False
- file_name: if not found, a `StreamHandler` is created, if exists, a `FileHandler` is created
- logdir: directory where the log file goes; default '.'
- format: formatting of the message, default %(levelname)s: %(message)s
- level: level of the handler, default INFO

Parameters

conf [`ConfigParser` section] configuration options for the handler; it can be created using *`add_parent()`*

Returns

None, if enabled is ''False''

:class:'~logging.StreamHandler': if not file name is given

:class:'~logging.FileHandler': otherwise

`vdat.libvdat.loggers.`**`_to_log_level`**(*level*, *default=20*)

Try to convert `level` to int or to extract the level `level` from the logging module. If both fail, return the default.

See here for more info about logging levels.

Parameters

level [string] numerical or literal value of the level

Returns

int handler level

2.2.3 `vdat.config` – The configuration functionality and files

`vdat.config.core` – The Core

Manage configuration

Mostly copied from vhc/libvhc/config.py

exception `vdat.config.core.`**`ConfigurationError`**

Generic error raised by the configuration sub-package

exception `vdat.config.core.`**`MissingConfigurationError`**

Raised when a configuration item does not exist

exception `vdat.config.core.`**`MissingConfigurationSectionError`**

Raised when a configuration section does not exist

`vdat.config.core.`**`load_all_configs`**(*args*)

Convenience function to load the vdat configuration files.

It loads:

- `"main"`: the standard configuration file `vdat_conf_fname`; given from the command line
- `"command"`: the YAML file(s) defining the commands
- `"tasks"`: the YAML file(s) that instruct the GUI how to display the files; the names are in the `tasks_config` of the `general` section of the "default" configuration

Parameters

args [Namespace] Command line arguments

`vdat.config.core.`**`load_std_conf`**(*name*, *args*, **fnames*)

Load standard configuration files

Parameters

name [string] name to associate to the configurations

args [Namespace] Command line arguments

fnames [list of strings] the filename(s) of the configuration files

Returns

flist [list] files successfully loaded

`vdat.config.core.`**`load_yaml`**(*name*, **fnames*)

Load YAML configuration files. Skip non existing files.

Parameters

name [string] name to associate to the configurations

fnames [list of strings] the filename(s) of the configuration files

Returns

flist [list] files successfully loaded

`vdat.config.core.`**`default_dict`**()

Default values of the configuration object

Returns

defaults [dict] options-values pairs

`vdat.config.core.`**`get_config`**(*name*, *section=None*)

Returns the configuration file with the specified name.

Parameters

name [string, optional] name associated with the configuration object; by default returns the main configuration

section [string, optional] optionally returns only one section

Returns

configuration object can be either a `ConfigParser` or a dictionary(-like) object

`vdat.config.entry_point` – The `vdat_config` executable

Implementation of the `vdat_config` entry point

`vdat.config.entry_point.`**`main`**(*argv=None*)

Main function of the implementation of the `vdat_config` executable

Parameters

argv [list] command line, if None taken from `sys.argv`

`vdat.config.entry_point.`**`parse`**(*argv=None*)

Create the parser and parse the command line arguments

Parameters

argv [list] command line, if None taken from `sys.argv`

Returns

Namespace parsed command line

`vdat.config.entry_point.`**`resource_replace`**(*file_content*, *filename*, *path*)

Replace in the file content the `<++>CONFIG_DIR<++>` and `<++>VERSION<++>` placeholders.

Parameters

file_content [string] content of resource

filename [string] name of the file from which the resource is coming

path [string] path where the file will be copied

Returns

resource [string] modified resource

class `vdat.config.entry_point.`**`VDATCopyResource`**(*name*, *backup=False*, *force=False*, *verbose=False*)

Overrides `manipulate_resource()` to replace placeholders

`manipulate_resource`(*file_content*)

call *`resource_replace()`*

`vdat.config.entry_point.`**`copy`**(*args*)

Copy the configuration files

Parameters

args [Namespace] arguments to make the copy run

`vdat.config.entry_point.`**`compare`**(*args*)

Compare the configuration files

Parameters

args [Namespace] arguments to make the copy run

`vdat.config.entry_point.`**`_existing_files`**(*args*)

Check which files exist and which not.

Parameters

args [Namespace] arguments to make the copy run

Returns

existing_files [list of strings] existing files

`vdat.config.entry_point.`**`_compare_versions`**(*fnames*, *args*)

Check the version of the files.

Parameters

fnames [list of strings] list of files for which to check the version

args [Namespace] arguments to make the copy run

Returns

remaining_files [list of strings] files for which no version is expected

`vdat.config.entry_point.`**`_file_diff`**(*fnames*, *args*)
Make a diff for all the files

Parameters

fnames [list of strings] list of files for which to check the version

args [Namespace] arguments to make the copy run

Returns

equal_files, diff_files [list of strings] files that are the same and that are different

`vdat.config.entry_point.`**`_file_load`**(*fnames*, *args*)
Try to load the files with the appropriate function to check if they are reasonable

Parameters

fnames [list of strings] list of files for which to check the version

args [Namespace] arguments to make the copy run

Returns

success_files, failed_files, skipped_files [list of strings] files that where successfully loaded, failed or where skipped

`vdat.config.versions` – Versioning the configuration files

Configuration and support files versioning and checking

`vdat.config.versions.`**`COPY_FILES = ['vdat_setting.cfg', 'vdat_commands.yml', 'tasks.yml', '`**
files to copy

`vdat.config.versions.`**`IMPORTANT_FILES = ['vdat_setting.cfg', 'vdat_commands.yml', 'tasks.yml`**
Files essential to run VDAT but that can have different names and be however passed to the code via configuration or command line options

`vdat.config.versions.`**`VERSION_VDAT_SETTING = '2.0.0'`**
version of the vdat_setting file

`vdat.config.versions.`**`VERSION_VDAT_COMMANDS = '1.3.0'`**
version of the vdat_commands file

`vdat.config.versions.`**`VERSION_TASKS = '1.3.0'`**
version of the tasks file

`vdat.config.versions.`**`version_name`**(*filename*)
Return the name of the variable storing the version of `filename`

Parameters

filename [string] name of the file for which to find the version

Returns

string name of the variable storing the configuration file

`vdat.config.versions.`**`version_of_file`**(*filename*)
If available, returns the version string for the `filename` as defined in *`vdat.config.versions`*

Parameters

filename [string] name of the file for which to find the version

Returns

version [string] version or None, if no version is available

`vdat.config.versions.`**`version_from_file`**(*path*, *filename*)

Try to extract the version number from `path + filename`

Parameters

path [string] path where the file resides

filename [string] name of the file

Returns

version [string] version as extracted from the file or None, if no version is found

`vdat.config.versions.`**`files_with_version`**()

Return the list of file names, that can be copied, that have an associated version.

Returns

list of strings file names

`exception` `vdat.config.versions.`**`LoaderError`**

Raised when a loader fails for any reason

`vdat.config.versions.`**`configparser_loader`**(*fname*)

Try to load the file using configparser

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

`vdat.config.versions.`**`yaml_loader`**(*fname*)

Try to load the file using yaml

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

`vdat.config.versions.`**`fplane_loader`**(*fname*)

Try to load the file as a fplane file.

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

`vdat.config.versions.`**`dist_loader`**(*fname*)

Try to load the file as a distortion file.

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

`vdat.config.versions.`**`ifucent_loader`**(*fname*)

Try to load the file as an ifu center file.

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

`vdat.config.versions.`**`lines_loader`**(*fname*)

Try to load the file as an line file.

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

`vdat.config.versions.`**`dither_position_loader`**(*fname*)

Try to load the file as dither position file.

Parameters

fname [string] file to load

Raises

LoaderError if it fails to load

2.2.4 `vdat.gui` – The Gui Code

This part of the documentation strives at describing the code itself in the best possible way and to give detailed information about custom signals and slots defined and connected in every class.

The Gui

`vdat.gui.central` – The widget at the center of VDAT

This describe the central VDAT widget

class `vdat.gui.central.`**`VDATCentral`**(*parent=None*)

Bases: `PyQt5.QtWidgets.QWidget`

This widget groups together the focal plane widget and the buttons into a Qsplitter, to allow resizing and decouple them from the rest of the gui.

Table 1: **Custom signals**

Name	Signature	Description
sig_ifuSelected	str, bool	emitted when a user selects/deselects an IFU; the parameters are the SLOTID of the IFU and whether the IFU has been selected
sig_selectAllIFUs		emitted when a user selects all IFUs
sig_deselectAllIFUs		emitted when a user deselects all IFUs

Table 2: **Custom slot**

Name	Signature	Description
change_target()	str, str	Change the view to the path passed as first argument; the second argument is the type of files associated to the path.
change_task()	str, dict	Switch the change the focal plane to the view for the `task` passed as first argument; the second argument contains the dictionary describing the task tabs and buttons.
ifuToggled()	str, bool	Mark `ifuslot`, given in the first argument as selected, if the second argument is True.
selectAllIFUs()		Select all the IFUs.
deselectAllIFUs()		Deselect all the IFUs.
runCommand()	str, list	When clicking a button create the commands and submit them to the queue

Table 3: **Connections between custom signals and/or slots**

Signal	Slot
vdat.gui.tasks.VDATTaskStack.sig_taskChanged	*change_task()*
vdat.gui.tasks.VDATTaskStack.sig_runCommand	*runCommand()*
sig_selectAllIFUs	*vdat.gui.fplane.FplaneWidget.selectAllIFUs()*
sig_deselectAllIFUs	*vdat.gui.fplane.FplaneWidget.deselectAllIFUs()*
sig_ifuSelected	*vdat.gui.fplane.FplaneWidget.ifuSelected()*
vdat.gui.fplane.FplaneWidget.sig_ifuToggled	*ifuToggled()*

Parameters

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

sig_ifuSelected

sig_selectAllIFUs

sig_deselectAllIFUs

change_target(*target*, *typ*)

Change the view to the path `target` of `typ`. If the selected type has not tasks associated, show the overlay.

This method is also a pyqt slot with signature `str, str`.

Parameters

target [string] path of the selected directory

typ [string] type of the `target`

change_task(*task*, *task_dict*)

Switch the change the focal plane to the view for the `task`

This method is also a pyqt slot with signature `str, dict`.

Parameters

task [string] name of the task to view

task_dict [dict] dictionary with the configuration for the tabs and buttons to display

ifuToggled(*ifuslot*, *val*)

Mark `ifuslot` as selected or not

This method is also a pyqt slot with signature `str, bool`.

Parameters

ifuslot [str] SLOTID of one IFU

val [bool] True to select the IFU, False otherwise

selectAllIFUs()

Select all the IFU

This method is also a pyqt slot.

deselectAllIFUs()

Deselect all the IFU

This method is also a pyqt slot.

runCommand(*s*, *l*)

When clicking a button create the commands and submit them to the queue

This method is also a pyqt slot with signature `str, list`

Parameters

s [string] name of the button clicked or cumulative name of the commands to execute

l [list] list of command to execute

_config_with_dirs(*command_name*)

Get the command configuration and add `target_dir`, `zero_dir` and `cal_dir` to the configuration.

Parameters

command_name [string] name of the command

Returns

command_conf [dict] configuration dictionary for the command at hand

_error_dialog(*error*, *command*)

Create the dialog to show the error

Parameters

error [`Exception` instance] error raised by the constructor

command [string] full command string

vdat.gui.fplane – The fplane viewer

Panel with the focal plane

The FplaneWidget

class vdat.gui.fplane.**FplaneWidget**(*parent=None*)

Bases: PyQt5.QtWidgets.QTabWidget

Widget containing the tabs showing the focal plane.

Table 4: **Custom signals**

Name	Signature	Description
sig_ifuToggled	str, bool	emitted when an IFU is selected in a widget contained in this one; the id of the IFU and whether is selected (True) or deselected (False) are passed as arguments.
sig_ifuSelected	str, bool	emitted when an IFU is selected in a widget that contains this one; the id of the IFU and whether is selected (True) or deselected (False) are passed as arguments.
sig_selectAllIFUs		emitted when all the IFUs are selected in a widget that contains this one
sig_deselectAllIFUs		emitted when all the IFUs are deselected in a widget that contains this one

Table 5: **Custom slot**

Name	Signature	Description
change_fplane()	str, dict	Upon selecting a new directory or task, update the tabs for the directory passed as first argument according to the instruction in the dictionary passed as second argument.
ifuToggled()	str, bool	Emit the *sig_ifuToggled* signal
ifuSelected()	str, bool	Emit the *sig_ifuSelected* signal
selectAllIFUs		Emit the *sig_selectAllIFUs* signal
deselectAllIFUs		Emit the *sig_deselectAllIFUs* signal

Table 6: **Connections between custom signals and/or slots**. If the signal or slot belongs to tab, it is connected in change_fplane() and disconnected in empty_fplane()

Signal	Slot
tab.sig_ifuToggled	*ifuToggled()*
sig_selectAllIFUs	tab.selectAllIFUs
sig_ifuSelected	tab.ifuSelected
sig_deselectAllIFUs	tab.deselectAllIFUs

Parameters

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

sig_ifuToggled

sig_ifuSelected

sig_selectAllIFUs

sig_deselectAllIFUs

empty_fplane()

Loop through the current tabs and for each one of them:

- remove from the *FplaneWidget*
- disconnect all the signals connected in *change_fplane()*
- if the `use_cache` property is `True`, save the widget in the cache
- if the `use_cache` property is `False`, mark the widget for deletion

change_fplane(*target*, *task_dict*)

Clear the old tabs and create a set of new ones.

It loops through the list in the `tabs` section of `task_dict`, load and call the plugins implementing each `tab_type`. If the plugin exists and the execution succeed, for each widget returned:

- connect the *sig_selectAllIFUs*, *sig_deselectAllIFUs* and *sig_ifuSelected* signals and the *ifuToggled()* slot
- add it as a tab

All errors when loading and calling the plugins are collected and shown before return the method. If no tab is found or no tab could be plugged in because of errors, show an overlay to notify the user.

Parameters

target [string] path of the selected directory

task_dict [dict] dictionary with the configuration for the tabs and buttons to display

show_tab_errors(*errors*)

Shows the errors into a QErrorMessage window.

Parameters

errors [list of strings] errors to report

ifuToggled(*id_*, *val*)

Emit the *sig_ifuToggled* signal. The method is also a PyQt slot.

Parameters

id_ [string] SLOTID of the ifu that is toggled

val [bool] `True` if selected, `False` if deselected

ifuSelected(*id_*, *val*)

Emit the *sig_ifuSelected* signal. The method is also a PyQt slot.

Parameters

id_ [string] SLOTID of the ifu that is toggled

val [bool] `True` if selected, `False` if deselected

selectAllIFUs()

Emit the *sig_selectAllIFUs* signal. The method is also a PyQt slot.

deselectAllIFUs()
Emit the *sig_selectAllIFUs* signal. The method is also a PyQt slot.

The cache

class vdat.gui.fplane.**FplaneCache**
Cache of fplane objects.

This object allows to store and retrieve objects in dictionary of lists. Each entry has the type of the object (as provided by the user) as key and a list of object as value.

from_cache(*fp_type*)
Returns one object stored under the name `fp_type`.

Parameters

fp_type [string] name of the type to store

Returns

An object or ‘‘None‘‘ if the cache is empty.

into_cache(*fp_type*, *fp*)
Store the object (a widget) into the cache.

Parameters

fp_type [string] name of the type to store

fp [object] object to be stored.

vdat.gui.help_window – The offline help window

class vdat.gui.help_window.**HelpBrowser**(*help_engine*, *parent=None*)
Bases: `PyQt5.QtWidgets.QTextBrowser`

Custom `QTextBrowser` that can deal with documentation and external links

Table 7: **Custom slot**

Name	Signature	Description
setSource()	`PyQt5.QtCore.QUrl`	if the input url is http/https opens in the default internet browser, other wise pass it further

Parameters

help_engine [`PyQt5.QtHelp.QHelpEngine`] help engine containing the documentation

setSource(*url*)
Reimplement slot to properly handle http/https links via `PyQt5.QtGui.QDesktopServices.openUrl()`. All the other types are passed to the original implementation.

Parameters

url [`PyQt5.QtCore.QUrl`] url that has being clicked

Returns

resource to show

loadResource(*type_*, *url*)

Reimplement the method to get `qthelp` urls from the help engine. All the other types are passed unchanged to the original implementation.

Parameters

type_ [int] ignored

url [`PyQt5.QtCore.QUrl`] url that has being clicked

Returns

resources to show

class vdat.gui.help_window.**HelpWidget**(*parent=None*)

Bases: `PyQt5.QtWidgets.QWidget`

Create a widget containing the offline help.

Table 8: **Custom slot**

Name	Signature	Description
expand_and_set()		show the first page of the documentation and expand the first level of the content
on_click_content	`PyQt5.QtCore.QModelIndex`	Extract the url of the selected item and send it to *HelpBrowser* for displaying

Table 9: **Connections between custom signals and/or slots**

Signal	Slot
`PyQt5.QtHelp.QHelpContentModel.contentsCreated`	*expand_and_set()*
`PyQt5.QtHelp.QHelpContentWidget.clicked`	*on_click_content()*
`PyQt5.QtHelp.QHelpIndexWidget.linkActivated`	*HelpBrowser.setSource()*

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] parent of the menu' bar

setup()

Setup the gui elements into a resizable splitter and connect all the signals

expand_and_set()

Custom slot: it gets the index of the first element, set it as first page in the help browser and expand the sections below it.

on_click_content(*index*)

Custom slot: receive the clicked index in the content tree, get the underlying url and send it to the *HelpBrowser*

Parameters

index [`PyQt5.QtCore.QModelIndex`] index of the selected item

class vdat.gui.help_window.**HelpWindow**(*parent=None*)

Bases: `PyQt5.QtWidgets.QMainWindow`

Window wrapping the HelpWidget

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] parent of the menu' bar

vdat.gui.logger_widget – Logging widget

Logging handlers

class vdat.gui.logger_widget.**TextWindowHandler**(*browser*, *parent=None*)

Bases: PyQt5.QtCore.QObject, logging.Handler

This is an implementation of a logging Handler that prints log messages to a QTextEdit widget

Table 10: **Custom signals**

Name	Signature	Description
postText	string	emit the signal with the text to submit

Table 11: **Custom slot**

Name	Signature	Description
post_to_panel()	string	append the incoming message to textBrowser

Table 12: **Connections between custom signals and/or slots**.

Signal	Slot
postText	*post_to_panel()*

Parameters

browser [PyQt5.QtWidgets.QTextEdit] A text edit widget to write the information to

parent [qobject, optional] parent of the QObject used to create the logger

Attributes

textBrowser [PyQt5.QtWidgets.QTextEdit] text widget where the log message is posted

postText

emit(*record*)

Construct a string in HTML out of the record, and emit a signal that tells another function to update the text browser

Parameters

record [logging.LogRecord] a record from a logger

post_to_panel(*msg*)

Update the text window in the GUI by appending msg.

This method is also a PyQt slot.

Parameters

msg [String] A message to post to the test browser window

class vdat.gui.logger_widget.**LoggerWidget**(*parent=None*)

Bases: PyQt5.QtWidgets.QTextEdit

Widget where the logging messages are shown

vdat.gui.mainwidget – The biggest widget

Create the main central widget of the VDAT GUI

class vdat.gui.mainwidget.**VDATMainWidget**(*parent=None*)

Bases: PyQt5.QtWidgets.QWidget

Central widget containing the focal plane, splitter etc.

sig_selectAllIFUs

sig_deselectAllIFUs

setup()

selectAllIFUs()

deselectAllIFUs()

redoSymlink()

Redo the symlinking, create a new model, plug into the tree view and trigger the redraw

_redo_symlink()

Function to execute in a thread when *redoSymlink()* is triggered

vdat.gui.mainwindow – The main VDAT window

Create the main window of the VDAT GUI

class vdat.gui.mainwindow.**VDATMainWindow**

Bases: PyQt5.QtWidgets.QMainWindow

VDAT customisation of the main window.

Table 13: **Custom slot**

Name	Signature	Description
remove_files()	list, list	run the removal of files in a thread and change the cursor to the wait cursor

Table 14: **Connections between custom signals and/or slots**

Signal	Slot
vdat.gui.queue.Queue.global_logger	logging.Logger.log()
vdat.gui.queue.Queue.progress	*vdat.gui.progress.VDATProgressBar.update_bar()*
vdat.gui.queue.Queue.n_primaries	*vdat.gui.progress.VDATProgressBar.setup_bar()*
vdat.gui.queue.Queue.command_done	*vdat.gui.progress.VDATProgressBar.reset_bar()*
vdat.gui.queue.Queue.command_string	*vdat.gui.progress.VDATStatusBar.clear_message()*
vdat.gui.queue.Queue.command_done	*vdat.gui.progress.VDATStatusBar.running_command()*
vdat.gui.menubar.VDATMenuBar.sig_close	vdat.gui.mainwidget.VDATMainWidget.close()
vdat.gui.menubar.VDATMenuBar.sig_symlink	*vdat.gui.mainwidget.VDATMainWidget.redoSymlink()*
vdat.gui.menubar.VDATMenuBar.sig_selectAll	*vdat.gui.mainwidget.VDATMainWidget.selectAllIFUs()*
vdat.gui.menubar.VDATMenuBar.sig_selectNone	*vdat.gui.mainwidget.VDATMainWidget.deselectAllIFUs()*
vdat.gui.menubar.VDATMenuBar.sig_remove_files	*remove_files()*
vdat.gui.treeview_model.ReductionQTreeView.sig_selectionChanged	*vdat.gui.menubar.VDATMenuBar.enable_on_selection()*
vdat.gui.menubar.TreeViewMenu.sig_collapse	vdat.gui.treeview_model.ReductionQTreeView.collapseAll()
vdat.gui.menubar.TreeViewMenu.sig_expand	vdat.gui.treeview_model.ReductionQTreeView.expandAll()

connect_queue()

connect signals from the queue

connect_menubar()

Connect the signals from the menu bar

setup()

Set-up the user interface for VDAT

Parameters

queue [*vdat.gui.queue.Queue* instance] queue where to push the command. It must have a add_command method

closeEvent(*event*)

Override the user closing the window. Instead wait for anything on the immediate queue to finish running and then quit.

remove_files(*paths*, *matches*)

Slot to connect the *vdat.gui.menubar.VDATMenuBar.sig_remove_files* signal that removes the files in a thread. The mouse cursor is temporarily set to wait.

Parameters

paths [list of strings] paths containing the files to remove

matches [list of string] file names or wildcards pattern for removal

vdat.gui.menubar – The menu bar

class vdat.gui.menubar.**VDATMenuBar**(*parent=None*)

Bases: PyQt5.QtWidgets.QMenuBar

Menu bar of the main VDAT window

Table 15: **Custom signals**

Name	Signature	Description
sig_close		Emitted when the user click on the "Quit" action
sig_symlink		Emitted when the user click on the "symlink" action
sig_selectAll, *sig_selectNone*		Emitted when the actions to select or deselect all the IFUs are clicked
sig_remove_files	list, list	Emitted when one of the remove file options are clicked. The arguments are a list of directories containing the files to remove and a list of file names/wildcards to remove
sig_clear_log		Emitted when the user click 'Clear the log window' button
sig_collapse		emitted when the "Reduction Browser"'s "Collapse" action is triggered
sig_expand		emitted when the "Reduction Browser"'s "Expand" action is triggered

Table 16: **Custom slot**

Name	Signature	Description
enableSymlink()	bool	whether to enable or not the symlink action
enable_on_selection()	str, str	save the first string into a path attribute and enabled all the actions disabled at startup
clear_files_slot()		If a directory is selected, get the current selected directory and the wildcards/filenames and emit the *sig_remove_files* signal
clear_all_files_slot()		Get all the directories from the database and the wildcards/filenames and emit the *sig_remove_files* signal
clear_thumb_slot()		If a directory is selected, get the thumbnail directory under it and emit the *sig_remove_files* signal with '*' as wildcard
clear_all_thumb_slot()		Get all the thumbnail directories and emit the *sig_remove_files* signal with '*' as wildcard

Table 17: **Connections between custom signals and/or slots**

Signal	Slot
`triggered` signal of the `self.clear_files` action	*clear_files_slot()*
`triggered` signal of the `self.clear_all_files` action	*clear_all_files_slot()*
`triggered` signal of the `self.clear_thumb` action	*clear_thumb_slot()*
`triggered` signal of the `self.clear_all_thumb` action	*clear_all_thumb_slot()*
sig_clear_log	`menus_actions.LogMenu.clear_log.triggered`
sig_collapse	*menus_actions.TreeViewMenu.sig_collapse*
sig_expand	*menus_actions.TreeViewMenu.sig_expand*

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] parent of the menu' bar

sig_close

sig_symlink

sig_selectAll

sig_selectNone

sig_remove_files

sig_clear_log

sig_collapse

sig_expand

enable_on_selection(*path*, *typ*)

Save the path and the type and enable disabled actions when the path is non-null.

Parameters

path [string] a path, usually the directory selected in three view

typ [string] type of the path

create_file()

Create and fill the "File" menu

Returns

view_menu [`PyQt5.QtWidgets.QMenu`] menu for the "View" entry

enableSymlink(*b*)

deletions(*menu*)

Create deletion actions and add them to the menu

Parameters

menu [`PyQt5.QtWidgets.QMenu`] menu to which the action must be added

Returns

menu input menu

clear_files_slot()
Get the current directory and the wildcards/filenames and emit the *sig_remove_files* signal

clear_all_files_slot()
Get all the directories and the wildcards/filenames and emit the *sig_remove_files* signal

clear_thumb_slot()
Get the thumbnail dir under current directory and emit a *sig_remove_files* signal matching every file

clear_all_thumb_slot()
Get all the thumbnail directories and emit the *sig_remove_files* signal matching every file

create_view()
Create and fill the "View" menu

Returns

view_menu [`PyQt5.QtWidgets.QMenu`] menu for the "View" entry

create_select()
Create and fill the "Select" menu

Parameters

fplane [`vdat.gui.FplaneWidget`] instance of fplate to connect to the buttons for selection

Returns

select_menu [`PyQt5.QtWidgets.QMenu`] menu for the "Select" entry

`vdat.gui.menus_actions` – Custom menus and actions

Module that implements custom menus and action

class `vdat.gui.menus_actions.`**RemoveExposuresMenu**(*parent=None*)
Bases: `PyQt5.QtWidgets.QMenu`

A menu that removes the exposures matching the `path` and create a list of actions

Table 18: **Custom signals**

Name	Signature	Description
sig_exposure_removed	str	emitted when an exposure is removed; the value is the `expname` from the database

Table 19: **Custom slot**

Name	Signature	Description
enable_on_selection()	str, str	save the first string into a `path` attribute
about_to_show_slot()		dynamically adds actions to the menu just before showing it
remove_exposure()	`PyQt5.QtWidgets.QAction`	receive the triggered action, using its object name find the basename, remove files with that basename and the corresponding entry in the database and in the exposure file

Table 20: **Connections between custom signals and/or slots**

Signal	Slot
`RemoveExposuresMenu.aboutToShow`	*`about_to_show_slot()`*
`RemoveExposuresMenu.triggered`	*`remove_exposure()`*

`sig_exposure_removed`

`path`

Path where the exposures to remove are located. When setting a non `None` path, the menu is activated; when setting a `None` path or the path gets deleted, the menu is deactivated

`enable_on_selection`(*path*, *typ*)

Save the path and the type and enable disabled actions when the path is non-null.

Parameters

path [string] a path, usually the directory selected in three view

typ [string] type of the path, ignored

`about_to_show_slot`()

Slot to connect with the `aboutToShow` signal. If no path has been selected, nothing happens, otherwise dynamically create actions in the menu

`remove_exposure`(*action*)

Get the triggered action, remove the files for the given exposure. Then remove the exposure from the VDATExposures database and the metadata file in the directory.

Parameters

action [`PyQt5.QtWidgets.QAction`] action triggered

class `vdat.gui.menus_actions.`**`QuitAction`**(*connect_to=None*, *parent=None*)

Bases: `PyQt5.QtWidgets.QAction`

Quit action to plug in menus or toolbars.

Set the icon to `window-close`, the text to `Quit` and the shortcut to `Ctrl+Q`.

Table 21: **Connections between custom signals and/or slots**

Signal	Slot
`triggered`	`connect_to`, if present

Parameters

connect_to [signal or slot, optional] if present connect the `triggered`

parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the menu' bar

class `vdat.gui.menus_actions.`**`HelpMenu`**(*parent=None*, *windows_parent=None*)

Bases: `PyQt5.QtWidgets.QMenu`

Create a menu with the various help actions.

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the menu

help_parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the windows opened by the actions in this menu

class vdat.gui.menus_actions.**HandbookAction**(*parent=None*, *window_parent=None*)

Bases: PyQt5.QtWidgets.QAction

"Show Handbook" action. It Shows *vdat.gui.help_window.HelpWindow*

Table 22: **Custom slot**

Name	Signature	Description
show_handbook()		Create a *vdat.gui.help_window.HelpWindow* and show it

Table 23: **Connections between custom signals and/or slots**

Signal	Slot
triggered	*show_handbook()*

Parameters

parent [PyQt5.QtWidgets.QWidget instance, optional] parent of the action

window_parent [PyQt5.QtWidgets.QWidget instance, optional] parent of the help window

show_handbook()

open the VDAT handbook in a new window

This method is also a PyQt slot

class vdat.gui.menus_actions.**VDATLinksAction**(*parent=None*, *window_parent=None*)

Bases: PyQt5.QtWidgets.QAction

Show links with the online VDAT documentation.

Table 24: **Custom slot**

Name	Signature	Description
show_links()		Create a QMessageBox showing the links

Table 25: **Connections between custom signals and/or slots**

Signal	Slot
triggered	*show_links()*

Parameters

parent [PyQt5.QtWidgets.QWidget instance, optional] parent of the action

window_parent [PyQt5.QtWidgets.QWidget instance, optional] parent of the help window

show_links()

Display the links to the VDAT documentation and the coverage reports, both for the latest and the development version.

This method is also a PyQt slot

class vdat.gui.menus_actions.**VDATAboutAction**(*parent=None*, *window_parent=None*)

Bases: PyQt5.QtWidgets.QAction

Show a short "about" VDAT. It Shows *vdat.gui.help_window.HelpWindow*

Table 26: **Custom slot**

Name	Signature	Description
show_about()		Create a `QMessageBox` showing some information about authors and such

Table 27: **Connections between custom signals and/or slots**

Signal	Slot
`triggered`	*show_about()*

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the action

window_parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the help window

show_about()

Open the VDAT about in a message box.

This method is also a PyQt slot

class vdat.gui.menus_actions.**LogAction**(*fname*, *parent=None*)

Bases: `PyQt5.QtWidgets.QAction`

Action about the log files

Parameters

fname [string] name of the file to open

parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the menu

Attributes

fname [string] name of the file to open

class vdat.gui.menus_actions.**LogViewerWindow**(*file_name*, *parent=None*)

Bases: `PyQt5.QtWidgets.QMainWindow`

Editor window used to display log files.

Table 28: **Custom slot**

Name	Signature	Description
load_file()		Load the file and disable the refresh action
enable_refresh()	string	Enable the refresh button

Table 29: **Connections between custom signals and/or slots**.

Signal	Slot
`refresh.triggered`	*load_file()*
`watcher.fileChanged`	*enable_refresh()*

Parameters

file_names [string] file to display

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

Attributes

file_name [string] name of the file to display

text_edit [`PyQt5.QtWidgets.QTextEdit`] display the text

refresh [`PyQt5.QtWidgets.QAction`] action to refresh the content

watcher [`PyQt5.QtCore.QFileSystemWatcher`] file watcher

make_text_edit()
Create a QTextEdit edit and returns it

load_file()
Load the file in the `text_edit` widget and disable the refresh action

This method is also a PyQt slot

enable_refresh(*fname*)
Enable the refresh button

make_common_actions()
Create actions.

make_menubar()
Create and return the menu

make_toolbar()
Create and return the toolbar

class vdat.gui.menus_actions.**LogMenu**(*parent=None*)
Bases: `PyQt5.QtWidgets.QMenu`

Create a menu with actions related with log files.

Table 30: **Custom slot**

Name	Signature	Description
open_log_file()	*LogAction*	open the selected log file in a read-only window

Table 31: **Connections between custom signals and/or slots**.

Signal	Slot
`triggered` of the log files menu	*open_log_file()*

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the menu

open_log_file(*action*)
Open the given log files

class vdat.gui.menus_actions.**TreeViewMenu**(*parent=None*)
Bases: `PyQt5.QtWidgets.QMenu`

Menu to collapse/expand the tree view

Create the menu with the entries to expand/collapse the menu

Table 32: **Custom signals**

Name	Signature	Description
sig_collapse		emitted when the "Collapse" action is triggered
sig_expand		emitted when the "Expand" action is triggered

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance, optional] parent of the menu

sig_collapse

sig_expand

`vdat.gui.progress` – Monitor the execution progress

Custom progress and status bar objects

class `vdat.gui.progress.`**VDATProgressBar**(*parent=None*)

Bases: `PyQt5.QtWidgets.QProgressBar`

VDAT progress bar.

Table 33: **Custom slot**

Name	Signature	Description
setup_bar()	int	setup the format, the maximum to the input value and the value to zero
update_bar()	int, int, int, int	update the value of the progress bar; only the second value used
reset_bar()	bool	if the input is `True` reset the progress bar

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] the parent widget

setup_bar(*max_*)

Setup the format, the maximum to the input value and the value to zero. Designed to be connected with the `n_primaries` signal from *`vdat.command_interpreter.signals`*

Parameters

max_ [int] maximum value for the progress bar

update_bar(*int_tot*, *int_done*, *int_skipped*, *int_fail*)

Set the value of the progress bar to `int_done`. Designed to be connected with the `progress` signal from *`vdat.command_interpreter.signals`*.

reset_bar(*bool_global*)

If `bool_global == True`, reset the progress bar. Designed to be connected with the `command_done` signal from *`vdat.command_interpreter.signals`*.

class `vdat.gui.progress.`**VDATStatusBar**(*parent=None*)

Bases: `PyQt5.QtWidgets.QStatusBar`

Customized status bar

Table 34: **Custom slot**

Name	Signature	Description
running_command()	int, str	show a message with the input string
clear_message()	bool	if the input is `True` show the message "Done" for 3 seconds

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] the parent widget

running_command(*int_*, *str_*)

Show the input string. Designed to be connected with the `command_string` signal from *vdat.command_interpreter.signals*.

clear_message(*global_*)

If `bool_global == True`, show "Done" for 3 seconds. Designed to be connected with the `command_done` signal from *vdat.command_interpreter.signals*.

vdat.gui.queue – The Queue window

Create a window representing a queue via a list of items.

The original implementation has been generated from reading ui file 'listWindow.ui'

Created: Mon Jun 15 16:25:52 2015 by: PyQt4 UI code generator 4.10.4

class vdat.gui.queue.**QueuedCommand**(*command*, *label*, *tool_tip=None*, *parent=None*)

Bases: `PyQt5.QtWidgets.QListWidgetItem`

Class describing the objects stored in the *ModifyableListWidget*.

Each object represent a command

Parameters

command [*QCommandInterpreter*] instance of the command interpreter to add to the queue

label [string] a label to appear for this command on the queue

tool_tip [string, optional] tool tip to show

parent [`PyQt5.QtWidgets.QWidget` instance] the parent widget

class vdat.gui.queue.**ModifyableListWidget**

Bases: `PyQt5.QtWidgets.QListWidget`

List widget with the possibility to remove items

keyPressEvent(*event*)

Override the default method, removing the selected entry

Parameters

event [`PyQt5.QtGui.QKeyEvent`] object describing a key being pressed or released

class vdat.gui.queue.**Queue**(*parent=None*)

Bases: `PyQt5.QtWidgets.QMainWindow`

A queue that stores user commands and displays them in a GUI window.

Table 35: **Custom signals**

Name	Signature	Description
closeSignal		emitted when the queue window is closed
job_added		emitted when a job is added to the queue
run_signal		emitted when a new command can be run
queue_empty_signal		emitted when the queue is empty
command_done	bool	these five signals are a Qt re-implementation of the signals described in the *signals*
command_string	int, str	
global_logger	int, str	
n_primaries	int	
progress	int, int, int, int	

Table 36: **Custom slot**

Name	Signature	Description
toggle()	bool	hide (False) or show (True) the panel
run()		grab a command from the queue and run it
toggle_and_rerun()		prepare to run a new command

Table 37: **Connections between custom signals and/or slots**

Signal	Slot/Signal
job_added	*run()*
run_signal	*run()*
QCommandInterpreter.command_done	*command_done*
QCommandInterpreter.command_string	*command_string*
QCommandInterpreter.global_logger	*global_logger*
QCommandInterpreter.n_primaries	*n_primaries*
QCommandInterpreter.progress	*progress*
`PyQt5.QtCore.QThread.started`	*QCommandInterpreter.run()*
QCommandInterpreter.finished	*toggle_and_rerun()*

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] the parent widget

Attributes

is_command_running [bool] mark whether a command is running or not in a thread

closeSignal

job_added

run_signal

queue_empty_signal

command_done

command_string

global_logger

n_primaries

progress

_reconnect_names = ['command_done', 'command_string', 'global_logger', 'n_primaries',

setupUi()

Setup the queue window

closeEvent(*event*)

When the user closes the window, ignore the request, hide the window and emit the `closeSignal` signal.

Parameters

event [`PyQt5.QtGui.QKeyEvent`] object describing a key being pressed or released

toggle(*tggl*)

Hide or show the panel. Alias of `setVisible()`.

Parameters

toggle [bool] whether the window is visible or not

add_command(*command*, *label*, *tool_tip=None*)

Add a command to the queue. Emit the `job_added` signal.

Parameters

command [`QCommandInterpreter`] instance of the command interpreter to add to the queue

label [string] A label to appear for this command on the queue

tool_tip [string, optional] tool tip to show

get_command()

Get the top item from the queue.

Returns

:class:'QCommandInterpreter' instance command to run, or None if the list is empty

connect_worker_signals(*worker*)

Connect the `worker` signals to the corresponding ones in this class

Parameters

worker [`QCommandInterpreter`] worker instance with the signals to connect with this object ones

remove_old_thread_worker()

Quit the used thread and mark the it and worker for deletion

run()

Grab a job from the queue and run it on a newly created `QThread`.

If a command is already running, and return. If the queue is empty notify the user and return.

The `QThread` and `QCommandInterpreter` instances are saved in a local cache and removed the next time `run()` is called.

toggle_and_rerun()

Remove the old thread and command, mark that the command is not running and emit the `run_signal` signal

class vdat.gui.queue.**QCommandInterpreter**(**args*, ***kwargs*)

Bases: `vdat.command_interpreter.core.CommandInterpreter`, `PyQt5.QtCore.QObject`

Create a QObject from the `CommandInterpreter`.

Reimplement the *make_signals()* to use only the `command_logger` signals from the command_interpreter and use `Signal` for the other signals

Table 38: **Custom signals**

Name	Signature	Description
command_done	bool	these five signals are a Qt re-implementation of the signals described in the *signals*
command_string	int, str	
global_logger	int, str	
n_primaries	int	
progress	int, int, int, int	
finished		emitted when the run is finished

Table 39: **Custom slot**

Name	Signature	Description
run()		Run the command

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] the parent widget

args, kwargs: passed to the *CommandInterpreter*

command_done

command_string

global_logger

n_primaries

progress

finished

make_signals()

Use Signals instead of *vdat.command_interpreter.signals*, except for the command_logger one

connect_signals()

disconnect_signals()

run()

Transform the method to a Slot

class vdat.gui.queue.**QueuAction**(**args*, ***kwargs*)

Bases: `PyQt5.QtWidgets.QAction`

Action for the queue window.

Create the menu entry and binds signals known by the queue window to show/hide it

Table 40: **Custom slot**

Name	Signature	Description
update_text()	bool	change the text to show in the action item

Table 41: **Connections between custom signals and/or slots**

Signal	Slot/Signal
`toggled`	*`update_text()`*
`Queue.closeSignal`	`toggle()`
`toggled`	*`Queue.toggle()`*

Parameters

***args, **kargs:** arguments passed to the parent class

connect_with_queue()
Connect with signals and slots in *`Queue`*

update_text(*toggled*)
Update the text in the action when the queue window is opened or closed

Parameters

toggled [bool] whether is checked or not

vdat.gui.queue.**set_queue**(*parent=None*)
Create a *`Queue`* instance and save it. You can access it with *`get_queue()`*

Parameters

parent [`PyQt5.QtWidgets.QWidget` instance] the parent widget

vdat.gui.queue.**get_queue**()
Get the locally stored *`Queue`* instance

`vdat.gui.tasks` – Reduction steps and buttons

exception vdat.gui.tasks.**TaskError**
Bases: `Exception`

class vdat.gui.tasks.**VDATStack**(*cls*, *parent=None*, *name='tasks_stack'*)
Bases: `PyQt5.QtWidgets.QStackedWidget`

Parent widget containing the buttons to run the reduction steps

Parameters

parent [`QtWidget` instance] The QtWidget that the menu is attached to

name [string] name of the widget

get_subwidget(*name*)
Returns a sub-widget called `name`. Create it if needed

setCurrentWidgetByName(*name*)
Set the current widget by name

Parameters

name [string] name associated with the widget

class vdat.gui.tasks.**VDATTaskStack**(*parent=None*, *name='tasks_stack'*)
Bases: *`vdat.gui.tasks.VDATStack`*

Parent widget containing the buttons to run the reduction steps

Parameters

parent [`QtWidget` instance] The QtWidget that the menu is attached to

name [string] name of the widget

sig_taskChanged

sig_runCommand

get_subwidget(*name*)
Returns a sub-widget called `name`. Create it if needed

setCurrentWidgetByName(*name*)
Set the current widget by name

Parameters

name [string] name associated with the widget

taskSelected(*s*, *d*)

commandTriggered(*s*, *l*)

class vdat.gui.tasks.**VDATButtonStack**(*parent=None*, *name='tasks_stack'*)
Bases: *vdat.gui.tasks.VDATStack*

Parent widget containing the buttons to run the reduction steps

Parameters

parent [`QtWidget` instance] The QtWidget that the menu is attached to

name [string] name of the widget

sig_runCommand

get_subwidget(*name*)
Returns a sub-widget called `name`. Create it if needed

commandTriggered(*s*, *l*)

class vdat.gui.tasks.**VDATTaskWidget**(*parent: QWidget = None*, *flags: Union[Qt.WindowFlags, Qt.WindowType] = Qt.WindowFlags()*)
Bases: `PyQt5.QtWidgets.QWidget`

sig_selected

sig_command

selectFirst()

add_task(*task_dict*, *tool_tip=None*)
Adds a button to the widget.

Parameters

name [string] name of the button

commands [list of strings, optional] strings defining the commands

tool_tip [string, optional] text that appears as a tool tip when the user hovers their mouse over the button

add_stretch(*stretch=1*)
Add a stretch to the layout containing the buttons

Parameters

stretch [int, optional] stretch factor

buttonSelected(*i*)
Slot to catch the the button pressed from the button group, and re-emit with updated information

commandTriggered(*s*, *l*)
Slot to catch the button pressed from the button group, and re-emit with updated information

class vdat.gui.tasks.**VDATButtonWidget**(*parent: QWidget = None, flags: Union[Qt.WindowFlags, Qt.WindowType] = Qt.WindowFlags()*)

Bases: PyQt5.QtWidgets.QWidget

sig_runCommand

add_button(*name*, *command*, *tool_tip=None*)
Adds a button to the widget.

Parameters

name [string] name of the button

commands [list of strings, optional] strings defining the commands

tool_tip [string, optional] text that appears as a tool tip when the user hovers their mouse over the button

clicked(*i*)

vdat.gui.tasks.**setup_tasks**(*parent=None*)
Set up and return the buttons

Parameters

parent [PyQt5.QtWidgets.QWidget] parent of the button widget

Returns

:class:'ButtonsMenu' button widgets container

vdat.gui.treeview_model – A Custom Model for the Treeview Widget

Model and Tree view implementations to show and navigate the directory tree used by VDAT. The code has being written following this tutorial.

To represent the following structure

```
+- 20150622
    +-- cal
        +-- time_stamp_1
        +-- time_stamp_2
    +-- sci
        +-- object_1
        +-- object_2
    +-- zro
        +-- time_stamp_3
        +-- time_stamp_4
```

we first create a *ReductionTreeviewModel* instance and create a root node as a *ReductionNode*:

```
>>> model = ReductionTreeviewModel()
>>> nights = ReductionNode("Nights", '/path/to/redux', model)
>>> model.rootnode = nights
```

Then we need to create the node for the night:

```
>>> night1 = ReductionNode("20150622", '/path/to/redux/20150622',
...                         model, parent=nights)
>>> nights.add_subnode(night1)
```

and below it the nodes for the types. E.g. the calibration node would be:

```
>>> cal = ReductionNode("cal", '/path/to/redux/20150622/cal',
...                     model, parent=night1)
>>> night1.add_subnode(cal)
```

Finally we can add the final directories with, e.g.:

```
>>> cal_1 = ReductionNode('time_stamp_1',  '/path/to/redux/20150622/cal',
...                       model, parent=cal, selectable=True, checkable=True,
...                       tooltip='all relevant info go here')
>>> cal.add_subnode(cal_1)
```

The model must then been added to the *ReductionQTreeView* using the `ReductionQTreeView.setModel()` method as done in *ReductionQTreeView.set_model()*

class vdat.gui.treeview_model.**ReductionNode**(*name*, *path*, *model*, *type_=''*, *selectable=False*, *checkable=False*, *parent=None*, *tooltip=None*)

Bases: `object`

A class to store nodes in the custom treeview model.

Parameters

name [String] a label for the node to appear in the treeview

path [string] path associated with the name

model [*ReductionTreeviewModel* instance] the model to attach the node to

type_ [string] type of the shot in the node

selectable [bool, optional] where the current node is selectable or not

checkable [bool, optional] where the current node can have a check box associated

parent [*ReductionNode* instance, optional] parent of the current node

tooltip : if not `None` converted to a string with `pprint.pformat()`

Attributes

name, model, parent [as in the parameters]

column [int] column index

subnodes [list] sub-nodes of the current node

_stringify(*value*)

Unless `None` or a string, convert `value` to a string using pprint.

Parameters

value : value to stringify

Returns

string or None `None` if `value == None`, string otherwise

index()
Return the index of the node

Returns

index [PyQt5.QtCore.QModelIndex instance] the index to this node

add_subnode(*node*)
Add a subnode to the current node

Parameters

node [*ReductionNode* instance] the subnode to add

subnode(*row*)
Get the child at row

Parameters

row [int] index of the child

Returns

:class:'ReductionNode' instance required child

class vdat.gui.treeview_model.**ReductionTreeviewModel**(*parent=None, column_title='Reduction Browser'*)

Bases: PyQt5.QtCore.QAbstractItemModel

A model that stores the tree structure for the treeview widget

Parameters

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

column_title [string, optional] title of the column

rootnode
Get the rootnode of this tree

Returns

node [*ReductionNode* instance] the index of the node you want to be root node

checked_nodes
Return the checked nodes.

setData() makes sure that at most one node per type is selected

Returns

dictionary key: type (str) of the node ('sci', 'cal') value: corresponding node instance

columnCount(*parentIndex*)
Return number of columns, for us this is always 1

Parameters

parentIndex [PyQt5.QtCore.QModelIndex] index to the parent node

Returns

int the number of columns under parent: always 1

rowCount(*parentIndex*)
Return the number of subnodes under a parent node

Parameters

parentIndex [`PyQt5.QtCore.QModelIndex`] index to the parent node

Returns

nrows [int] the number of rows under parent

headerData(*section*, *orientation*, *role*)

Return information about the header items

Parameters

section [int] the section for which this is the header

orientation [int] flag indicating whether the header is horizontal or vertical

role [int] flag specifying the type of info requested (i.e. a title for the header, or an icon etc.)

Returns

string

data(*index*, *role*)

Return information about the specified node.

Parameters

index [`PyQt5.QtCore.QModelIndex` instance] the index of the node you want data for

role [int] flag specifying the type of info requested (i.e. a title or an icon etc.)

Returns

string or int depending of the input role:

- `PyQt5.QtCore.Qt.DisplayRole`: return the name of the node associated with the index
- `PyQt5.QtCore.Qt.CheckStateRole`: returns `PyQt5.QtCore.Qt.Checked`: `PyQt5.QtCore.Qt.Unchecked` whether the check-box is checked or not
- `PyQt5.QtCore.Qt.ToolTipRole`: if available, returns the string to put in the tooltip

setData(*index*, *value*, *role*)

If the role is `PyQt5.QtCore.Qt.CheckStateRole`, change the check status of the `index`, making sure that at most one element per checkable node type is selected.

Parameters

index [`QModelIndex` instance] the index of the node you want data for

value : value to set (ignored)

role [int] flag specifying the type of info requested (i.e. a title or an icon etc.)

Returns

success [Bool] `True` if the operation worked

flags(*index*)

Set the flag for every index according to the selectable/checkable status of the corresponding node

Parameters

index [`QModelIndex` instance] the index of the node you want data for

Returns

int flags for the index as defined here

index(*row*, *column*, *parentIndex*)

Return the index of the node with row, column and parent

Implementation from https://www.mail-archive.com/pyqt@riverbankcomputing.com/msg19414.html

Parameters

row, column [int] index of the row and the column

parentIndex [`PyQt5.QtCore.QModelIndex`] index to the parent node

Returns

:class:'PyQt5.QtCore.QModelIndex' index of the node

parent(*index*)

Return the index of the parent of the input

Parameters

index [`PyQt5.QtCore.QModelIndex` instance] the index you want the parent of

Returns

:class:'PyQt5.QtCore.QModelIndex' instance the index of the parent (if the node has a parent)

insertRow(*node*, *row*, *parent=<PyQt5.QtCore.QModelIndex object>*)

Insert `node` in a new row after the given row.

Parameters

node [class:*ReductionNode* instance] node to insert

row: int index of the row after which add the `node`

index [`PyQt5.QtCore.QModelIndex` instance] index of the parent

Returns

bool `True` if the insertion succeed, `False` otherwise, whatever exception is raised is logged to the main logger

removeRows(*row*, *count*, *parent=<PyQt5.QtCore.QModelIndex object>*)

Remove `count` rows starting at `row`

Parameters

row: int index of the first row to remove

count: int number of rows to remove

index [`PyQt5.QtCore.QModelIndex` instance] index of the parent

Returns

bool `True` if the removal succeed, `False` otherwise, whatever exception is raised is logged to the main logger

class `vdat.gui.treeview_model.`**ReductionQTreeView**(*parent=None*)

Bases: `PyQt5.QtWidgets.QTreeView`

Custom tree view widget to display the reduction directories

Table 42: **Custom signals**

Name	Signature	Description
sig_selectionChanged	str, str	emitted when a use select a directory; the parameters are the full path to the directory and its type (e.g. `sci`, `cal`, `zro`)
sig_exposure_removed	str	emitted when an exposure is removed; the value is the `expname` from the database

Table 43: **Custom slot**

Name	Signature	Description
set_model()		create a model from the database and set it
on_press()	`PyQt5.QtCore.QModelIndex`	put the path of the selected node into the config file and emit *sig_selectionChanged*
option_menu()	`PyQt5.QtCore.QPoint`	create the actions for the right-click menu
clone_slot()		slot that triggers the cloning of the selected directory
remove_slot()		slot that triggers the removal of the selected directory

Table 44: **Connections between custom signals and/or slots**

Signal	Slot
`customContextMenuRequested`	*option_menu()*
`pressed`	*on_press()*
`triggered` signal of the "Clone" action	*clone_slot()*
`triggered` signal of the "Remove" action	*remove_slot()*
sig_exposure_removed	`vdat.gui.RemoveExposuresMenu.sig_exposure_removed`

Parameters

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

sig_selectionChanged

sig_exposure_removed

set_model()

Create the model and set it.

This method is a pyqt slot.

create_model()

Create the redux directory structure using the information stored in the database

Returns

model [*ReductionTreeviewModel* instance]

keyPressEvent(*event*)

If enter is pressed, trigger the pressed signal with the currently selected index

Parameters

event [`PyQt5.QtGui.QKeyEvent`] event happened

on_press(*index*)
Add the path of the underlying node and emit the `sig_selectionChanged` signal

Parameters

index [`PyQt5.QtCore.QModelIndex` instance] index of the selected directory

option_menu(*position*)
Add an action menu to the tree view

Parameters

position [`PyQt5.QtCore.QPoint` instance] position of the mouse

Returns

menu [`PyQt5.QtWidgets.QMenu`] menu created; mostly for testing purposes

clone_slot()
Slot triggered when the clone action is clicked.

remove_slot()
Slot triggered when the remove action is clicked.

_clone_dir(*index*, *node*)
Clone the directory associated to `node` and add it to the tree view and to the database

Parameters

index [`PyQt5.QtCore.QModelIndex` instance] index of the selected node

node [class:*ReductionNode* instance] selected node

_do_clone_dir(*index*, *node*, *new_name*)

_new_dir_name(*original_name*, *parent_path*)
Ask the user for the new directory name.

Parameters

original_name [string] name of the directory we are about to copy

parent_path [string] path of the parent directory of `original_name`

Returns

new_name [string] name of the new directory

_clone_dialog(*default_text*, *label_prefix=''*)
Create a text dialog with the default text.

Parameters

default_text [string] text set by default in the dialog

label_prefix [string] if not empty added in the line before the standard label

Returns

string text from the dialog

_copy_dir(*parent_path*, *src*, *dst*, *db_entry*)
Copy the directory `src` to `dst`. Both are children of `parent_path`.

Also set to true the `is_clone` entry in the shot file

Parameters

parent_path [string] path where the original and new directories live

src, dst [string] copy `src` into `dst`

db_entry [*vdat.database.models.VDATDir*] database entry for the new directory

_insert_row(*index*, *new_name*)

Clone `node`, update it and insert it

Parameters

index [`PyQt5.QtCore.QModelIndex` instance] index of the selected node

new_name [string] name of the new entry

Returns

bool whether the insertion is successful or not

_remove_dir(*index*, *node*)

Remove the directory associated to `node` and remove it from the tree view and from the database.

Parameters

index [`PyQt5.QtCore.QModelIndex` instance] index of the selected node

node [class:*ReductionNode* instance] selected node

_confirm_remove_dialog(*dir_name*)

Create the dialog to ask if you are sure

Parameters

dir_name [string] name of the directory

Returns

bool where the directory can be removed

showEvent(*event*)

When the *ReductionQTreeView* becomes visible, this method is triggered by Qt.

Parameters

event [`PyQt5.QtGui.QShowEvent`] the tab is show

vdat.gui.utils – PyQt related utilities

Utilities for qt/gui stuff

vdat.gui.utils.**THUMB_DIR = 'thumbs'**

Name of the directory containing thumbnails

vdat.gui.utils.**THUMB_PREFIX = 'thmb_'**

Prefix of the thumbnail images

vdat.gui.utils.**RECON_PREFIX = 'rec_'**

Prefix of the reconstructed images

vdat.gui.utils.**static_directory**()

Return the absolute path of the static directory. Construct it from the location of vdat.gui

Returns

string name of the directory

vdat.gui.utils.**help_collection**()

Return the absolute path of the collection file. If no or more than one files are found return None.

Returns

string name of the file

`vdat.gui.utils.wait_cursor()`

Context manager that replaces the default cursor with a WaitCursor and then reset it upon exiting

class `vdat.gui.utils.FuncQThread(func, *args, **kwargs)`

Bases: `PyQt5.QtCore.QThread`

QThread that accepts a function and its arguments in the constructor and execute it in the `run` method

Parameters

func [callable] function to execute

args [arguments of the function]

kwargs['parent'] [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

kwargs [keyword arguments of the function]

run(*self*)

`vdat.gui.utils.delete_files(paths, matches)`

Loop through all the path and the matches and remove all the files matching each match in each path.

Parameters

paths [list of strings] paths containing the files to remove

matches [list of string] file names or wildcards pattern for removal

`vdat.gui.utils.run_wait_func_qthread(func, *args, **kwargs)`

Start a thread and wait for it to finish. The cursor is set to *`wait_cursor()`* and the thread is marked for deletion before returning

Parameters

msecs [int, optional] if given and not `None` force the application to process events every ‘‘msecs‘ milliseconds; otherwise wait until the thread is done

all : see *`FuncQThread`*

`vdat.gui.utils.get_reconstructed()`

Get a `pyhetdex.het.reconstruct_ifu.QuickReconstructedIFU` object. If the reconstruction fails, `None` will be returned.

The object is created the first time the function is called and then cached. Subsequent calls will return always the same object.

Returns

reconstructed [`pyhetdex.het.reconstruct_ifu.QuickReconstructedIFU`] a newly created or cached reconstructed object

`vdat.gui.utils.rebin_fits(infile, outfile=None, rebin=20, z_indx=None, ext=0)`

Create a thumbnail of the fits `infile`, save and return it.

If it is a datacube, flatten the third axis using a nan-sensitive median before rebinning.

Parameters

infile [string] name of the fits file to use to create the thumbnail

outfile [string, optional] if not `None` save the thumbnail as a fits file

rebin [integer, optional] reduce the number of bins by `rebin`, if smaller or equal to 1, no rebinning happens

z_indx [tuple of two integers, optional] if the input file is a cube, flatten only [z_indx[0], z_indx[1])

ext [int or string, optional] extension number or name to use

Returns

data [numpy.ndarray] rebinned data

`vdat.gui.utils.`**`bin_image`**(*data*, *b*)

Re-bin the input data grouping `b*b` bins. Some pixel can be lost if the size of `data` is not multiple of `b`

Parameters

data [numpy.ndarray] array to rebin

b [int] number of bins to join

Returns

b_data [numpy.ndarray] rebinned data

`vdat.gui.utils.`**`line_separator`**(*parent=None*)

Create and return a vertical line separator

Parameters

parent [`QWidget` or derived instance, optional] the parent of the current widget

Returns

line [`PyQt5.QtWidgets.QFrame`] vertical sunken line

The tab plugins

`vdat.gui.tabs.interface` – The interface

Tab class interface and plugin function prototype

`vdat.gui.tabs.interface.`**`plugin_interface`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Interface of the functions implementing the plugin.

Each tab type must implement a function with the this signature and can be advertised via the `vdat.tab_types` entry point.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

step_name [string] name of the step to which the tab(s) belong

cache [*`vdat.gui.fplane.FplaneCache`* instance] cache object

parent_widget [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tabs

Returns

list instances to be plugged into the *`FplaneWidget`* as tabs; they should be derived from *`FplaneTabTemplate`* or implement the same interface.

class vdat.gui.tabs.interface.**FplaneTabTemplate**(*tab_type*, *parent=None*)

Bases: PyQt5.QtWidgets.QWidget

Template class representing a tab in the focal plane. All the classes should be either derived from this one or implement the same interface

Table 45: **Custom signals**

Name	Signature	Description
sig_ifuToggled	str, bool	emitted when a user selects/deselects an IFU; the parameters are the SLOTID of the IFU and whether the IFU has been selected

Table 46: **Custom slot**

Name	Signature	Description
ifuSelected()	str, bool	selects (second argument True)/deselects (second argument False) an IFU, whose SLOTID is given in the first argument.
selectAllIFUs()		selects all IFUs
deselectAllIFUs()		deselect all IFUs

Parameters

tab_type [str] a name of the tab as advertised in the entry points. When implementing the plugins, care must be taken to pass the correct tab type to allow caching. The tab_type is passed to *plugin_interface()*

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

Attributes

tab_type [str] name of tab type; if required, the instance is cached and should be retrieved under this name. By default is set to the tab_type name passed to *FplaneTabTemplate*.

use_cache [bool] indicate whether the instance can be cached after popping it from the list of tabs. This must be implemented in all derived classes as an attribute or property.

title [string] title to assign to the tab containing the widget. This must be implemented in all derived classes as an attribute or property.

tool_tip [string, optional] if present, is used as the string for the tooltip associated with the current tab

enabled [bool, optional] whether the table is enabled or not. If not present default to True

sig_ifuToggled

cleanup()

Cleanup method called when a tab is popped. The default implementation hides the widget.

ifuSelected(*ifuslot*, *val*)

Select or deselect an IFU. This method is also a PyQt slot.

The current implementation does nothing. Reimplement it to react to the *vdat.gui.fplane.FplaneWidget.sig_ifuSelected* signal. This slot is automatically connected and disconnected when plugging and popping a tab.

Parameters

ifuslot [string] SLOTID of the selected IFU

val [bool] If `True` selected, otherwise deselected

selectAllIFUs()

Select all IFU. This method is also a PyQt slot.

The current implementation does nothing. Reimplement it to react to the *vdat.gui.fplane.FplaneWidget.sig_selectAllIFUs* signal. This slot is automatically connected and disconnected when plugging and popping a tab.

deselectAllIFUs()

Deselect all IFU. This method is also a PyQt slot.

The current implementation does nothing. Reimplement it to react to the *vdat.gui.fplane.FplaneWidget.sig_deselectAllIFUs* signal. This slot is automatically connected and disconnected when plugging and popping a tab.

vdat.gui.tabs.tab_widget – The tab widgets

class vdat.gui.tabs.tab_widget.**UpdateIFUTask**(*parent=None*)

Bases: `PyQt5.QtCore.QThread`

`PyQt5.QtCore.QThread` that runs `ifu.prepare_image` on every of the ifus passed to the `init()` method

Table 47: **Custom signals**

Name	Signature	Description
finished	bool	Emitted when the *run()* is stopped (`True`) or finishes normally (`False`)
ifu_ready	str	after preparing the image(s) to show, emits the signal with the ifu.ifuslot string. The main thread can then retrieve the ifu and display it.

Table 48: **Custom slot**

Name	Signature	Description
stop()		stop and exit from *run()* before finishing the list of IFUs. If the method is processing one IFU, finishes before exiting.

Note: It is not possible to update a pixmap from withing a running `QThread`. Therefore we emit the *ifu_ready* signal when the image is ready for display. It is responsibility of the process owning the thread to pain the image into the pixmap and display it.

Parameters

parent [`QWidget` or derived instance, optional] the parent of the current widget

Attributes

ifus [list] IFUs to process. Each object must implement the *prepare_image()*

finished

ifu_ready

run()

Run the thread. Triggered by calling the `start()`.

Loop through all the `ifus` and call the `prepare_image` method. After it returns, the `ifuslot` string attribute of the processed ifu is emitted via the *ifu_ready* signal.

stop()

Make the *run()* method stop when starting a new loop. This method is also a PyQt slot.

class vdat.gui.tabs.tab_widget.**BaseFplanePanel**(*tab_type*, *parent=None*)

Bases: *vdat.gui.tabs.interface.FplaneTabTemplate*

Implementation of the *FplaneTabTemplate* that creates a working fplane panel using the *ifu_widget.BaseIFUWidget*. Connect all the relevant slots to be able to select/deselect the IFUs.

Table 49: **Custom signals**

Name	Signature	Description
sig_thread_stop		Emitted to stop the `update_thread`

Table 50: **Custom slot**

Name	Signature	Description
update_finished	bool	react to the *UpdateIFUTask.finished* signal. The current implementation does nothing.
ifuSelected(), *selectAllIFUs()*, *deselectAllIFUs()*		see *interface.FplaneTabTemplate*
update_ifus()		If the `update_thread` runs, stop it; then pass to it the IFUs list and start the thread.

Table 51: **Connections between custom signals and/or slots**.

Signal	Slot
UpdateIFUTask.finished	*update_finished()*
UpdateIFUTask.ifu_ready	*show_ifu_image()*
sig_thread_stop	*UpdateIFUTask.stop()*
ifu_widget.BaseIFUWidget.sig_ifuToggled	`sig_ifuToggled`

Parameters

all : see *FplaneTabTemplate*

Attributes

update_thread [*UpdateIFUTask*] QThread used to update the images to show in the IFUs

ifu_widget_class Return the name of the class to use to represent IFUs in the focal plane.

fplane [`FPlane`] instance of the focal plane constructed using the *ifu_widget_class*

main_layout [`PyQt5.QtWidgets.QVBoxLayout`] main layout of the widget.

fplane_layout [`PyQt5.QtWidgets.QGridLayout`] layout containing the IFU widgets

initialized [bool] if `False`, react to *`showEvent()`* drawing the tab content. It is set to `False` at build time and in *`cleanup()`*

others see *`FplaneTabTemplate`*

sig_thread_stop

setup_qthread()

Create and setup the *`UpdateIFUTask`*. Saves it into the `update_thread` attribute.

Overrides this method to use other QThread implementations. The rest of the implementation assumes that `update_thread` has a `start()` method and a `ifus` attibute.

This method is called during the initialisation of the object

build_gui()

Build the GUI calling *`set_main_layout()`* and adding to the main layout the fplane layout from *`build_fplane()`*

set_main_layout()

Create and set the default layout. The layout is saved into the `main_layout`.

Default to `PyQt5.QtWidgets.QVBoxLayout`. Override to use a different layout.

build_fplane()

Create a grid layout, fill it with the IFU widgets returned by *`ifu_widget_class`*, connect the relevant signals and return the layout.

The name of the fplane file is taken from the `fp_filename` option of the `fplane` sectoin in the main vdat configuration file.

Returns

fplane_layout [`PyQt5.QtWidgets.QGridLayout` instance] layout containing the IFU widgets

ifu_widget_class

Return the name of the class to use to represent IFUs in the focal plane.

Override in derived classes to use a different IFU widget.

Returns

:class:'.ifu_widget.BaseIFUWidget'

update_finished(*stopped*)

Do nothing method. This method is also a pyqt Slot.

Override this method in derived classes to act upon the *`UpdateIFUTask.finished`* signal

Parameters

stopped [bool] `True` if the updated has been forcefully stopped, `False` if it finished.

show_ifu_image(*ifuslot*)

Call the `show_image()` of the ifu with `ifuslot` ID.

This method is also an pyqt slot.

Parameters

ifuslot [str] ifuslot ID of the IFU to show

ifuSelected(*ifuslot*, *val*)

Select or deselect an IFU. This method is also a PyQt slot.

Parameters

ifuslot [string] SLOTID of the selected IFU

val [bool] If `True` selected, otherwise deselected

selectAllIFUs()

Select all IFU. This method is also a PyQt slot.

It emits the `sig_ifuToggled` signal to make sure that all the ifus are registered as selected.

deselectAllIFUs()

Deselect all IFU. This method is also a PyQt slot.

It emits the `sig_ifuToggled` signal to make sure that all the ifus are registered as deselected.

showEvent(*e*)

When the tab becomes visible, this method is triggered by Qt. If `initialized` is False, calls *`update_ifus()`* and then set it to True.

Parameters

e [`PyQt5.QtGui.QShowEvent`] the tab is show

update_ifus()

If the `update_thread` is running, stop it. Pass to it the IFUs list and start the thread.

This method is also a PyQt slot.

stop_update_thread()

Stop the thread and wait for it to return

cleanup()

Call the super method, stop the running thread, set `initialized` to False and run the *`cleanup()`* on all the IFUs

class vdat.gui.tabs.tab_widget.**BaseFplanePanelSetup**(*tab_type*, *parent=None*)

Bases: *`vdat.gui.tabs.tab_widget.BaseFplanePanel`*

Add a default setup implementation that set the title and the tooltip and then call the setup method for the IFUs

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations to pass to the tab and to the IFUs.

The method uses directly the following keys of the `tab_dict` argument:

- title (mandatory): title of the current widget; the title is formatted using `format_dict` and saved in `title`.
- tool_tip (optional): tool tip to show for the current widget; if present, the string is formatted using `format_dict` and saved in `tool_tip`.

Other keys might be used by the ifu widget setup method.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [string] dictionary with the fields and values that can be replaced in file names

class vdat.gui.tabs.tab_widget.**FitsFplanePanel**(*tab_type*, *parent=None*)

Bases: *`vdat.gui.tabs.tab_widget.BaseFplanePanelSetup`*

Class that shows fits files in the focal plane. It allows to select the scaling for the IFUs.

Unless documented below, the class inherits signals, slots and connection from `BaseFplanePanel`.

Table 52: **Custom slot**

Name	Signature	Description
`toggle_individual_scale()`	bool	When the input is true, redraw the IFU with individual scaling enabled
`toggle_global_scale()`	bool	When the input is true, redraw the IFU with using a global scaling enabled
`toggle_custom_scale()`	bool, optional	When the input is true, redraw the IFU using a custom scaling enabled

Table 53: **Connections between custom signals and/or slots**.

Signal	Slot
`toggled` of "Individual" radio button	`toggle_individual_scale()`
`toggled` of "Global" radio button	`toggle_global_scale()`
`toggled` of "Custom" radio button	`toggle_custom_scale()` (with bool argument)
`released` of custom "Rescale" button	`toggle_custom_scale()` (with no argument)

Parameters

all : see `BaseFplanePanel`

Attributes

scaling_bar [`PyQt5.QtWidgets.QHBoxLayout`] layout containing the scaling bar

min_scale_custom Float value shown in the minimum scale box.

max_scale_custom Float value shown in the maximum scale box.

min_scale_global, max_scale_global [int] minimum and maximum global scale used when clicking on the global scale button

radio_individual_scale Individual scale radio button

radio_global_scale Global scale radio button

radio_custom_scale Custom scale radio button

ifu_widget_class

Return the name of the class to use to represent IFUs in the focal plane.

Returns

:class:'.ifu_widget.IFUFitsWidget'

build_gui()

Add extra functionalities to the basic gui

add_scaling_bar()

Create a `PyQt5.QtWidgets.QHBoxLayout`, add it to the `main_layout`. On this layout add buttons and text boxes to chose the scaling to use for the IFUs. Connect all the relevant signals and slots.

_radio_button(*label*, *layout*)

Create a radio button, add it to `layout` and return it. The buttons are not marked by default.

Parameters

label [string] label of the button

layout [`PyQt5.QtWidgets.QLayout`] layout to which the button is added

Returns

button [`PyQt5.QtWidgets.QRadioButton`]

`_scale_box`()

Create and return a box where is possible to insert numbers.

Returns

scale_box [`PyQt5.QtWidgets.QLineEdit`] box with width of 80 and a double validator

`_custom_scale_setter`(*layout*)

Setup all the parts needed to set the custom scale.

Parameters

layout [`PyQt5.QtWidgets.QLayout`] layout to which the pieces are added

`custom_scale_set_enabled`(*enable*)

Enable/disable the labels, input boxes and button used to set the custom scaling.

Parameters

enable [bool] whether to enable or disable the widgets.

`min_scale_custom`

Float value shown in the minimum scale box. Clear the text box with `del min_scale_custom`

`max_scale_custom`

Float value shown in the maximum scale box. Clear the text box with `del max_scale_custom`

`radio_individual_scale`

Individual scale radio button

`radio_global_scale`

Global scale radio button

`radio_custom_scale`

Custom scale radio button

`toggle_individual_scale`(*enabled=True*)

When `enabled` is `True` redraw the IFUs using the individual scaling.

This method is also a PyQt slot.

Parameters

enabled [bool] whether the individual scaling is enabled or not

`toggle_global_scale`(*enabled*)

When `enabled` is `True` redraw the IFUs using the global scaling, saved in `min_scale_global` and `max_scale_global`.

This method is also a PyQt slot.

Parameters

enabled [bool] whether the global scaling is enabled or not

`toggle_custom_scale`(*enabled=True*)

When `enabled` is `True` redraw the IFUs using the minimum and maximum values provided by the user.

This method is also a PyQt slot.

Parameters

enabled [bool] whether the custom scaling is enabled or not

_zscale_to_ifus(*zmin*, *zmax*)

Loop over all the IFUs setting `ifu.zmin_global` and `ifu.zmax_global` to the input values.

Parameters

zmin, zmax [float or None] values to set

reset_individual_scale()

Reset the individual scale to checked

update_finished(*stopped*)

If the update of the IFUs finished without being stopped, collect all the zmin/zmax from the IFUs and save it for the global scaling. If none of the IFUs has such values disable the global scaling button.

This method is also a pyqt Slot.

Override this method in derived classes to act upon the *UpdateIFUTask.finished* signal

Parameters

stopped [bool] `True` if the updated has been forcefully stopped, `False` if it finished.

get_global_scale()

Try to get the zmin and zmax from the IFUs.

If none of the IFU have those values, disable the global button. Otherwise save the values in the `zmin_global` and `zmax_global` and add their values into a tool tip and to the text boxes if they are empty

cleanup()

Call the parent class implementation and unset all the min/max scales

class vdat.gui.tabs.tab_widget.**QuickReconFplanePanel**(*tab_type*, *parent=None*)

Bases: *vdat.gui.tabs.tab_widget.FitsFplanePanel*

Do a quick reconstruction with the input files

Unless documented below, the class inherits signals, slots and connection, as well as attributes from *FitsFplanePanel*.

Parameters

all : see *BaseFplanePanel*

Attributes

enabled [bool] if the reconstruction object exists, is set to True, else to False

ifu_widget_class

Return the name of the class to use to represent IFUs in the focal plane.

Returns

:class:'.ifu_widget.IFUQuickReconWidget'

basenames(*basenames*)

Set the basenames in all the ifus

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations to pass to the tab and to the IFUs.

See `FitsFplanePanel.setup()` for the keys of `tab_dict` used here.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [string] dictionary with the fields and values that can be replaced in file names

cleanup()

On top of what *FitsFplanePanel.cleanup()* does, reset `enabled` to True

class vdat.gui.tabs.tab_widget.**FitsAndReconFplanePanel**(*tab_type*, *parent=None*)

Bases: *vdat.gui.tabs.interface.FplaneTabTemplate*

This class provides a tab grouping a *FitsFplanePanel* and a *QuickReconFplanePanel* and providing a button to switch between them.

Unless documented below, the class inherits signals, slots and connection from *interface.FplaneTabTemplate*.

Table 54: **Custom slot**

Name	Signature	Description
next_widget()		go to the next widget

Table 55: **Connections between custom signals and/or slots**.

Signal	Slot
"Reconstructed" checkbox added in the lower right corner of the widgets	*next_widget()*

Parameters

all : see *FplaneTabTemplate*

Attributes

others see *FplaneTabTemplate*

_make_widget(*Widget*, *tab_type*, *is_checked*)

Create a `Widget` instance, connect the `sig_ifuToggled`, add a check box with "Reconstructed" as text and connect it to a *next_widget()*.

Parameters

Widget [class derived from *BaseFplanePanel*] the widget to create

tab_type [string] name of the tab

is_checked [bool] whether the checkbox needs to be always checked or not

Returns

w [instance of Widget]

button [`PyQt5.QtWidgets.QPushButton`] button added to the Widget

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations to pass to the stacked widgets and to the IFUs.

Make sure that the *FitsFplanePanel* is always shown first. If the reconstruction object does not exist, disable the checkbox to switch to the reconstructed widget. The title and tool tip, if present, are taken from the first widget.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [string] dictionary with the fields and values that can be replaced in file names

cleanup()
Cleanup the widgets

next_widget()
Switch to the next widget.

This method is also a Qt slot.

ifuSelected(*ifuslot*, *val*)
Select or deselect an IFU. This method is also a PyQt slot.

Call the corresponding method in the stacked widgets.

Parameters

ifuslot [string] SLOTID of the selected IFU

val [bool] If `True` selected, otherwise deselected

selectAllIFUs()
Select all IFU. This method is also a PyQt slot.

Call the corresponding method in the stacked widgets.

deselectAllIFUs()
Deselect all IFU. This method is also a PyQt slot.

Call the corresponding method in the stacked widgets.

class vdat.gui.tabs.tab_widget.**CubeFplanePanel**(*tab_type*, *parent=None*)
Bases: *vdat.gui.tabs.tab_widget.FitsFplanePanel*

Class that shows data cube fits files in the focal plane.

Unless documented below, the class inherits signals, slots and connection from *FitsFplanePanel*.

Parameters

all : see *FitsFplanePanel*

Attributes

all : see *FitsFplanePanel*

ifu_widget_class
Return the name of the class to use to represent IFUs in the focal plane.

Returns

:class:'.ifu_widget.IFUCubeWidget'

class vdat.gui.tabs.tab_widget.**MultiExtFplanePanel**(*tab_type*, *parent=None*)
Bases: *vdat.gui.tabs.tab_widget.FitsFplanePanel*

Class that shows one desired extension from fits files in the focal plane.

Unless documented below, the class inherits signals, slots and connection from *FitsFplanePanel*.

Parameters

all : see *FitsFplanePanel*

Attributes

all : see *FitsFplanePanel*

ifu_widget_class

Return the name of the class to use to represent IFUs in the focal plane.

Returns

:class:'.ifu_widget.IFUMultiExtWidget'

class vdat.gui.tabs.tab_widget.**StaticCheckBox**(*text*, *is_checked*, *parent=None*)

Bases: PyQt5.QtWidgets.QCheckBox

A checkbox always checked or un-checked

Table 56: **Custom slot**

Name	Signature	Description
stay_checked()		Keep the "checked" state of the button always the same

Table 57: **Connections between custom signals and/or slots**.

Signal	Slot
clicked	*stay_checked()*

Parameters

text [string] text showing in the button

is_checked [bool] whether the checkbox must stay checked or not

parent [QWidget or derived instance, optional] the parent of the current widget

stay_checked(*state*)

Force the checkbox to stay checked or unchecked.

This method is also a Qt Slot.

class vdat.gui.tabs.tab_widget.**TextFilesPanel**(*tab_type*, *parent=None*)

Bases: *vdat.gui.tabs.tab_widget.BaseFplanePanelSetup*

Tab to show the content of a text file.

The IFU in the focal plane shows the number of lines in the file. A double click on the widget brings up a window with the content of the file

Unless documented below, the class inherits signals, slots and connection from *BaseFplanePanel*.

Parameters

all : see *FplaneTabTemplate*

ifu_widget_class

Return the class to use to represent IFUs in the focal plane.

Returns

:class:'.ifu_widget.TextFileWidget'

class vdat.gui.tabs.tab_widget.**DistPanel**(*tab_type*, *parent=None*)

Bases: *vdat.gui.tabs.tab_widget.BaseFplanePanelSetup*

Show the number of lines in the distortion file in the IFUs. A double click on the widget brings up a window with the content of the file. From the window, it is possible to send a region file computed from the distortion to the ds9.

Unless documented below, the class inherits signals, slots and connection from `BaseFplanePanel`.

Parameters

all : see `FplaneTabTemplate`

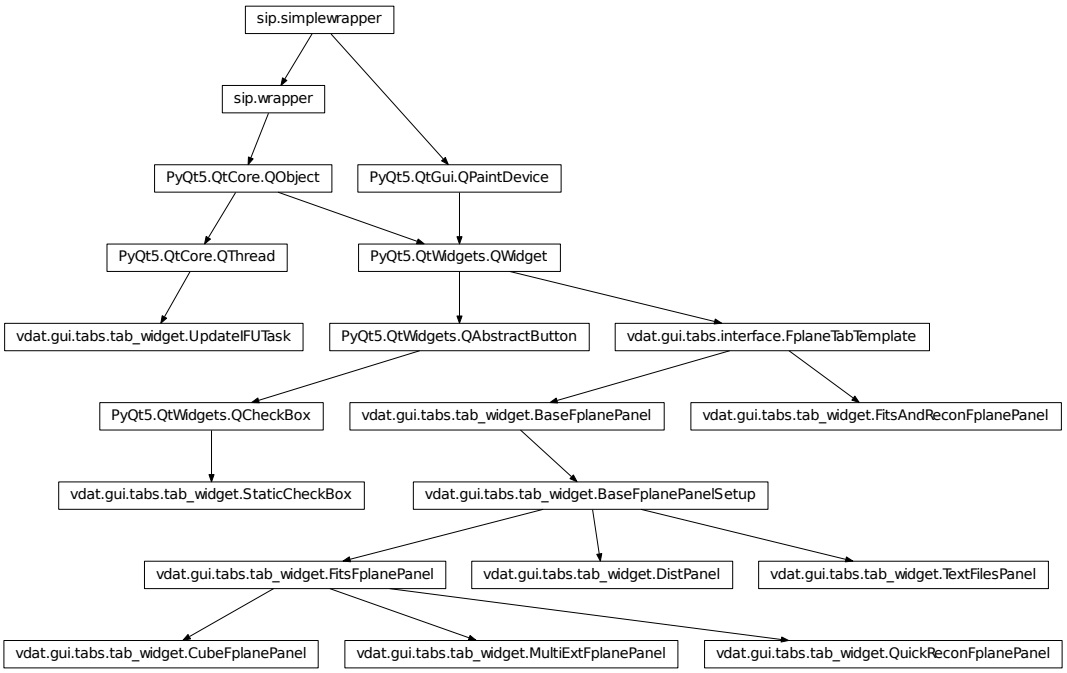
`ifu_widget_class`

Return the class to use to represent IFUs in the focal plane.

Returns

:class:‘.ifu_widget.BaseIFUWidget‘

Inheritance scheme



`vdat.gui.tabs.ifu_widget` – The widget representing one IFU

Widgets showing the IFU in the focal plane

`vdat.gui.tabs.ifu_widget.`**`update_painter_font_size`**(*painter*, *rect*, *string*)

If the string is too large, reduce the font. The code is taken from https://stackoverflow.com/questions/2202717/for-qt-4-6-x-how-to-auto-size-text-to-fit-in-a-specified-width

Parameters

painter [`PyQt5.QtGui.QPainter`] painter object

rect [`PyQt5.QtCore.QRectF`] rectangle in which the string is embedded

string [string] string to paint

Returns

painter [`PyQt5.QtGui.QPainter`] updated painter

class `vdat.gui.tabs.ifu_widget.`**`BaseItem`**(***kwargs*)

Bases: `object`

An editable version of `collections.namedtuple`. The name and number of attributes is fixed and the values are set to None by default.

Derived classes must only define `__slots__` which should only contain names of any additional slots. For more information see the documenatation

Attributes

n_col, n_row [int] index of the column and of the row of the item

col_name, row_name [string] name associated to the column and row

format_dict [dict] dictionary with the fields and values that can be replaced in file names

image [`PyQt5.QtGui.QImage` or `None`] the thumbnail to display; set it to `None` if there is nothing to display

col_name

format_dict

image

n_col

n_row

row_name

class `vdat.gui.tabs.ifu_widget.`**`FitsItem`**(***kwargs*)

Bases: *`vdat.gui.tabs.ifu_widget.BaseItem`*

Items used in the widgets showing fits file

Attributes

fname [str] full file name of the file to show

data [nd.array] array representing the image

zmin, zmax [float] minimum and maximum value of data after applying zscaling

ctime [float] time of creation of the thumbnail stored in `image`

ctime

data

fname

zmax

zmin

class `vdat.gui.tabs.ifu_widget.`**`DistItem`**(***kwargs*)

Bases: *`vdat.gui.tabs.ifu_widget.BaseItem`*

Items used when showing the distortion. *`fits_names`* is initialized to an empty list.

Attributes

fname [str] full file name of the distortion file

reg_fname [str] full name of the region file

fits_names [list strings] name of the fits files to load into DS9 to use as base to display the regions

fits_names

fname

reg_fname

class vdat.gui.tabs.ifu_widget.**BaseIFUWidget**(*ifuslot*, *x*, *y*, *specid*, *specslot*, *ifuid*, *ifurot*, *platescl*, *parent=None*)

Bases: pyhetdex.het.fplane.IFU, PyQt5.QtWidgets.QLabel

Base class representing one IFU in the focal plane.

This implement the basic functionalities that other IFU widgets have and all the hooks needed to make this work together with *BaseFplanePanel*.

Upon initialization it:

- creates tooltip and whatsThis
- sets the default style
- customizes single and double click behaviour

Table 58: **Custom signals**

Name	Signature	Description
sig_ifuToggled	str, bool	emitted when a user selects/deselects the current IFU; the parameters are the SLOTID of the IFU and whether the IFU has been selected

Table 59: **Custom slot**

Name	Signature	Description
ifuSelected(str, bool)		selects (second argument True)/deselects (second argument False) the IFU, if its SLOTID is the one given in the first argument.

Parameters

ifuslot [string] id of the slot in the focal plane

x, y [string or float] x and y position of the ifuslot in the focal plane

specid, specslot: int id of the spectrograph where the ifu is plugged into and of the slot of the spectrograph

ifuid [string] id of the ifu

platescl [float] focal plane plate scale at the position in the IHMP

parent [QWidget or derived instance, optional] the parent of the current widget

Attributes

hw_size Return the smaller between width and height of the widget

selected Whether the IFU is selected or not.

empty_img [`PyQt5.QtGui.QImage`] empty image, black with a white cross

current_img [`PyQt5.QtGui.QImage`] image to show in the gui. The desired image should be assigned to this variable in *`prepare_image()`* and shown in *`show_image()`*.

default style attibutes see *`_default_aspect()`*

sig_ifuToggled

_default_aspect()

Set the default aspect of the Widget.

Attributes

border_style_selected, border_style_unselected [int]

defaultsize [int] default to 42

_default_infos()

Create and set the default tooltip and whatsThis message

_create_empty_image()

Create an empty image, a white cross on black, and save it into the `empty` attribute.

hw_size

Return the smaller between width and height of the widget

mouseDoubleClickEvent(*event*)

Make sure that a double click is not interpreted as a two single clicks. Beside this doesn't do anything else.

Qt don't deal with single and double click, so we do it here. If you overrides either of *`mouseReleaseEvent()`* or *`mouseDoubleClickEvent()`*, make sure to execute the parent class implementation as first thing to avoid unexpected results.

Parameters

event [`PyQt5.QtGui.QMouseEvent`]

mouseReleaseEvent(*event*)

On a user clicking on the button, swap the logical value of the *`selected`* attribute and change the look of the button to reflect this.

Qt don't deal with single and double click, so we do it here. If you overrides either of *`mouseReleaseEvent()`* or *`mouseDoubleClickEvent()`*, make sure to execute the parent class implementation as first thing to avoid unexpected results.

Parameters

event [`PyQt5.QtGui.QMouseEvent`]

_on_timer_timout()

Selected or deselect the IFU.

This method is also a PyQt slot.

selected

Whether the IFU is selected or not.

prepare_image()

Prepare the image to show in the gui saving it into the `current_img`. The base implementation saves the empty image `empty_img` into `current_img`.

Subclasses can override this method and should call it to make sure that sensible defaults are set

show_image()
Create pixmap using the `current_img`, save it under `pixmap_img` and set it. Then update the widget to show it.

cleanup()
Cleanup method. Create the empty image and paint it.

class vdat.gui.tabs.ifu_widget.**IFUSplitWidget**(**args*, ***kwargs*)
Bases: *vdat.gui.tabs.ifu_widget.BaseIFUWidget*

Base IFU widget for all the cases in which the image displayed is composed of multiple elements.

This class implements boilerplate code to setup a list of items to show, with their position, and to display them.

Parameters

all same as *BaseIFUWidget*

Attributes

all same as *BaseIFUWidget*

thumb_items [list] list of *th_item*, initialized to an empty list by *setup()*

target_dir [string] the selected directory

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

n_rows, n_cols [int] number of rows and columns to show in the IFU

th_item Class representing one of the items to display in the IFU.

rect_thumbnail Rectangle used to add the thumbnails to the IFU.

th_item
Class representing one of the items to display in the IFU.

Returns

:class:'BaseItem' thumbnail item

setup(*target_dir*, *tab_dict*, *format_dict*)
Set `target_dir`, `tab_dict`, `n_cols` and `n_rows` and fill the `thumb_items` list.

The method uses directly the following keys of the `tab_dict` argument:

- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file. If not given they default to a list with one empty string. `n_cols` and `n_rows` attributes are set to the number of elements in the two variables.

Each element of the `thumb_items` list is an instance of *th_item* with:

- `n_row` and `n_col` set to the position of the item;
- `row_name` and `col_name` set to the names of the row and column (from the configuration as described above) associated to the element;
- `image` is the image of the item to show in the quadrant and is initialized to `None`;
- `format_dict` is a dictionary containing the following entries:
  - all the elements in the input parameter `format_dict`
  - `ifuslot`, `ifuid`, `specid`: ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
  - `col`, `row`: replaced with each of the elements in the `cols` and `rows` configuration options.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [dictionary] dictionary with extra fields and values that can be used to e.g. format file names

rect_thumbnail

Rectangle used to add the thumbnails to the IFU. It uses the values of `n_cols` and `n_rows` set in *setup()* to divide the widget in equal areas.

upper_left_qpoint(*i*, *j*)

Return the upper left point to use to add thumbnails to the IFU widget given `n_cols` columns and `n_rows` rows.

Parameters

i, j [int] column/row index of the image to add

Returns

:class:'PyQt5.QtCore.QPointF' position of the upper left corner of the rectangle

load_thumbnail(*item*)

Create the thumbnail to be displayed in the IFU. If no image can be created, set `item.image` to `None`. Otherwise create a `PyQt5.QtGui.QImage` and assign it.

This implementation returns the input `item` unmodified.

Parameters

item [*BaseItem* or child] item representing one file to display

Returns

item [*BaseItem* or child] updated item with the `image`

prepare_image()

Create the images to show in the GUI.

Invoke the parent class method, to create empty images, then loop through the elements of `thumb_items`, for each calls *load_thumbnail()*. Then filter the items for which the `th_item.image` is `None`. If nothing remains, do nothing, otherwise add the images in the IFU and replace the empty image. The size of the displayed image is *rect_thumbnail* and the position is given by *upper_left_qpoint()*. If some image is missing the corresponding quadrant it is left black.

Derived classes probably don't need to override this method, but the ones called from here, notably *load_thumbnail()*.

cleanup()

Remove the thumbnail from the gui and clear the `thumb_items` list

class vdat.gui.tabs.ifu_widget.**IFUFitsWidget**(**args*, ***kwargs*)

Bases: *vdat.gui.tabs.ifu_widget.IFUSplitWidget*

A custom class designed to contain one or more fits files for the IFU.

Parameters

all same as *IFUSplitWidget*

Attributes

all same as *IFUSplitWidget*

zmin_ifu Returns the minimum zmin for the fits files shown in the widget,

zmax_ifu Returns the maximum zmax for the fits files shown in the widget,

zmin_global, zmax_global [float] global zmin and zmax. If they are both not `None`, they are used instead of the individual zscale values. On cleanup they are reset to `None`

files_for_window Return the name of the files to pass to the fits viewer window.

titles_for_window Return the tab titles for the files passes to the fits viewer window.

tooltips_for_window Return the tab tool tips for the files passes to the fits viewer window.

th_item

Item to display in the IFU.

Returns

:class:'FitsItem' thumbnail item

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations needed to create the images to show.

After executing the corresponding method of the base class, fills the `th_item.fname` attributes using `th_item.format_dict`. It also ensures that the thumbnail directory gets created (with *ensure_thumb_dir()*)

Besides what the parent method needs, this method uses directly the following keys of the `tab_dict` argument:

- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [string] dictionary with the fields and values that can be replaced in file names

ensure_thumb_dir(*target_dir*)

If necessary creates the thumbnail directory *THUMB_DIR* in `target_dir`

Parameters

target_dir [string] directory selected by the user

load_thumbnail(*item*)

Create, if necessary, and load the thumbnail. To avoid reloading it multiple times, store the data into `item.data` and its min/max into `item.zmin` and `item.zmax`.

Steps:

- *get_thumb()*: if possible creates the thumbnail and returns its name; if not possible, return None and set the above elements of `item` are set to `None`;
- if the file name is returned and the file is newer than the stored version, reloads the thumbnail and recompute `zmin` and `zmax`;
- if there is a thumbnail, create a `PyQt5.QtGui.QImage` with the data, using either the image or the global `zmin`/`zmax`, and store it into `item.image`

Derived classes probably don't need to override this method, but the ones called from here, notably *get_thumb()* and *file_ctime()*.

Parameters

item [*th_item*] item representing one file to display

Returns

item [*th_item*] updated item with the `data`, `image`, `zmin` and `zmax` attributes filled

get_thumb (*fname*)

Create the thumbnail from the file name.

It gets the name of the thumbnail file using *thumb_name()* and the creation times of the input file and the thumbnail using *file_ctime()*. If `fname` does not exist, it makes sure that there is also no thumbnail file, to make sure that files are not shown after being removed. If `fname` exists and/or is newer than the thumbnail (re)create it using *create_thumb()*

Parameters

fname [string] name of the file for which the thumbnail is to be created

Returns

thumb_name [string] name of the thumbnail file, or None if no thumbnail is created

create_thumb (*fname*, *thumb_name*)

Create the thumbnail from `fname`.

Parameters

fname [string] name of the file for which the thumbnail is to be created

thumb_name [string] name of the file where to save the thumbnail

thumb_name (*fname*)

Create the name to use for the thumbnail, prepending *THUMB_DIR*/*THUMB_PREFIX* to the file name

Parameters

fname [string] name of the file for which the thumbnail is to be created

Returns

string name of the thumbnail file

file_ctime (*fname*)

Get the ctime from the file.

Parameters

fname [string] name of the file for which the thumbnail is to be created

Returns

float time of creation of a file or None if the file does not exist

cleanup ()

Remove the thumbnail from the gui and clear the `thumb_items` list

zmin_ifu

Returns the minimum zmin for the fits files shown in the widget, or None if no file is shown

zmax_ifu

Returns the maximum zmax for the fits files shown in the widget, or None if no file is shown

mouseDoubleClickEvent(*event*)

On double click open a popup window of type `ifu_viewer.FitsViewerWindow` with details on the selected IFU.

The list of files passed to the window is taken from `files_for_window`. If the list is empty no window will be shown. The new window tab titles and tooltips are taken from `titles_for_window` and `tooltips_for_window`. Derived classes can overrides these three properties to give the appropriate file and tabs names and tooltips.

Parameters

event [`PyQt5.QtGui.QMouseEvent`]

files_for_window

Return the name of the files to pass to the fits viewer window. The file names are created in `setup()`, stored in `thumb_items` and returned only if they exist.

Returns

flist [list of string] file names to plug display in the fits viewer window

titles_for_window

Return the tab titles for the files passes to the fits viewer window.

It must return either `None` or a list of strings with the same length of `files_for_window`.

Returns

"None"

tooltips_for_window

Return the tab tool tips for the files passes to the fits viewer window.

It should return either `None` or a list of strings with the same length of `files_for_window`.

Returns

"None"

class vdat.gui.tabs.ifu_widget.**IFUQuickReconWidget**(**args*, ***kwargs*)

Bases: `vdat.gui.tabs.ifu_widget.IFUFitsWidget`

Create IFU reconstructed images

Parameters

all same as `IFUFitsWidget`

Attributes

all same as `IFUFitsWidget`

enabled [bool] whether the reconstruction is disabled or not, i.e. if the reconstructed object returned by `vdat.gui.utils.get_reconstructed()` returns None or not

basenames List of strings used as basenames.

basenames

List of strings used as basenames. If the property is present, setup will loop through `cols`, `rows` and `basenames` and replace the `{basename}` placeholder in the file names by each of the elements in this property, shadowing the placeholder in `setup()`'s `format_dict`, if present. If the property is not set, the loop over the basenames is not done in `setup()`.

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations needed to create the reconstructed image to show.

The information needed to build the ifu reconstructed image are stored into the `thumb_items`. The `fname` attribute from *IFUFitsWidget.th_item* is used to store a list of file names needed for the reconstruction.

The method uses the keys in `tab_dict` and provide the formatting entries as described in *IFUFitsWidget.setup()*. It overrides the `basename` formatting entry as described in *IFUQuickReconWidget.basenames*. `n_cols` and `n_rows` attributes are set to one.

It also ensures that the thumbnail directory is created.

Parameters

all : as in *IFUFitsWidget.setup()*

get_thumb (*fname*)

Create the reconstructed image from the file names.

This method is almost identical to *IFUFitsWidget.get_thumb()* except that it doesn't attempt to create thumbnails if the quick reconstruction object is not present. However permits to show it if available.

Todo: Replicating code is in general a bad idea, but I haven't found yet a better solutions. This method implementation needs some re-thinking.

Parameters

fname [list of string] names of the file for which the reconstructed image is to be created

Returns

thumb_name [string] name of the thumbnail file, or None if no thumbnail is created

create_thumb (*fnames*, *thumb_name*)

Create the reconstructed image from the names and save it

Parameters

fnames [list of string] names of the file for which the reconstructed image is to be created

thumb_name [string] name of the file where to save the thumbnail

thumb_name (*fnames*)

Alias of *reconstructed_name()*

reconstructed_name (*file_names*)

Creates the name to use to store the reconstructed image, as a md5 hash of the `file_names`. This way we can get a unique name for every combination of files.

Parameters

file_names [list of strings] name of the files to use to create the thumbnail

Returns

string name of the reconstructed file

file_ctime (*fname*)

Get the ctime from the file. If fname is a list, get the newest time.

Parameters

fname [string or list of strings] name of the file for which the thumbnail is to be created

Returns

float newest time of creation (ctime) of a file or None if the file does not exist

files_for_window
Return the name of the reconstructed file that should go into the fits viewer window.

If the file does not exist, try to create it. If it fails, return an empty list.

Returns

flist [list of string] file name to plug display in the fits viewer window

titles_for_window
The title of the reconstructed tab in the fits window is just "Reconstructed"

tooltips_for_window
The tooltip show all the files making up the reconstructed image

cleanup()
Cleanup the widget as in *IFUFitsWidget()*, plus delete the basenames

class vdat.gui.tabs.ifu_widget.**IFUCubeWidget**(**args*, ***kwargs*)
Bases: *vdat.gui.tabs.ifu_widget.IFUFitsWidget*

Display thumbnails from data cubes instead of from 2D fits files.

This widget is very similar to the *IFUFitsWidget*

Unless documented below, the class inherits signals, slots and connection from *IFUFitsWidget*.

Parameters

all : see *IFUFitsWidget*

Attributes

all : see *IFUFitsWidget*

setup(*target_dir*, *tab_dict*, *format_dict*)
Informations needed to create the images to show.

The widget uses the same `tab_dict` keys as *IFUFitsWidget* and the same formatting entries. It also accept the following key of the `tab_dict` argument:

- `z_indx` (optional): before creating the thumbnail for the data cubes, the image is compressed along the z-dimension using the median; if `z_indx` is not given, it uses the whole range, otherwise it uses only the part of the cube in the range [z_indx[0], z_indx[1])

Parameters

all : same as *IFUFitsWidget.setup()*

create_thumb(*fname*, *thumb_name*)
Create the thumbnail from `fname`.

Parameters

fname [string] name of the file for which the thumbnail is to be created

thumb_name [string] name of the file where to save the thumbnail

thumb_name(*fname*)
Create the name to use for the thumbnail, prepending *THUMB_DIR*/*THUMB_PREFIX* to the file name. If `z_indx` is given, add it to the thumbnail name

Parameters

fname [string] name of the file for which the thumbnail is to be created

Returns

string name of the thumbnail file

class vdat.gui.tabs.ifu_widget.**IFUMultiExtWidget**(**args*, ***kwargs*)

Bases: *vdat.gui.tabs.ifu_widget.IFUFitsWidget*

Display thumbnails from one of the extensions of a multi-extension fits file

This widget is very similar to the *IFUFitsWidget*

Unless documented below, the class inherits signals, slots and connection from *IFUFitsWidget*.

Parameters

all : see *IFUFitsWidget*

Attributes

all : see *IFUFitsWidget*

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations needed to create the images to show.

The widget uses the same `tab_dict` keys as *IFUFitsWidget* and the same formatting entries. It also accept the following key of the `tab_dict` argument:

- `ext` (int or string): index or number of the extension to display in IFU

Parameters

all : same as *IFUFitsWidget.setup()*

create_thumb(*fname*, *thumb_name*)

Create the thumbnail from `fname`.

Parameters

fname [string] name of the file for which the thumbnail is to be created

thumb_name [string] name of the file where to save the thumbnail

thumb_name(*fname*)

Create the name to use for the thumbnail, prepending *THUMB_DIR*/*THUMB_PREFIX* to the file name. If `z_indx` is given, add it to the thumbnail name

Parameters

fname [string] name of the file for which the thumbnail is to be created

Returns

string name of the thumbnail file

class vdat.gui.tabs.ifu_widget.**TextFileWidget**(*ifuslot*, *x*, *y*, *specid*, *specslot*, *ifuid*, *ifurot*, *platescl*, *parent=None*)

Bases: *vdat.gui.tabs.ifu_widget.BaseIFUWidget*

Widget that paints the number of lines in the widget. On double click open a window to show the file content.

Parameters

all same as *BaseIFUWidget*

rect_thumbnail

Rectangle with the size of the thumbnails of the IFU. Use default values

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations needed to create the images to show.

The method uses directly the following keys of the `tab_dict` argument:

- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.

The following formatting entries are explicitly used in this method:

- `ifuslot`, `ifuid`, `specid`: ID of the slot, of the IFU bundle and of the spectrograph it is connected to.

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [string] dictionary with the fields and values that can be replaced in file names

prepare_image()

Create the image to show in the GUI.

If the text file, whose name is created in *setup()*, exists, paint the number of lines of the file.

mouseDoubleClickEvent(*event*)

On double click, if the text file exists, open a window of type *ifu_viewer.FileEditorWindow*.

Parameters

event [`PyQt5.QtGui.QMouseEvent`]

class vdat.gui.tabs.ifu_widget.**DistWidget**(**args*, ***kwargs*)

Bases: *vdat.gui.tabs.ifu_widget.IFUSplitWidget*

Widget showing the distortion file and the corresponding region file created with `$CUREBIN/distview`

Parameters

all same as *IFUSplitWidget*

Attributes

all same as *IFUSplitWidget*

th_item

Item to display in the IFU.

Returns

:class:'DistItem' thumbnail item

setup(*target_dir*, *tab_dict*, *format_dict*)

Informations needed to create the images to show.

After executing the corresponding method of the base class, fills the `th_item.fname`, `th_item.reg_fname' and :attr:`th_item.fits_names` attributes using `th_item.format_dict`.

Besides what the parent method needs, this method uses directly the following keys of the `tab_dict` argument:

- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.

- `fits_names` (mandatory): list of names of the fits files to use then displaying the distortion in DS9. If the list is empty, it is not possible to display the data on DS9. Use the same formatting as `file_name` ``

Parameters

target_dir [string] directory selected by the user

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

format_dict [string] dictionary with the fields and values that can be replaced in file names

load_thumbnail(*item*)

Create, if necessary, the region file and create the thumbnail to display.

Steps:

- *get_region()*: if possible creates the region file and returns whether it exists or not;
- if the distortion and/or the region file exists, create an image with the number or lines in either file and save it into `item.image`

Parameters

item [*th_item*] item representing one file to display

Returns

item [*th_item*] updated item with the `image` attribute filled

get_region(*item*)

Create the region file using `$CUREBIN/distview`.

Steps:

- if the distortion exists, but not the region file, create the latter;
- if the distortion exists and is newer than the region file, recreate the latter;
- if the distortion file does not exist but the region file does not, remove the region file;
- if none exists, do nothing.

Parameters

item [*th_item*] item representing one file to display

Returns

dist_exists, reg_exists [bool] whether the distortion and the region file exist

_run_distview(*fname*, *reg_fname*)

Run `$CUREBIN/distview` to create the region file

Parameters

fname, reg_fname [string] name of the distortion and the region files

Returns

success [bool] whether the operation is successful

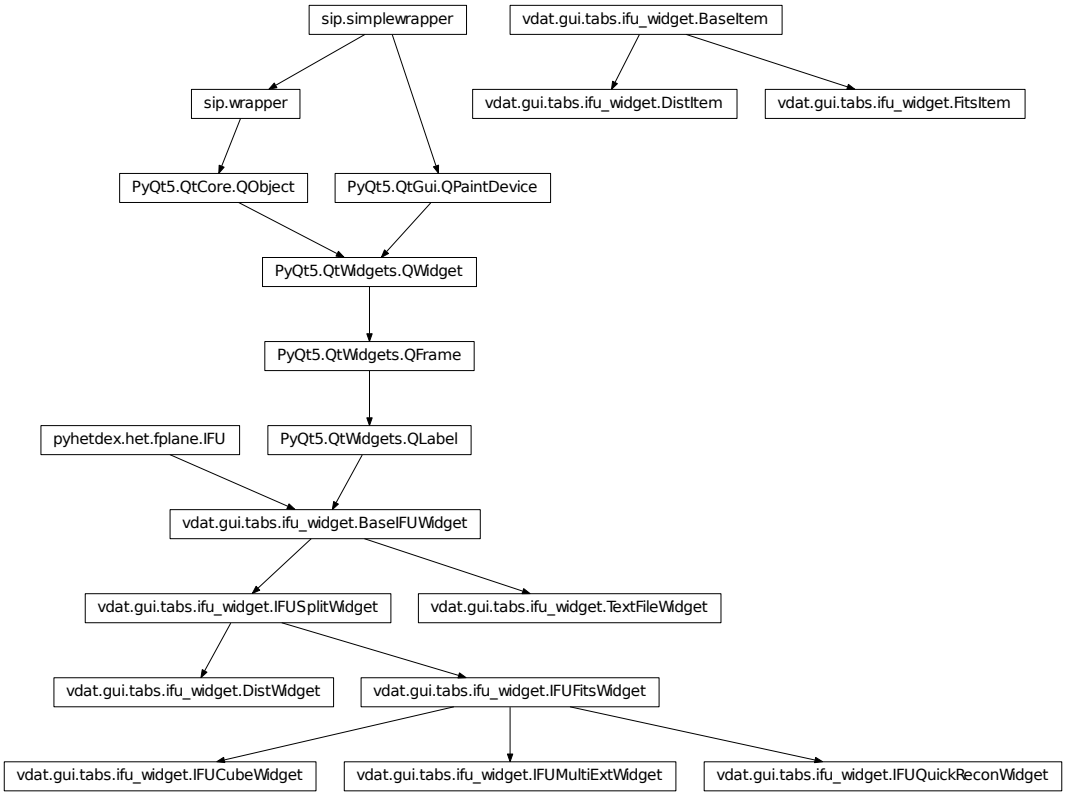
mouseDoubleClickEvent(*event*)

On double click, if the distortion files exist, open a window of type *ifu_viewer.DistWindow*.

Parameters

event [PyQt5.QtGui.QMouseEvent]

Inheritance diagram



vdat.gui.tabs.ifu_viewer – The IFU viewer windows

exception vdat.gui.tabs.ifu_viewer.**WindowViewerError**

Bases: Exception

Base exception for the viewer classes

exception vdat.gui.tabs.ifu_viewer.**FitsViewerTitleTooltipError**

Bases: *vdat.gui.tabs.ifu_viewer.WindowViewerError*, TypeError

Exception raised when the titles and/or tooltips for the fits viewer are malformed

exception vdat.gui.tabs.ifu_viewer.**DistViewerFileNumberError**

Bases: *vdat.gui.tabs.ifu_viewer.FitsViewerTitleTooltipError*

Exception raised when the number of files, the titles and/or tooltips for the dist viewer are malformed

class vdat.gui.tabs.ifu_viewer.**DS9Menu**(*title='ds9'*, *new_ds9_name=None*, *new_ds9_menu=True*, *parent=None*)

Bases: PyQt5.QtWidgets.QMenu

Menu item with the ds9 actions.

Table 60: **Custom signals**

Name	Signature	Description
sig_ds9	pyds9.DS9	emitted, with the ds9 instance to use, when clicking on a ds9 menu option

Table 61: **Custom slot**

Name	Signature	Description
populate_ds9_menu()		dynamically create the ds9 menu
pyds9_error_popup()		launch an pop up window with the explanation of the pysd9 import failure
launch_ds9()	PyQt5.QtWidgets.QAction, optional	Launch DS9. If no action is passed, create a new window, otherwise add a new frame into an existing window; then emit *sig_ds9*

Table 62: **Connections between custom signals and/or slots**.

Signal	Slot
aboutToShow	*populate_ds9_menu()*
'There was a problem loading pyds9.' menu entry	*pyds9_error_popup()*
Open in DS9 menu entries	*launch_ds9()*

Parameters

title [string, optional] name of the menu

new_ds9_name [list of strings, optional] list of string that are concatenated to create a name for a new DS9 window. If nothing is provided, a name will be created

new_ds9_menu [bool, optional] add a menu to create a new DS9 window

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

Attributes

sig_ds9

sig_ds9

populate_ds9_menu()

Fill the DS9 menu of the IFU viewer. If ds9 instances are running, list them underneath the 'Send selected to' tab. Connect these so that on click the selected files are sent to that ds9 instance.

pyds9_error_popup()

Create an error popup with the pyds9 message. This method is also a PyQt slot.

error_popup(*error*)

Create a popup reporting the error.

Parameters

error [list] error[0]: short error message, set as informative text; error[1]: full traceback, set as detailed text

launch_ds9(*action=None*)

Send files to DS9. If action is specified send to the DS9 window associated to it. Otherwise, create a new DS9 window with an automatically generated name.

Parameters

action [`PyQt5.QtWidgets.QAction`, optional] action clicked; if provided its text is used as the DS9 instance name

_ds9_name(*action*)

Get the name from the action or create a new name, if action is `None`.

Parameters

action [`PyQt5.QtWidgets.QAction`, optional] action clicked; if provided its text is used as the DS9 instance name

Returns

name [string] name of the ds9 windows

class vdat.gui.tabs.ifu_viewer.**FitsViewerTab**(*fname*, *tab_dict*, *parent=None*)

Bases: `PyQt5.QtWidgets.QWidget`

Widget that display a fits file using Ginga in the upper part and the file header on the lower part.

Parameters

fname [string] name of the file to show

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

Attributes

fname [string] name of the file displayed in the tab

class vdat.gui.tabs.ifu_viewer.**FitsViewerWindow**(*file_names*, *tab_dict*, *new_ds9_name=None*, *titles=None*, *tooltips=None*, *parent=None*)

Bases: `PyQt5.QtWidgets.QMainWindow`

Display the given files using a ginga based viewer. Show the header information and allow to open files in DS9.

Table 63: **Custom slot**

Name	Signature	Description
send_to_ds9()	`pyds9.DS9`	Push the file to the ds9 window passed as argument

Table 64: **Connections between custom signals and/or slots**.

Signal	Slot
DS9Menu.sig_ds9	*send_to_ds9()*

Parameters

file_names [list] list of files to display

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

new_ds9_name [list of strings, optional] list of string that are concatenated to create a name for a new DS9 window. If nothing is provided, a name will be created

titles, tooltips [list of strings, optional] if given they are used as title page and tooltip, respectively, for each of the tabs shown. If `None` the file names are used. Otherwise they must be a list with the same length of `file_names`

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

Attributes

ds9_menu [`PyQt5.QtWidgets.QMenu`] menu containing the ds9 buttons

tab_dict [dictionary] same as input

current_tab Tab currently shown

current_tab
Tab currently shown

make_menubar()
Create a menu bar with a ds9 button.

show_ginga_help()
Open the ginga url into a browser.

Ths method is also a PyQt slot

send_to_ds9(*ds9*)
Send files to DS9. If action is specified send to the DS9 window associated to it. Otherwise, create a new DS9 window with an automatically generated name.

Parameters

ds9 [`pyds9.DS9`] ds9 instance to where the data are sent

make_main_widget(*file_names*, *tab_dict*, *titles*, *tooltips*)
Create a tab widget and add *`FitsViewerTab`* classes as tabs.

Parameters

file_names [list of strings] files to display

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

titles, tooltips [list of strings or None] if None, use the file names, otherwise check the length is the same of `file_names`

Returns

tab_widget [`PyQt5.QtWidgets.QTabWidget`] widget to plug as main in the window

_make_title_tooltip(*file_names*, *titles*, *tooltips*)
Prepare the titles and tooltips for the tabs

Parameters

file_names [list of strings] files to display

titles, tooltips [list of strings or None] if None, use the file names, otherwise check the length is the same of `file_names`

Returns

file_names, titles, tooltips [list of strings] if titles and tooltips are `None` return lists with default values

Raises

FitsViewerTitleTooltipError when titles or tooltips is malformed

class vdat.gui.tabs.ifu_viewer.**GingaWidget**(*fname*, *tab_dict*, *parent=None*)

Bases: PyQt5.QtWidgets.QWidget

Widget that display a fits file using Ginga.

Parameters

fname [string] name of the file to show

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

ginga_panel(*fn*)

Setups up a Ginga fits viewer panel.

Parameters

fn [string] the filename of the fits file to view

Returns

fitsview [ImageViewCanvasQt]

showEvent(*event*)

When the tab becomes visible, this method is triggered by Qt.

If it is the first time it is shown, zoom the image to fit.

Parameters

event [PyQt5.QtGui.QShowEvent] the tab is show

load_fits_image(*fn*, *log*)

Load a FITS file as a Ginga AstroImage and return it

Parameters

fn [String] the filename of the image to load

log : a logger

Returns

image [AstroImage instance]

class vdat.gui.tabs.ifu_viewer.**HeaderWidget**(*fname*, *tab_dict*, *parent=None*)

Bases: PyQt5.QtWidgets.QTextEdit

Widget that display a fits file headers.

Parameters

fname [string] name of the file to show

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

parse_fits_header(*fn*, *tab_dict*)

Convert a FITS header to a string and return it.

The method uses directly the following keys of the tab_dict argument:

- header_keys (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Parameters

fn [string] the filename of the fits file

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

Returns

ostr [string] The header information

_order_header_keys(*istr*, *header_keys*)

Split the incoming string on new lines and move to the beginning all the lines starting with a keyword in header_keys.

Parameters

istr [string] string with the full headers

header_keys [list of strings] header keywords to put at the beginning

Returns

ostr [string] string reorganized

class vdat.gui.tabs.ifu_viewer.**FileEditorWindow**(*file_name*, *tab_dict*, *parent=None*)

Bases: PyQt5.QtWidgets.QMainWindow

Simple single file editor window.

This is loosely based on https://www.binpress.com/tutorial/building-a-text-editor-with-pyqt-part-one/143

Table 65: **Custom signals**

Name	Signature	Description
sig_saved	str	emitted when a file is emitted

Table 66: **Custom slot**

Name	Signature	Description
set_title()	str	set the window title with the given file name
mark_not_modified()	str, optional	mark the window as not modified, input parameters ignored
save_file()	str, optional	save the document
save_file_as()		save the document as a new file

Table 67: **Connections between custom signals and/or slots**.

Signal	Slot
vdat.gui.menus_actions.QuitAction.triggered	close()
save_action.triggered	*save_file()*
save_as_action.triggered	*save_file_as()*
sig_saved	*set_title()*
sig_saved	*mark_not_modified()*

Parameters

file_names [string] file to display

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

Attributes

file_name [string] name of the file to display

text_edit [`PyQt5.QtWidgets.QTextEdit`] display the text

save_action [`PyQt5.QtWidgets.QAction`] action to save the document

save_as_action [`PyQt5.QtWidgets.QAction`] action to save the document under a new name

sig_saved

make_text_edit()
Create a QTextEdit edit and returns it

make_common_actions()
Create a number of actions than are plugged into the task and menu bars

make_menubar()
Create and return the menu

make_toolbar()
Create and return the toolbar

set_title(*file_name*)
Set the title of the window.

This method is also a PyQt slot.

Parameters

file_name [string] name of the file

mark_not_modified(_=*None*)
Mark the window as not modified. The parameter is ignored

This method is also a PyQt slot.

save_file(*fname=None*)
Get the text and save it to file. If `fname` is `None`, overwrite the input file.

This method is also a PyQt slot

Parameters

fname [string, optional] name of the file to use

save_file_as()
Ask for a new file name and save there the file. If no file name is given, nothing happens.

This method is also a PyQt slot

Parameters

fname [string, optional] name of the file to use

closeEvent(*event*)
If the file is modified, ask what to do before closing. If `Cancel` is pressed, the window is not closed.

Note: from the documentation it seems that it's preferable to reimplement *`closeEvent()`* and the `close()` method.

Parameters

event [`PyQt5.QtGui.QCloseEvent`] event emitted when a close attempt is made

class vdat.gui.tabs.ifu_viewer.**DistTab**(*dist_name*, *reg_name*, *fits_names*, *tab_dict*, *parent=None*)

Bases: PyQt5.QtWidgets.QWidget

Widget that display a distortion solution and the region

Parameters

dist_name [string] name of the distortion file to show

reg_name [string] name of the region file attached to dist_name

fits_names [list of strings] name of the fits file that can be sent to DS9

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

parent [PyQt5.QtWidgets.QWidget or derivate] parent object of the tree view model

Attributes

dist_name [string] same as input

reg_name [string] same as input

fits_names [list of strings] same as input

dist_text_widget()

Create a read-only text area displaying the distortion file

Returns

text_display [PyQt5.QtWidgets.QTextEdit] widget showing the dist file

class vdat.gui.tabs.ifu_viewer.**DistWindow**(*dist_files*, *reg_files*, *fits_files*, *tab_dict*, *new_ds9_name=None*, *titles=None*, *tooltips=None*, *parent=None*)

Bases: PyQt5.QtWidgets.QMainWindow

Show the distortion file and the region file and add button to send the distortion region to DS9.

Table 68: **Custom slot**

Name	Signature	Description
send_dist_fits_to_ds9()	(pyds9.DS9	Push the fits files and the regions to the ds9 window passed as argument
send_dist_to_ds9()	pyds9.DS9	Push the regions to the ds9 window passed as argument

Table 69: **Connections between custom signals and/or slots**.

Signal	Slot
DS9Menu.sig_ds9 (menu 'Distortion and fits')	*send_dist_fits_to_ds9()*
DS9Menu.sig_ds9 (menu 'Distortion only')	*send_dist_to_ds9()*

Parameters

dist_files [list of strings] distortion files to display

reg_files [list of strings] region files to display, must have the same size of dist_files

fits_names [2d list of strings] list of list of fits files, the first dimension must have the same size of dist_files

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

new_ds9_name [list of strings, optional] list of string that are concatenated to create a name for a new DS9 window. If nothing is provided, a name will be created

titles, tooltips [list of strings, optional] if given they are used as title page and tooltip, respectively, for each of the tabs shown. If `None` the file names are used. Otherwise they must be a list with the same length of `dist_files`

parent [`PyQt5.QtWidgets.QWidget` or derivate] parent object of the tree view model

Attributes

ds9_menu [`PyQt5.QtWidgets.QMenu`] menu containing the ds9 buttons

current_tab Tab currently shown

current_tab
Tab currently shown

make_menubar()
Create a menu bar with a ds9 button.

make_main_widget(*dist_files*, *reg_files*, *fits_files*, *tab_dict*, *titles*, *tooltips*)
Create a tab widget and add *`DistTab`* classes as tabs.

Parameters

dist_files [list of strings] distortion files to display

reg_files [list of strings] region files to display, must have the same size of `dist_files`

fits_names [2d list of strings] list of list of fits files, the first dimension must have the same size of `dist_files`

tab_dict [dictionary] dictionary with the specifications to use to build the tabs

titles, tooltips [list of strings, optional] if given they are used as title page and tooltip, respectively, for each of the tabs shown. If `None` the file names are used. Otherwise they must be a list with the same length of `dist_files`

Returns

tab_widget [`PyQt5.QtWidgets.QTabWidget`] widget to plug as main in the window

_make_title_tooltip(*file_names*, *titles*, *tooltips*)
Prepare the titles and tooltips for the tabs

Parameters

file_names [list of strings] files to display

titles, tooltips [list of strings or None] if None, use the file names, otherwise check the length is the same of `file_names`

Returns

file_names, titles, tooltips [list of strings] if titles and tooltips are `None` return lists with default values

Raises

DistViewerFileNumberError when titles or tooltips is malformed

send_dist_fits_to_ds9(*ds9*)
Send the fits files to DS9 and overplot the region files.

This method is also a PyQt slot.

Parameters

ds9 [`pyds9.DS9`] ds9 instance to where the data are sent

send_dist_to_ds9(*ds9*)

Send the region file to a DS9 window.

This method is also a PyQt slot.

Parameters

ds9 [`pyds9.DS9`] ds9 instance to where the data are sent

send_region_to_ds9(*ds9*, *regions_file*, *chunk_size=50*)

Send the regions from the region file to DS9.

This method assumes that each line is a region command. Because of this it cannot simply load the file into DS9, but have to send the regions in chunks of size `chunk_size`.

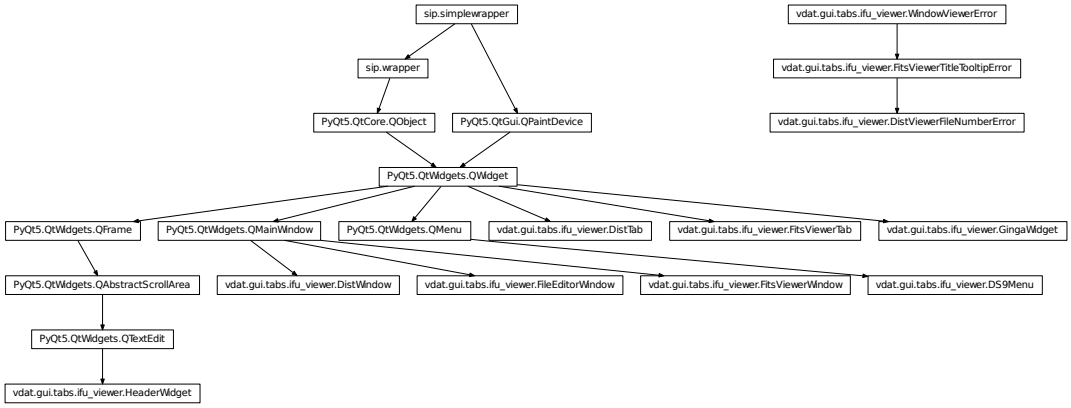
Parameters

ds9 [`pyds9.DS9`] ds9 instance to where the data are sent

regions_file [string] name of the file containing the regions

chunk_size [int, optional] number of region elements to send in one go to DS9

Inheritance diagram



`vdat.gui.tabs.entry_points` - Tab types definitions

This module implements all the entry points shipped with vdat

`vdat.gui.tabs.entry_points.`**exp_fits**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return tabs of type `tab_widget.FitsFplanePanel`. Each tab represent one exposure in one of the types in `target_dir`.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type} {exp}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.
- `header_keys` (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `basename`: date-time part of the file name.
- `col`, `row` (`file_name` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type of the file(s) in the target directory.
- `object`: value of the OBJECT header keyword.
- `exp`: exposure number.

See *`interface.plugin_interface()`* for the signature of this function.

`vdat.gui.tabs.entry_points.`**`exp_combined`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as *`exp_fits()`*, but using tabs of type *`tab_widget.FitsAndReconFplanePanel`*.

`vdat.gui.tabs.entry_points.`**`_exp_tabs`**(*cls*, *target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Implementation of *`exp_fits()`* and *`fits_combined()`* types.

Parameters

cls [*`tab_widget.BaseFplanePanel`* or child] class to initialize if not found in cache

others : same as *`fits_combined()`*

`vdat.gui.tabs.entry_points.`**`fits`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one or more tabs of type *`tab_widget.FitsFplanePanel`*. Each tab represents one file type chosen by the user.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `file_types` (optional): can be `'type'`, `'orig_type'`, `'object'` or a user defined string. This entry is used in the following way:

  - If the value of `file_types` is any of `'type'`, `'orig_type'`, `'object'`: the corresponding information is extracted from the internal database and interpreted as a list. If e.g. the `'orig_type'` is used and the database contains for the entry the following `'cmp, flt'` value, two tabs will be created and the `{orig_type}` formatting key will be replaced with `'cmp'` in one tab and `'flt'` in the other.
  - If the value of `file_types` is any of `'type'`, `'orig_type'`, `'object'` **and** if it present as a keyword in the tab configuration: the content of the keyword is used, instead of the internal information. The value of the keyword must be a list. If e.g. the `'orig_type'` is used and there is one keyword whose value is `['flat', 'arc']`, two tabs will be created and the `{orig_type}` formatting key will be replaced with `'flat'` in one tab and `'arc'` in the other.
  - If the value is a custom string: this string must be present in the tab configuration and its value must be a list. It is also added in the list of available formatting names to allow substitutions. If e.g. the `'custom'` string is used, there must be one such key. If its value is `['my', 'tab']`, two tabs will be created and the `{custom}` formatting key will be replaced with `'my'` in one tab and `'tab'` in the other.

  Default: `'orig_type'`.

- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.
- `header_keys` (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `col`, `row` (`file_name` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type of the file(s) in the target directory.
- `object`: value of the OBJECT header keyword.

See *`interface.plugin_interface()`* for the signature of this function.

`vdat.gui.tabs.entry_points.`**`fits_combined`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as *`fits()`*, but using tabs of type *`tab_widget.FitsAndReconFplanePanel`*.

`vdat.gui.tabs.entry_points.`**`fits_cube`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as *`fits()`*, but using tabs of type *`tab_widget.CubeFplanePanel`*.

On top of the configuration options described in *`fits()`*, this tab type accepts the following options:

- `z_indx` (optional): before creating the thumbnail for the data cubes, the image is compressed along the z-dimension using the median; if `z_indx` is not given or is `[null, null]`, it uses the whole range, otherwise it uses only the part of the cube in the range [z_indx[0], z_indx[1])

`vdat.gui.tabs.entry_points.`**`fits_multiext`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Same as `fits()`, but using tabs of type `tab_widget.MultiExtFplanePanel`.

On top of the configuration options described in `fits()`, this tab type accepts the following options:

- `extensions` (list of ints or strings): indices or names of the fits extensions to display in the IFU.

On top of the formatting names described in `fits()`, this tab type has this additional formatting name:

- `ext`: index or name of the extension displayed

`vdat.gui.tabs.entry_points.`**`_fits_tabs`**(*cls*, *target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*, *extra_format=None*)

Implementation of `fits()` and `fits_combined()`.

Parameters

cls [`tab_widget.BaseFplanePanel` or child] class to initialize if not found in cache

extra_format [dict, optional] extra formatting values to add to the `format_dict` passed to the `cls`

others : same as `fits_combined()`

`vdat.gui.tabs.entry_points.`**`reconstruct`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return a tab of type `tab_widget.QuickReconFplanePanel`. It collects all the exposures for the `target_dir` and combine all of them in a single reconstructed image.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.
- `header_keys` (optional): list of strings. Header keywords to show on top of the others in the fits viewer window.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `col`, `row` (`file_name` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `basename` (`file_name` only): date-time part of the file name.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type(s) of the file(s) in the target directory.
- `object`: value(s) of the OBJECT header keyword.

See `interface.plugin_interface()` for the signature of this function.

`vdat.gui.tabs.entry_points.text_file`(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one tab of type `tab_widget.TextFileWidget`.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the file(s) to show. It is possible to format the file name using the python formatting syntax.
- `title` (optional): title to use for the tab; if absent `'{step}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.
- `orig_type`: original type(s) of the file(s) in the target directory.
- `object`: value(s) of the OBJECT header keyword.

See `interface.plugin_interface()` for the signature of this function.

`vdat.gui.tabs.entry_points.dist`(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one tabs of type `tab_widget.DistPanel`.

This tab type accepts the following configuration options:

- `tab_type` (mandatory): name of the type.
- `file_name` (mandatory): name of the distortion file(s) to show. It is possible to format the file name using the python formatting syntax.
- `fits_names` (mandatory): list of names of the fits files to use then displaying the distortion in DS9. If the list is empty, it is not possible to display the data on DS9.
- `cols`, `rows` (optional): list of objects, typically strings. The thumbnail gets divided into len(cols)*len(rows) quadrants and each one shows one file.
- `title` (optional): title to use for the tab; if absent `'{step} {orig_type}'` is used. It is possible to format the title similarly to the `file_name`.
- `tool_tip` (optional): tooltip to show when hovering on the tab name; it is possible to format the tool_tip similarly to the `file_name`.

Available formatting names:

- `ifuslot`, `ifuid`, `specid` (`file_name` and `fits_names` only): ID of the slot, of the IFU bundle and of the spectrograph it is connected to.
- `col`, `row` (`file_name` and `fits_names` only): replaced with each of the elements in the `cols` and `rows` configuration options.
- `step`: name of the step at hand
- `type`: type of the file(s) in the target directory, i.e. the name shown in the GUI.

- `orig_type`: original type of the file(s) in the target directory.
- `object`: value of the OBJECT header keyword.

See *interface.plugin_interface()* for the signature of this function.

`vdat.gui.tabs.entry_points.`**`_empty_fplane`**(*target_dir*, *tab_dict*, *step_name*, *cache*, *parent_widget*)

Create or retrieve and return one tab of type *tab_widget.BaseFplanePanel*.

This entry point has been created for testing purposes but can be used to display the focal plane and be able to select/deselect IFUs when no plugin is available to display data for a step

See *interface.plugin_interface()* for the signature of this function.

New tab types

This document provides guidance and examples on how to create new tab types and reuse some of the functionalities implemented in VDAT.

Entry point and packaging

Each and every tab type, included the ones shipped with VDAT, is a plugin that is loaded and executed when needed. This system offers various advantages, among which:

- new tab types from third party packages;
- clear decoupling between the container (the GUI) and the content (the tabs).

All the tab types are advertised and loaded using the entry point mechanism. VDAT expect them to be advertised in the `vdat.tab_types` entry point group.

An example is probably the best way to illustrate how to use this to feature to extend VDAT. Lets assume that a user needs a new tab type, that we will call `cool_tab`. The best way to implement it is to create a new package, e.g. `vdat_ext` with the following structure:

```
vdat_ext
  ├─ setup.py
  ├─ vdat_ext
  │    ├─ __init__.py
  │    └─ core.py
  └─ tests/
```

The `setup.py` file is need to tell tools like pip how the package is to be installed, which dependences it needs and so on. The minimal setup instruction looks like:

```
from setuptools import setup, find_packages

setup(
    name='vdat_ext',
    packages=find_packages(),
    entry_points={'vdat.tab_types':
                  ['cool_tab = vdat_ext.core:cool_tab_function'],
                 }
)
```

The tab type is implemented in `vdat_ext/core.py` following the template provided in *vdat.gui.tabs.interface*

in a function called `cool_tab_function`:

```
from vdat.gui.tabs.interface import FplaneTabTemplate

def cool_tab_function(target_dir, tab_dict, step_name, cache,
                      parent_widget):
    '''This function implements the ``cool_tab`` plugin. It receive a
    number of input parameters, create and/or setup the tab(s) and return
    it(them).

    If the tab type provides multiple tabs, create all of them in this
    function appropriately and then return a list containing all of them in
    the order they should appear'''
    tab_type = tab_dict['tab_type']
    # if using the cache, try to get one object from the cache. If
    # use_cache == False there is no need to do this
    obj = cache.from_cache(tab_type)
    if not obj:
        obj = CoolTab(tab_type, parent=parent_widget)
    # prepare the arguments and keyword arguments
    obj.setup(arguments, kw_arguments)

    return [obj, ]

class CoolTab(FplaneTabTemplate):
    '''Class implementing the tab(s) for the ``cool_tab`` tab_type'''
    def __init__(self, tab_type, parent=None):
        super(CoolTab, self).__init__(tab_type, parent=parent)
        self.use_cache = True  # cache the tabs when removing them
        # self.use_cache = False  # do not cache the tabs
        self.title = 'A nice name'  # name to associate with the tab
        # any extra code to run at instantiation time

    def setup(self, arguments, keyword_arguments):
        '''This method use the input arguments to setup a tab that can be
        used. When using the cache system, this method can be used to
        re-set the content of the tab'''
        # implement as needed

    # implement all the needed method/slots/etc
```

Once the package is implemented (and of course tested), it can be be installed with `pip install`. This way the plugin becomes available to VDAT and it's possible to use it in the same way of the built-in tab types:

```
tabs:
    - tab_type: cool_tab
      [all the options for the cool_tab type]
```

The plugin anatomy

In this section we illustrate the plugin mechanism and the interface of the tabs.

- When selecting a directory or reduction step, the old tabs are removed from the GUI (see below for more details).
- The `tabs` section for the given directory type and step is retrieved from the configuration file and the code iterates through the items.

- For each item, get the `tab_type` and load the entry point(s) belonging to the `vdat.tab_types` group and with the name given in the aforementioned entry. Ideally there should be only one entry point per name, but identical names provided by different packages are not forbidden. In the future we will provide some tool to check existing plugin names (see issue #1613).
- For each entry point, load it. It should provide a function with a signature compatible with `vdat.gui.tabs.interface.plugin_interface()`. The input, output and what is should do are:
  - Input: `target_dir`, i.e. the directory selected by the user. `tab_dict`: the dictionary with the instructions to build the tab(s); except for the `tab_type` entry, all the remaining entries are specific for the plugin. `step_name`: name of the selected step. `cache`: a `FplaneCache` instance; see *The tabs cache* for more info. `parent_widget` the widget to which the tabs belong.
  - Implementation: each function can create zero or more widgets, as necessary. Here are the main steps and some hints for the implementation.
    * Retrieve from the cache, using the `cache.from_cache` method, or create a widget. We strongly suggest deriving the widget from `FplaneTabTemplate` or a child thereof[1]. See the `FplaneTabTemplate` documentation about the widget interface.
    * Setup the widget for use.
    * If creating multiple widgets, store or yield them.
    * `target_dir` is the absolute path to the selected directory. Information about the directory content can be obtained from the `VDATDir` and `VDATExposures` SQL tables. For info about querying the tables see the peewee documentation or the examples in the code.
  - Output: a list containing the widget(s) to add to the GUI as tabs
- For each widget in the output list:
  - The following signals <-> slots are connected

Table 70: Connections between custom signals and/or slots

Signal	Slot	Explanation
`sig_ifuToggle()`	`ifuToggle()`	The signal should be emitted when a user select or deselect and IFU from the widget. VDAT reacts storing the selected IFUs. They are then used when running commands or switching step/directory.
`sig_ifuSelected()`	`ifuSelected()`	The signal is emitted multiple times from VDAT when switching steps/directory. Tabs can reimplement the slot to react to this, e.g. pre-selecting IFUs.
`sig_selectAllIFUs()`	`selectAllIFUs()`	The signal is emitted when all IFUs are selected at once, e.g. from a menu entry. Tabs should reimplement the slot to react to this, e.g. to show that all IFUs has being selected
`sig_deselectAllIFUs()`	`deselectAllIFUs()`	The signal is emitted when all IFUs are deselected at once, e.g. from a menu entry. Tabs should reimplement the slot to react to this, e.g. to show that all IFUs has being deselected

  - Add the widget to the GUI as a tab. The title for the tab is taken from the widget attribute `title`, that must be present.
  - If the widget has a `tool_tip` attribute, its value is used as tooltip when hovering over the tab name. This could be used to use more information or complete tab names in case of truncations.

[1] `FplaneTabTemplate` implements the the minimal interface needed by the plugin system. It provides the signals and do-nothing implementations of the slots that are connected with the rest of the GUI. Also will make easier to keep plugins up to date in future backward-compatible API changes.

- If error happens between loading the plugin and adding the tab widgets to the GUI, it is reported to the user via error boxes. If no tab can be shown, because of missing configurations or error, an overlay with the motivation will be shown.
- Going back to the beginning of the list: when a new directory or step is selected the tabs shown in the GUI are removed. For each widget:
  - all the signals shown in the previous table are disconnected;
  - call the *cleanup* method; the default implementation hides the widget; override it for more complex behaviour;
  - if the `use_cache` attribute of the widget is set to `True`, it is cached under the tab type name, that is saved in the `tab_type` attribute; widget must implement the `use_cache` attribute;
  - if the `use_cache` attribute of the widget is set to `False`, mark the widget for deletion.

The tabs cache

When an object representing a tab is popped, the code checks if the `use_cache` attribute set to `True`. In this case the tab is stored into a *FplaneCache* instance under the `tab_type` name.

The same instance is passed to every tab type function. This way, the function can use the *from_cache()* to retrieve an existing widget, if available in the cache, instead of creating a new one. If the cache for the given `tab_type` is empty, a `None` is returned and the function should take care of creating a new instance. If an object is returned, it is responsibility of the plugin modify its content according to the context.

Note: Currently the cache doesn't have a size limits, so there could be memory leaks due to object put there but never reused. In general, we suggest to try to retrieve from the cache as in the example above, independently of the value of `use_cache`: this way if the cache needs to enabled or disable for some reasong, the only change to do is the value of the attribute.

The cache has been introduced to avoid memory leaks due to objects not correctly released when deleting them. If a tab types leads to memory leaks, using the cache might help.

Reuse the base classes

Since most of the tabs provided by VDAT share the same basic layout and functionality, we have created some base class implementing this. Here we show how it is possible to extend those classes to create new tab widgets. The classes are *ifu_widget.BaseIFUWidget* and *tab_widget.BaseFplanePanel*.

These two classes, together with the *tab_widget.UpdateIFUTask* thread, are designed together to provide a working, although not very useful, widget.

Here we will sketch some idea on how to extend the base classes and the guide line that should be followed.

Names

Let's start with defining the new classes and entry point. We will provide the implementation extra pieces as we proceed.

```
class NewIFUWidget(BaseIFUWidget):
    pass

class NewFplanePanel(BaseFplanePanel):
    @property
    def ifu_widget_class:
        'we want to use the ``NewIFUWidget`` class'
        return NewIFUWidget

def new_tab(target_dir, tab_dict, step_name, cache, parent_widget):
    pass
```

Setup the classes

The base classes are very basic. Typically we want to pass to the fplane and the ifu widget some more information about what needs to be displayed. Let's say that the `target_dir` and the `tab_dict` contain all the needed information. Some of those information need then to be pushed into the IFU widget. So we can do something like the following:

```
class NewIFUWidget(BaseIFUWidget):
    def setup(self, target_dir, file_template, cols, rows):
        '''Save the template of the file to save and the list of columns
        and rows. These values are used to create a ``len(cols) *
        len(rows)`` images that will be stitched and shown in the IFU.'''
        self.target_dir = target_dir
        self.file_template = file_template
        self.cols = cols
        self.rows = rows
        ## prepare all the sizes needed to deal with len(cols) and len(rows)
        ## images

class NewFplanePanel(BaseFplanePanel):
    def setup(self, target_dir, tab_dict):
        '''Get the selected directory and the configuration dictionary,
        manipulate the relevant entries and pass the to all the IFU widget.
        '''
        # from the target_dir extract the type of the file shown using
        # the database.
        file_type = query_database(target_dir)

        # set the title and the tooltip for the current tab
        self.title = tab_dict['title'].format(ftype=file_type)
        self.tool_tip = self.title
        file_template = tab_dict['fname'].format(ftype=file_type)

        for ifu in self.fplane.ifus:
            ## replace the {ftype} and the {ifuslot} in the file template
            ifu.setup(target_dir, file_template, tab_dict.get('cols', ['*', ]),
                      tab_dict.get('rows', ['*', ]))
```

With this we can now rewrite the entry point function:

```
def new_tab(target_dir, tab_dict, cache, parent_widget):
    '''Implement the ``new_tab`` tab type. It can be configured via the
    following entries in the configuration file:
```

(continues on next page)

(continued from previous page)

```
    * fname (mandatory): template used to construct the name of the file(s)
      to show. Can expand the following placeholders:

      * {ftype}: type of the files in the directory
      * {ifuslot}: slot ID
      * {col}: replaced with the elements from ``cols``
      * {row}: replaced with the elements from ``rows``

      Once all the placeholders has been filled, the resulting name is
      joined with the ``target_dir`` and then the first file matching it is
      take. The matching is done via :func:`glob.glob`.
    * title (mandatory): title of the tab. Can expand the {ftype}
      placeholder
    * cols, rows (optional): when creating the image to show in the IFU,
      loop through the cols and rows, for each one replace the {col} and
      {row} placeholder in the file template and place the corresponding
      file in the correct position in the IFU. They both default to
      ``['*']``
    '''
    tab_type = tab_dict['tab_type']
    # try to get one object from the cache.
    tab_obj = cache.from_cache(tab_type)
    if not tab_obj:
        tab_obj = NewFplanePanel(tab_type)
    # reset the parent. It is not strictly necessary but, hey, better safe
    # that sorrow
    tab_obj.setParent(parent_widget)
    # pass all the relevant pieces to the object to be able to show the target
    # directory
    tab_obj.setup(target_dir, tab_dict)

    return [tab_obj, ]
```

A note on showing the tab

The tabs defined by *tab_widget.BaseFplanePanel* and derivatives are drawn only when showing them. We do this because some of the tabs require time and/or CPU consuming tasks to create what need to be shown in the IFUs, so it is possible to show many tabs and then fill only the ones that we actually want to see.

New one widget is shown, Qt checks that it implements the `showEvent()` and calls it. So we implement *tab_widget.BaseFplanePanel.showEvent()* and, if the `tab_widget.BaseFplanePanel.initialized` is set to False, build the table content and set it to True. Successive show events will do nothing. When the tab is removed and *tab_widget.BaseFplanePanel.cleanup()* is called, the tab widget is reset to the non initialized status. If derived classes reimplement this method, they should take care of calling the parent class one and also to make sure that the IFU widgets `cleanup` method clears the the images shown.

Create the images to show

Now we need to implement the functionalities to actually create and show the images in the IFUs. We trigger it with the *tab_widget.BaseFplanePanel.update_ifus()* method. We don't have to override it, however to correctly build the IFUs we need to reimplement the *ifu_widget.BaseIFUWidget.prepare_image()* method to build the image to display.

```
class NewIFUWidget(BaseIFUWidget):
    [...]
    def prepare_image(self):
        '''Prepare the image to be shown in the GUI'''
        # first call the parent method to be sure that the IFU shows
        # something
        super(NewIFUWidget, self).prepare_image()
        image = QtGui.QImage(self.size, self.size,
                             QtGui.QImage.Format_RGB16)
        image.fill(QtCore.Qt.black)
        p = QtGui.QPainter()
        p.begin(img)
        has_file = False
        for (i, col), (j, row) in it.product(enumerate(cols),
                                             enumerate(rows)):
            fname = self.file_template(ifuslot=self.ifuslot,
                                       col=col, row=row)
            fname = os.path.join(self.target_dir, fname)
            try:
                fname = glob.glob(fname)[0]  # get the first file name
                has_file = True
            except IndexError:  # no matching file found
                continue
            # if the files are big you might want to make thumbnails
            # and show them. We are not going to show how to implement such
            # a function here
            thumbnail = self.get_thumbnail(fname)
            ## add the thumbnail to the image in position (i, j)
        p.end()
        # if at least one file was found, it is worth showing it
        # otherwise the default empty image will be created.
        if has_file:
            self.current_img = image
```

After the `prepare_image` method returns, the showing is triggered automatically.

Open a window on double click

The *`ifu_widget.BaseIFUWidget`* has a handler for dealing with double clicks, but it doesn't do anything beside making sure that a double click won't trigger two selections of the IFU. However you might want to use it to do more, like opening a window to better inspect the data.

```
class NewIFUWidget(BaseIFUWidget):
    [...]
    def mouseDoubleClickEvent(self, event):
        'open a window on double click'
        # make sure to call the parent class implementation to get the
        # single/double click distinction
        super(NewIFUWidget, self).mouseDoubleClickEvent(event)

        # prepare a window, pass it the file names to show and whatever
        # else is necessary.
        window = NewWindow(...)
        window.show()
```

If the window need some configuration option, pass the configuration dictionary to the `NewIFUWidget.setup()`

method and store it.

Extras

Of course you might want to have more complex behaviours. To have ideas and examples give a look at the current implementation. We might extend this part in the future.

Have fun coding.

Document the GUI, including signals and slots, as done e.g. in `vdat.gui.queue.Queue`

2.2.5 `vdat.database` – The VDAT database

The database

Database management for VDAT

`vdat.database.core.`**`init`**(*redux_dir=None*, *remove_old=True*)

Initialise the database

Parameters

redux_dir [string, optional] if given, create a physical database, otherwise creates it in memory

remove_old [bool, optional] drop existing tables if true

Fields and Models

Database management for VDAT

exception `vdat.database.models.`**`MergeTypeError`**

Bases: `vdat.database.base.VDATDBError`, `TypeError`

Raised when merging table entries of different types

class `vdat.database.models.`**`BoolField`**(*null=False*, *index=False*, *unique=False*, *column_name=None*, *default=None*, *primary_key=False*, *constraints=None*, *sequence=None*, *collation=None*, *unindexed=False*, *choices=None*, *help_text=None*, *verbose_name=None*, *db_column=None*, *_hidden=False*)

Bases: `peewee.Field`

sqlite doesn't play well with boolean, apparently. This field coerce `True` and `False` into strings `"true"` and `"false"` and vice versa

`db_field = 'text'`

`db_value`(*value*)

Convert the input bool into a string

`python_value`(*value*)

Convert the input string into a bool

class `vdat.database.models.`**`VDATDir`**(**args*, ***kwargs*)

Bases: `vdat.database.base.VDATModels`

Redux directories metadata

`version = '1.0'`

version_f = <TextField: VDATDir.version_f>

redux_dir = <TextField: VDATDir.redux_dir>

night = <TextField: VDATDir.night>

type_ = <TextField: VDATDir.type_>

name = <TextField: VDATDir.name>

path = <TextField: VDATDir.path>

is_clone = <BoolField: VDATDir.is_clone>

shot = <TextField: VDATDir.shot>

timestamp = <DateTimeField: VDATDir.timestamp>

original_type_ = <TextField: VDATDir.original_type_>

object_ = <TextField: VDATDir.object_>

zero_dir = <ForeignKeyField: VDATDir.zero_dir>

cal_dir = <ForeignKeyField: VDATDir.cal_dir>

make_path()

Fill the `path` field joining `redux_dir`, `night`, `type_` and `name`

merge_entries(*others, exclude=['id', 'zero_dir', 'cal_dir', 'ifus']*)

Merge the self with `other` excluding `exclude`.

The merging is done using *`vdat.utilities.merge_dicts()`*

Parameters

other [(list of) *`VDATDir`* instance] list of models to merge with

exclude [list of strings] list of field names to ignore when merging

Raises

MergeTypeError if the types of `others` are not the same as the owner class

data_clean

Expose the internal data of the model without the id and without recursing the foreign keywords

Returns

dict dictionary of data

DoesNotExist

alias of `VDATDirDoesNotExist`

_meta = <peewee.Metadata object>

_schema = <peewee.SchemaManager object>

cal_dir_id = <ForeignKeyField: VDATDir.cal_dir>

id = <AutoField: VDATDir.id>

target_cal_dir

target_zero_dir

zero_dir_id = <ForeignKeyField: VDATDir.zero_dir>

class vdat.database.models.**VDATExposures**(**args*, ***kwargs*)

Bases: *vdat.database.base.VDATModels*

Redux directories metadata

version = '1.0'

version_f = <TextField: VDATExposures.version_f>

path = <TextField: VDATExposures.path>

name = <TextField: VDATExposures.name>

basename = <TextField: VDATExposures.basename>

expname = <TextField: VDATExposures.expname>

exptype = <TextField: VDATExposures.exptype>

original_type = <TextField: VDATExposures.original_type>

object_ = <TextField: VDATExposures.object_>

DoesNotExist

alias of VDATExposuresDoesNotExist

_meta = <peewee.Metadata object>

_schema = <peewee.SchemaManager object>

id = <AutoField: VDATExposures.id>

The base

Base definitions common to all the other modules in the subpackage

exception vdat.database.base.**VDATDBError**

Bases: Exception

Base exceptions for database errors

class vdat.database.base.**VDATModels**(**args*, ***kwargs*)

Bases: peewee.Model

Base class for linking models with the VDAT database

data

Expose the full internal data of the model

Returns

dict dictionary of data

data_noid

Expose the internal data of the model excluding the id

Returns

dict dictionary of data

data_clean

Expose the internal data of the model without the id and without recursing the foreign keywords

Returns

dict dictionary of data

DoesNotExist
alias of `VDATModelsDoesNotExist`

_meta = <peewee.Metadata object>

_schema = <peewee.SchemaManager object>

id = <AutoField: VDATModels.id>

2.2.6 `vdat.utilities` – Generic utilities

Utilities

`vdat.utilities.`**ISOTIME_FMT = '%Y-%m-%dT%H:%M:%S.%f'**
Date time formatting in the json

`vdat.utilities.`**SHOT_FILE = 'shot_name.txt'**
The file contains basic information about the type of files and the original directory and is used to rebuild the database on subsequent runs of vdat

`vdat.utilities.`**EXPS_FILE = 'exposure_names.txt'**
Maps the base name of each virus fits file (basically the time stamp), with the exposure number

class `vdat.utilities.`**DatetimeEncoder**(**args*, ***kwargs*)
Encodes `datetime.date`, `datetime.time` or `datetime.datetime` as dictionary:

{"__datetime__": True, "date": formatted datetime, "type": datetime type, "fmt": format}

with `datetime type` is one of: "datetime", "date", "time"

Parameters

***args, **kwargs** [same as `json.JSONEncoder`]

dt_formatter [string] formatter used to encode the datetime; defaults to *ISOTIME_FMT*

d_formatter [string] formatter used to encode the date; defaults to the part before "T" in *ISOTIME_FMT*

t_formatter [string] formatter used to encode the time; defaults to the part after "T" in *ISOTIME_FMT*

default(*obj*)
Decode object `obj`. If it's not a date/time/datetime instance delegate to the parent class default

Parameters

obj [object to be serialised]

Returns

serialised object

_datetime_dic(*date*, *type_*, *fmt*)
Create the dictionary to feed to the encoder

`vdat.utilities.`**decode_datetime**(*dct*)
If the input dictionary has a `__datetime__` key set to true, uses the keys "date", "type" and "fmt" to decode the datetime encoded by *DatetimeEncoder*

Parameters

dct [dictionaries] object to decode

Returns

decoded object

`vdat.utilities.`**`json_dumps`**(*obj*)

Serialise `obj` into a json using `DatetimeEncoder` and %Y-%m-%dT%H:%M:%S.%f formatting for the date

Parameters

obj [object to serialise]

Returns

string serialised json

`vdat.utilities.`**`json_loads`**(*s*)

Deserialize the string `s` into a python object, undoing the datetime encoding done by `DatetimeEncoder`

Parameters

s [string to deserialize]

Returns

python object

`vdat.utilities.`**`read_json_file`**(*fname*, *decode=True*)

Read the content of the file and, if `decode` is `True` decode each line as a json entry

Parameters

fname [string] name of the file to read

decode [bool, optional] decode each line in the file as a json

Returns

list of string or of objects lines of the file

`vdat.utilities.`**`write_to_json_file`**(*fname*, *append=True*, ***kwargs*)

Serialize the keyword arguments and write them as a single line to `fname`.

Parameters

fname [string] name of the file where to write

append [bool, optional] if true append to the file, if false write to it

kwargs [dictionary] line to write

`vdat.utilities.`**`_read_file`**(*fname*)

Wrap `read_json_file()` to pass only the directory name

`vdat.utilities.`**`_write_file`**(*fname*)

Wrap `write_to_json_file()` to pass only the directory name

`vdat.utilities.`**`read_shot_file`**(*dir_*, *decode=True*)

Read the content of the shot file 'shot_name.txt' in `dir_`.

Parameters

dir_ [string] name of the directory where the file is located

decode [bool, optional] decode each line in the file as a json

Returns

list of string or of objects lines of the file

`vdat.utilities.read_exps_file(dir_, decode=True)`

Read the content of the shot file 'exposure_names.txt' in `dir_`.

Parameters

dir_ [string] name of the directory where the file is located

decode [bool, optional] decode each line in the file as a json

Returns

list of string or of objects lines of the file

`vdat.utilities.write_to_shot_file(dir_, append=True, **kwargs)`

Serialize the keyword arguments as a single line to the shot_name.txt file in directory `dir_`.

Parameters

dir_ [string] name of the directory where the file is located

append [bool, optional] if true append to the file, if false write to it

kwargs [dictionary] line to write

`vdat.utilities.write_to_exps_file(dir_, append=True, **kwargs)`

Serialize the keyword arguments as a single line to the exposure_names.txt file in directory `dir_`.

Parameters

dir_ [string] name of the directory where the file is located

append [bool, optional] if true append to the file, if false write to it

kwargs [dictionary] line to write

`vdat.utilities.merge_dicts(dicts, exclude=[])`

Merge the dictionaries into one

Unique entries are copied verbatim. For repeated entries: * if string: join them with ", " * if date or datetime: average them * if bool: all is used: so is True only if all the entries are True

Parameters

dicts [list of dictionaries] dictionaries to merge

exclude [list of strings] exclude entries from `out_dict`.

Returns

out_dict [dictionary] merged dictionaries

Raises

VDATUnknownDictEntry if it doesn't know what to do how to merge the entry

`vdat.utilities.collect_metadata(redux_dir, skip_empty=False, repair_redux=False, merge_shot=False, yield_dir=False)`

Recursively scan the `redux_dir` directory looking for shot_name.txt and exposure_names.txt files and yield their content.

Parameters

redux_dir [string] name of the directory to scan

skip_empty [bool, optional] if any of the two files is empty or does not exist, skip the directory

repair_redux [bool, optional] if the `redux_dir` entry in the shot_name.txt files is different from the input one, update it

merge_shot [bool, optional] if the file shot_name.txt contains multiple lines, merge them into one

yield_dir [bool, optional] if True, yields also the name of the directory containing the files

Yields

shot_file, exps_file [list of dict] content of the shot_name.txt and the exposure_names.txt files

`vdat.utilities.grouper`(*iterable*, *n*, *fillvalue=None*)

Collect data into fixed-length chunks or blocks.

From https://docs.python.org/3/library/itertools.html#itertools-recipes

Parameters

iterable : iterable to split in chunks

n [int] size of the chunks

fillvalue [anything, optional] if the size the interable isn't a multiple of `n`, fill the last chunk with `fillvalue`

Returns

iterable chunk of size `n` of the input iterable

Examples

```
>>> list(grouper('ABCDEFG', 3, 'x'))
[('A', 'B', 'C'), ('D', 'E', 'F'), ('G', None, None)]
```

exception `vdat.utilities.VDATError`

Generic vdat error

exception `vdat.utilities.VDATDirError`

Error raised when trying to create directories

exception `vdat.utilities.VDATSymlinkError`

Generic error raised when performing the symlinking

exception `vdat.utilities.VDATFitsParseError`

Exception raised when the parsing of the fits file name or headers to extract information during the symlinking fails

exception `vdat.utilities.VDATFitsTypeError`

Error raised when the type of the fits files is wrong or unknown

exception `vdat.utilities.VDATDateError`

Error raised when failing to parse dates

exception `vdat.utilities.VDATUnknownDictEntry`

Error raised when the shot file is malformed or contains unexpected entries

exception `vdat.utilities.VDATDatabaseError`

Database related errors

exception `vdat.utilities.VDATDatabaseUniquenessError`

The entry in the database is not unique

2.2.7 `vdat.exceptions` – VDAT exceptions

VDAT exceptions

exception `vdat.exceptions.`**`VDATException`**
Base exception for VDAT

exception `vdat.exceptions.`**`VDATValueError`**
`ValueError` for VDAT

2.2.8 `vdat.list_plugins` – List plugins used by VDAT

Entry point of the executable to use to list plugins

`vdat.list_plugins.`**`parse`**(*argv=None*)
Create the parser and parse the command line arguments

Parameters

argv [list] command line, if None taken from `sys.argv`

Returns

Namespace parsed command line

`vdat.list_plugins.`**`main`**(*argv=None*)
Main function of the implementation of the `vdat_plugins` executable

Parameters

argv [list] command line, if None taken from `sys.argv`

`vdat.list_plugins.`**`list_plugins`**(*entry_point_name*, *args*)
List the plugins.

Parameters

entry_point_name [string] name of the entry point to explore

args [Namespace] arguments to make the copy run

CHAPTER 3

About

3.1 LICENSE

3.1.1 Code Licence: GPL v3

```
                       GNU GENERAL PUBLIC LICENSE
                          Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                             Preamble

  The GNU General Public License is a free, copyleft license for
software and other kinds of works.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
the GNU General Public License is intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.  We, the Free Software Foundation, use the
GNU General Public License for most of our software; it applies also to
any other work released this way by its authors.  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.
```

(continues on next page)

(continued from previous page)

```
  To protect your rights, we need to prevent others from denying you
these rights or asking you to surrender the rights.  Therefore, you have
certain responsibilities if you distribute copies of the software, or if
you modify it: responsibilities to respect the freedom of others.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must pass on to the recipients the same
freedoms that you received.  You must make sure that they, too, receive
or can get the source code.  And you must show them these terms so they
know their rights.

  Developers that use the GNU GPL protect your rights with two steps:
(1) assert copyright on the software, and (2) offer you this License
giving you legal permission to copy, distribute and/or modify it.

  For the developers' and authors' protection, the GPL clearly explains
that there is no warranty for this free software.  For both users' and
authors' sake, the GPL requires that modified versions be marked as
changed, so that their problems will not be attributed erroneously to
authors of previous versions.

  Some devices are designed to deny users access to install or run
modified versions of the software inside them, although the manufacturer
can do so.  This is fundamentally incompatible with the aim of
protecting users' freedom to change the software.  The systematic
pattern of such abuse occurs in the area of products for individuals to
use, which is precisely where it is most unacceptable.  Therefore, we
have designed this version of the GPL to prohibit the practice for those
products.  If such problems arise substantially in other domains, we
stand ready to extend this provision to those domains in future versions
of the GPL, as needed to protect the freedom of users.

  Finally, every program is threatened constantly by software patents.
States should not allow patents to restrict development and use of
software on general-purpose computers, but in those that do, we wish to
avoid the special danger that patents applied to a free program could
make it effectively proprietary.  To prevent this, the GPL assures that
patents cannot be used to render the program non-free.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
```

(continues on next page)

(continued from previous page)

```
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.

  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
```

(continues on next page)

(continued from previous page)

```
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.

  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
```

(continues on next page)

(continued from previous page)

```
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.

  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.
```

(continues on next page)

(continued from previous page)

```
    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
```

(continues on next page)

(continued from previous page)

```
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or
```

(continues on next page)

(continued from previous page)

```
    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).

  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.
```

(continues on next page)

(continued from previous page)

```
  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
```

(continues on next page)

(continued from previous page)

```
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.
```

(continues on next page)

(continued from previous page)

```
  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Use with the GNU Affero General Public License.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU Affero General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the special requirements of the GNU Affero General Public License,
section 13, concerning interaction through a network will apply to the
combination as such.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU General Public License from time to time.  Such new versions will
be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
```

(continues on next page)

(continued from previous page)

```
  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.

  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program>  Copyright (C) <year>  <name of author>
```

(continues on next page)

(continued from previous page)

```
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, your program's commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<https://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs.  If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library.  If this is what you want to do, use the GNU Lesser General
Public License instead of this License.  But first, please read
<https://www.gnu.org/licenses/why-not-lgpl.html>.
```

3.1.2 Documentation Licence: FDL v1.3

```
                GNU Free Documentation License
                 Version 1.3, 3 November 2008


 Copyright (C) 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc.
     <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other
functional and useful document "free" in the sense of freedom: to
assure everyone the effective freedom to copy and redistribute it,
with or without modifying it, either commercially or noncommercially.
Secondarily, this License preserves for the author and publisher a way
to get credit for their work, while not being considered responsible
for modifications made by others.

This License is a kind of "copyleft", which means that derivative
works of the document must themselves be free in the same sense.  It
complements the GNU General Public License, which is a copyleft
license designed for free software.

We have designed this License in order to use it for manuals for free
software, because free software needs free documentation: a free
program should come with manuals providing the same freedoms that the
software does.  But this License is not limited to software manuals;
it can be used for any textual work, regardless of subject matter or
whether it is published as a printed book.  We recommend this License
principally for works whose purpose is instruction or reference.
```

(continues on next page)

(continued from previous page)

```
1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that
contains a notice placed by the copyright holder saying it can be
distributed under the terms of this License.  Such a notice grants a
world-wide, royalty-free license, unlimited in duration, to use that
work under the conditions stated herein.  The "Document", below,
refers to any such manual or work.  Any member of the public is a
licensee, and is addressed as "you".  You accept the license if you
copy, modify or distribute the work in a way requiring permission
under copyright law.

A "Modified Version" of the Document means any work containing the
Document or a portion of it, either copied verbatim, or with
modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of
the Document that deals exclusively with the relationship of the
publishers or authors of the Document to the Document's overall
subject (or to related matters) and contains nothing that could fall
directly within that overall subject.  (Thus, if the Document is in
part a textbook of mathematics, a Secondary Section may not explain
any mathematics.)  The relationship could be a matter of historical
connection with the subject or with related matters, or of legal,
commercial, philosophical, ethical or political position regarding
them.

The "Invariant Sections" are certain Secondary Sections whose titles
are designated, as being those of Invariant Sections, in the notice
that says that the Document is released under this License.  If a
section does not fit the above definition of Secondary then it is not
allowed to be designated as Invariant.  The Document may contain zero
Invariant Sections.  If the Document does not identify any Invariant
Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed,
as Front-Cover Texts or Back-Cover Texts, in the notice that says that
the Document is released under this License.  A Front-Cover Text may
be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy,
represented in a format whose specification is available to the
general public, that is suitable for revising the document
straightforwardly with generic text editors or (for images composed of
pixels) generic paint programs or (for drawings) some widely available
drawing editor, and that is suitable for input to text formatters or
for automatic translation to a variety of formats suitable for input
to text formatters.  A copy made in an otherwise Transparent file
format whose markup, or absence of markup, has been arranged to thwart
or discourage subsequent modification by readers is not Transparent.
An image format is not Transparent if used for any substantial amount
of text.  A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain
ASCII without markup, Texinfo input format, LaTeX input format, SGML
or XML using a publicly available DTD, and standard-conforming simple
```

(continues on next page)

(continued from previous page)

```
HTML, PostScript or PDF designed for human modification.  Examples of
transparent image formats include PNG, XCF and JPG.  Opaque formats
include proprietary formats that can be read and edited only by
proprietary word processors, SGML or XML for which the DTD and/or
processing tools are not generally available, and the
machine-generated HTML, PostScript or PDF produced by some word
processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself,
plus such following pages as are needed to hold, legibly, the material
this License requires to appear in the title page.  For works in
formats which do not have any title page as such, "Title Page" means
the text near the most prominent appearance of the work's title,
preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of
the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose
title either is precisely XYZ or contains XYZ in parentheses following
text that translates XYZ in another language.  (Here XYZ stands for a
specific section name mentioned below, such as "Acknowledgements",
"Dedications", "Endorsements", or "History".)  To "Preserve the Title"
of such a section when you modify the Document means that it remains a
section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which
states that this License applies to the Document.  These Warranty
Disclaimers are considered to be included by reference in this
License, but only as regards disclaiming warranties: any other
implication that these Warranty Disclaimers may have is void and has
no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either
commercially or noncommercially, provided that this License, the
copyright notices, and the license notice saying this License applies
to the Document are reproduced in all copies, and that you add no
other conditions whatsoever to those of this License.  You may not use
technical measures to obstruct or control the reading or further
copying of the copies you make or distribute.  However, you may accept
compensation in exchange for copies.  If you distribute a large enough
number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and
you may publicly display copies.


3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have
printed covers) of the Document, numbering more than 100, and the
Document's license notice requires Cover Texts, you must enclose the
copies in covers that carry, clearly and legibly, all these Cover
Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on
the back cover.  Both covers must also clearly and legibly identify
```

(continues on next page)

(continued from previous page)

```
you as the publisher of these copies.  The front cover must present
the full title with all words of the title equally prominent and
visible.  You may add other material on the covers in addition.
Copying with changes limited to the covers, as long as they preserve
the title of the Document and satisfy these conditions, can be treated
as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit
legibly, you should put the first ones listed (as many as fit
reasonably) on the actual cover, and continue the rest onto adjacent
pages.

If you publish or distribute Opaque copies of the Document numbering
more than 100, you must either include a machine-readable Transparent
copy along with each Opaque copy, or state in or with each Opaque copy
a computer-network location from which the general network-using
public has access to download using public-standard network protocols
a complete Transparent copy of the Document, free of added material.
If you use the latter option, you must take reasonably prudent steps,
when you begin distribution of Opaque copies in quantity, to ensure
that this Transparent copy will remain thus accessible at the stated
location until at least one year after the last time you distribute an
Opaque copy (directly or through your agents or retailers) of that
edition to the public.

It is requested, but not required, that you contact the authors of the
Document well before redistributing any large number of copies, to
give them a chance to provide you with an updated version of the
Document.


4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under
the conditions of sections 2 and 3 above, provided that you release
the Modified Version under precisely this License, with the Modified
Version filling the role of the Document, thus licensing distribution
and modification of the Modified Version to whoever possesses a copy
of it.  In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct
   from that of the Document, and from those of previous versions
   (which should, if there were any, be listed in the History section
   of the Document).  You may use the same title as a previous version
   if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities
   responsible for authorship of the modifications in the Modified
   Version, together with at least five of the principal authors of the
   Document (all of its principal authors, if it has fewer than five),
   unless they release you from this requirement.
C. State on the Title page the name of the publisher of the
   Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications
   adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice
   giving the public permission to use the Modified Version under the
```

(continues on next page)

(continued from previous page)

```
   terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections
   and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add
   to it an item stating at least the title, year, new authors, and
   publisher of the Modified Version as given on the Title Page.  If
   there is no section Entitled "History" in the Document, create one
   stating the title, year, authors, and publisher of the Document as
   given on its Title Page, then add an item describing the Modified
   Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on.  These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles.  Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements".  Such a section
   may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or
appendices that qualify as Secondary Sections and contain no material
copied from the Document, you may at your option designate some or all
of these sections as invariant.  To do this, add their titles to the
list of Invariant Sections in the Modified Version's license notice.
These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains
nothing but endorsements of your Modified Version by various
parties--for example, statements of peer review or that the text has
been approved by an organization as the authoritative definition of a
standard.

You may add a passage of up to five words as a Front-Cover Text, and a
passage of up to 25 words as a Back-Cover Text, to the end of the list
of Cover Texts in the Modified Version.  Only one passage of
Front-Cover Text and one of Back-Cover Text may be added by (or
through arrangements made by) any one entity.  If the Document already
includes a cover text for the same cover, previously added by you or
by arrangement made by the same entity you are acting on behalf of,
you may not add another; but you may replace the old one, on explicit
permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License
give permission to use their names for publicity for or to assert or
imply endorsement of any Modified Version.
```

(continues on next page)

(continued from previous page)

```
5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this
License, under the terms defined in section 4 above for modified
versions, provided that you include in the combination all of the
Invariant Sections of all of the original documents, unmodified, and
list them all as Invariant Sections of your combined work in its
license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and
multiple identical Invariant Sections may be replaced with a single
copy.  If there are multiple Invariant Sections with the same name but
different contents, make the title of each such section unique by
adding at the end of it, in parentheses, the name of the original
author or publisher of that section if known, or else a unique number.
Make the same adjustment to the section titles in the list of
Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History"
in the various original documents, forming one section Entitled
"History"; likewise combine any sections Entitled "Acknowledgements",
and any sections Entitled "Dedications".  You must delete all sections
Entitled "Endorsements".


6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other
documents released under this License, and replace the individual
copies of this License in the various documents with a single copy
that is included in the collection, provided that you follow the rules
of this License for verbatim copying of each of the documents in all
other respects.

You may extract a single document from such a collection, and
distribute it individually under this License, provided you insert a
copy of this License into the extracted document, and follow this
License in all other respects regarding verbatim copying of that
document.


7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate
and independent documents or works, in or on a volume of a storage or
distribution medium, is called an "aggregate" if the copyright
resulting from the compilation is not used to limit the legal rights
of the compilation's users beyond what the individual works permit.
When the Document is included in an aggregate, this License does not
apply to the other works in the aggregate which are not themselves
derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these
copies of the Document, then if the Document is less than one half of
the entire aggregate, the Document's Cover Texts may be placed on
```

(continues on next page)

(continued from previous page)

```
covers that bracket the Document within the aggregate, or the
electronic equivalent of covers if the Document is in electronic form.
Otherwise they must appear on printed covers that bracket the whole
aggregate.


8. TRANSLATION

Translation is considered a kind of modification, so you may
distribute translations of the Document under the terms of section 4.
Replacing Invariant Sections with translations requires special
permission from their copyright holders, but you may include
translations of some or all Invariant Sections in addition to the
original versions of these Invariant Sections.  You may include a
translation of this License, and all the license notices in the
Document, and any Warranty Disclaimers, provided that you also include
the original English version of this License and the original versions
of those notices and disclaimers.  In case of a disagreement between
the translation and the original version of this License or a notice
or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements",
"Dedications", or "History", the requirement (section 4) to Preserve
its Title (section 1) will typically require changing the actual
title.


9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document
except as expressly provided under this License.  Any attempt
otherwise to copy, modify, sublicense, or distribute it is void, and
will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license
from a particular copyright holder is reinstated (a) provisionally,
unless and until the copyright holder explicitly and finally
terminates your license, and (b) permanently, if the copyright holder
fails to notify you of the violation by some reasonable means prior to
60 days after the cessation.

Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, receipt of a copy of some or all of the same material does
not give you any rights to use it.


10. FUTURE REVISIONS OF THIS LICENSE
```

(continues on next page)

(continued from previous page)

```
The Free Software Foundation may publish new, revised versions of the
GNU Free Documentation License from time to time.  Such new versions
will be similar in spirit to the present version, but may differ in
detail to address new problems or concerns.  See
https://www.gnu.org/licenses/.

Each version of the License is given a distinguishing version number.
If the Document specifies that a particular numbered version of this
License "or any later version" applies to it, you have the option of
following the terms and conditions either of that specified version or
of any later version that has been published (not as a draft) by the
Free Software Foundation.  If the Document does not specify a version
number of this License, you may choose any version ever published (not
as a draft) by the Free Software Foundation.  If the Document
specifies that a proxy can decide which future versions of this
License can be used, that proxy's public statement of acceptance of a
version permanently authorizes you to choose that version for the
Document.

11. RELICENSING

"Massive Multiauthor Collaboration Site" (or "MMC Site") means any
World Wide Web server that publishes copyrightable works and also
provides prominent facilities for anybody to edit those works.  A
public wiki that anybody can edit is an example of such a server.  A
"Massive Multiauthor Collaboration" (or "MMC") contained in the site
means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0
license published by Creative Commons Corporation, a not-for-profit
corporation with a principal place of business in San Francisco,
California, as well as future copyleft versions of that license
published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in
part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this
License, and if all works that were first published under this License
somewhere other than this MMC, and subsequently incorporated in whole or
in part into the MMC, (1) had no cover texts or invariant sections, and
(2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site
under CC-BY-SA on the same site at any time before August 1, 2009,
provided the MMC is eligible for relicensing.


ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of
the License in the document and put the following copyright and
license notices just after the title page:

    Copyright (c)  YEAR  YOUR NAME.
    Permission is granted to copy, distribute and/or modify this document
    under the terms of the GNU Free Documentation License, Version 1.3
```

(continues on next page)

(continued from previous page)

```
    or any later version published by the Free Software Foundation;
    with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
    A copy of the license is included in the section entitled "GNU
    Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts,
replace the "with...Texts." line with this:

    with the Invariant Sections being LIST THEIR TITLES, with the
    Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other
combination of the three, merge those two alternatives to suit the
situation.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```

3.1.3 Artwork Licence: CC BY-NC 4.0

VDAT artwork by Majo Buro.

The artwork consist of:

- the logo with the `VDAT` text shown in upper left corner of the documentation;
- the two splash screen visible when starting VDAT;
- the VDAT window icon.

```
Attribution-NonCommercial 4.0 International

=======================================================================

Creative Commons Corporation ("Creative Commons") is not a law firm and
does not provide legal services or legal advice. Distribution of
Creative Commons public licenses does not create a lawyer-client or
other relationship. Creative Commons makes its licenses and related
information available on an "as-is" basis. Creative Commons gives no
warranties regarding its licenses, any material licensed under their
terms and conditions, or any related information. Creative Commons
disclaims all liability for damages resulting from their use to the
fullest extent possible.

Using Creative Commons Public Licenses

Creative Commons public licenses provide a standard set of terms and
conditions that creators and other rights holders may use to share
original works of authorship and other material subject to copyright
and certain other rights specified in the public license below. The
following considerations are for informational purposes only, are not
exhaustive, and do not form part of our licenses.

     Considerations for licensors: Our public licenses are
```

(continues on next page)

(continued from previous page)

```
     intended for use by those authorized to give the public
     permission to use material in ways otherwise restricted by
     copyright and certain other rights. Our licenses are
     irrevocable. Licensors should read and understand the terms
     and conditions of the license they choose before applying it.
     Licensors should also secure all rights necessary before
     applying our licenses so that the public can reuse the
     material as expected. Licensors should clearly mark any
     material not subject to the license. This includes other CC-
     licensed material, or material used under an exception or
     limitation to copyright. More considerations for licensors:
        wiki.creativecommons.org/Considerations_for_licensors

     Considerations for the public: By using one of our public
     licenses, a licensor grants the public permission to use the
     licensed material under specified terms and conditions. If
     the licensor's permission is not necessary for any reason--for
     example, because of any applicable exception or limitation to
     copyright--then that use is not regulated by the license. Our
     licenses grant only permissions under copyright and certain
     other rights that a licensor has authority to grant. Use of
     the licensed material may still be restricted for other
     reasons, including because others have copyright or other
     rights in the material. A licensor may make special requests,
     such as asking that all changes be marked or described.
     Although not required by our licenses, you are encouraged to
     respect those requests where reasonable. More considerations
     for the public:
        wiki.creativecommons.org/Considerations_for_licensees

=======================================================================

Creative Commons Attribution-NonCommercial 4.0 International Public
License

By exercising the Licensed Rights (defined below), You accept and agree
to be bound by the terms and conditions of this Creative Commons
Attribution-NonCommercial 4.0 International Public License ("Public
License"). To the extent this Public License may be interpreted as a
contract, You are granted the Licensed Rights in consideration of Your
acceptance of these terms and conditions, and the Licensor grants You
such rights in consideration of benefits the Licensor receives from
making the Licensed Material available under these terms and
conditions.


Section 1 -- Definitions.

  a. Adapted Material means material subject to Copyright and Similar
     Rights that is derived from or based upon the Licensed Material
     and in which the Licensed Material is translated, altered,
     arranged, transformed, or otherwise modified in a manner requiring
     permission under the Copyright and Similar Rights held by the
     Licensor. For purposes of this Public License, where the Licensed
     Material is a musical work, performance, or sound recording,
     Adapted Material is always produced where the Licensed Material is
     synched in timed relation with a moving image.
```

(continues on next page)

(continued from previous page)

```
b. Adapter's License means the license You apply to Your Copyright
   and Similar Rights in Your contributions to Adapted Material in
   accordance with the terms and conditions of this Public License.

c. Copyright and Similar Rights means copyright and/or similar rights
   closely related to copyright including, without limitation,
   performance, broadcast, sound recording, and Sui Generis Database
   Rights, without regard to how the rights are labeled or
   categorized. For purposes of this Public License, the rights
   specified in Section 2(b)(1)-(2) are not Copyright and Similar
   Rights.
d. Effective Technological Measures means those measures that, in the
   absence of proper authority, may not be circumvented under laws
   fulfilling obligations under Article 11 of the WIPO Copyright
   Treaty adopted on December 20, 1996, and/or similar international
   agreements.

e. Exceptions and Limitations means fair use, fair dealing, and/or
   any other exception or limitation to Copyright and Similar Rights
   that applies to Your use of the Licensed Material.

f. Licensed Material means the artistic or literary work, database,
   or other material to which the Licensor applied this Public
   License.

g. Licensed Rights means the rights granted to You subject to the
   terms and conditions of this Public License, which are limited to
   all Copyright and Similar Rights that apply to Your use of the
   Licensed Material and that the Licensor has authority to license.

h. Licensor means the individual(s) or entity(ies) granting rights
   under this Public License.

i. NonCommercial means not primarily intended for or directed towards
   commercial advantage or monetary compensation. For purposes of
   this Public License, the exchange of the Licensed Material for
   other material subject to Copyright and Similar Rights by digital
   file-sharing or similar means is NonCommercial provided there is
   no payment of monetary compensation in connection with the
   exchange.

j. Share means to provide material to the public by any means or
   process that requires permission under the Licensed Rights, such
   as reproduction, public display, public performance, distribution,
   dissemination, communication, or importation, and to make material
   available to the public including in ways that members of the
   public may access the material from a place and at a time
   individually chosen by them.

k. Sui Generis Database Rights means rights other than copyright
   resulting from Directive 96/9/EC of the European Parliament and of
   the Council of 11 March 1996 on the legal protection of databases,
   as amended and/or succeeded, as well as other essentially
   equivalent rights anywhere in the world.

l. You means the individual or entity exercising the Licensed Rights
```

(continues on next page)

(continued from previous page)

```
     under this Public License. Your has a corresponding meaning.


Section 2 -- Scope.

  a. License grant.

       1. Subject to the terms and conditions of this Public License,
          the Licensor hereby grants You a worldwide, royalty-free,
          non-sublicensable, non-exclusive, irrevocable license to
          exercise the Licensed Rights in the Licensed Material to:

            a. reproduce and Share the Licensed Material, in whole or
               in part, for NonCommercial purposes only; and

            b. produce, reproduce, and Share Adapted Material for
               NonCommercial purposes only.

       2. Exceptions and Limitations. For the avoidance of doubt, where
          Exceptions and Limitations apply to Your use, this Public
          License does not apply, and You do not need to comply with
          its terms and conditions.

       3. Term. The term of this Public License is specified in Section
          6(a).

       4. Media and formats; technical modifications allowed. The
          Licensor authorizes You to exercise the Licensed Rights in
          all media and formats whether now known or hereafter created,
          and to make technical modifications necessary to do so. The
          Licensor waives and/or agrees not to assert any right or
          authority to forbid You from making technical modifications
          necessary to exercise the Licensed Rights, including
          technical modifications necessary to circumvent Effective
          Technological Measures. For purposes of this Public License,
          simply making modifications authorized by this Section 2(a)
          (4) never produces Adapted Material.

       5. Downstream recipients.

            a. Offer from the Licensor -- Licensed Material. Every
               recipient of the Licensed Material automatically
               receives an offer from the Licensor to exercise the
               Licensed Rights under the terms and conditions of this
               Public License.

            b. No downstream restrictions. You may not offer or impose
               any additional or different terms or conditions on, or
               apply any Effective Technological Measures to, the
               Licensed Material if doing so restricts exercise of the
               Licensed Rights by any recipient of the Licensed
               Material.

       6. No endorsement. Nothing in this Public License constitutes or
          may be construed as permission to assert or imply that You
          are, or that Your use of the Licensed Material is, connected
          with, or sponsored, endorsed, or granted official status by,
```

(continues on next page)

(continued from previous page)

```
          the Licensor or others designated to receive attribution as
          provided in Section 3(a)(1)(A)(i).

  b. Other rights.

       1. Moral rights, such as the right of integrity, are not
          licensed under this Public License, nor are publicity,
          privacy, and/or other similar personality rights; however, to
          the extent possible, the Licensor waives and/or agrees not to
          assert any such rights held by the Licensor to the limited
          extent necessary to allow You to exercise the Licensed
          Rights, but not otherwise.

       2. Patent and trademark rights are not licensed under this
          Public License.

       3. To the extent possible, the Licensor waives any right to
          collect royalties from You for the exercise of the Licensed
          Rights, whether directly or through a collecting society
          under any voluntary or waivable statutory or compulsory
          licensing scheme. In all other cases the Licensor expressly
          reserves any right to collect such royalties, including when
          the Licensed Material is used other than for NonCommercial
          purposes.


Section 3 -- License Conditions.

Your exercise of the Licensed Rights is expressly made subject to the
following conditions.

  a. Attribution.

       1. If You Share the Licensed Material (including in modified
          form), You must:

            a. retain the following if it is supplied by the Licensor
               with the Licensed Material:

                 i. identification of the creator(s) of the Licensed
                    Material and any others designated to receive
                    attribution, in any reasonable manner requested by
                    the Licensor (including by pseudonym if
                    designated);

                ii. a copyright notice;

               iii. a notice that refers to this Public License;

                iv. a notice that refers to the disclaimer of
                    warranties;

                 v. a URI or hyperlink to the Licensed Material to the
                    extent reasonably practicable;

            b. indicate if You modified the Licensed Material and
               retain an indication of any previous modifications; and
```

(continues on next page)

(continued from previous page)

```
            c. indicate the Licensed Material is licensed under this
               Public License, and include the text of, or the URI or
               hyperlink to, this Public License.

       2. You may satisfy the conditions in Section 3(a)(1) in any
          reasonable manner based on the medium, means, and context in
          which You Share the Licensed Material. For example, it may be
          reasonable to satisfy the conditions by providing a URI or
          hyperlink to a resource that includes the required
          information.

       3. If requested by the Licensor, You must remove any of the
          information required by Section 3(a)(1)(A) to the extent
          reasonably practicable.

       4. If You Share Adapted Material You produce, the Adapter's
          License You apply must not prevent recipients of the Adapted
          Material from complying with this Public License.


Section 4 -- Sui Generis Database Rights.

Where the Licensed Rights include Sui Generis Database Rights that
apply to Your use of the Licensed Material:

  a. for the avoidance of doubt, Section 2(a)(1) grants You the right
     to extract, reuse, reproduce, and Share all or a substantial
     portion of the contents of the database for NonCommercial purposes
     only;

  b. if You include all or a substantial portion of the database
     contents in a database in which You have Sui Generis Database
     Rights, then the database in which You have Sui Generis Database
     Rights (but not its individual contents) is Adapted Material; and

  c. You must comply with the conditions in Section 3(a) if You Share
     all or a substantial portion of the contents of the database.

For the avoidance of doubt, this Section 4 supplements and does not
replace Your obligations under this Public License where the Licensed
Rights include other Copyright and Similar Rights.


Section 5 -- Disclaimer of Warranties and Limitation of Liability.

  a. UNLESS OTHERWISE SEPARATELY UNDERTAKEN BY THE LICENSOR, TO THE
     EXTENT POSSIBLE, THE LICENSOR OFFERS THE LICENSED MATERIAL AS-IS
     AND AS-AVAILABLE, AND MAKES NO REPRESENTATIONS OR WARRANTIES OF
     ANY KIND CONCERNING THE LICENSED MATERIAL, WHETHER EXPRESS,
     IMPLIED, STATUTORY, OR OTHER. THIS INCLUDES, WITHOUT LIMITATION,
     WARRANTIES OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR
     PURPOSE, NON-INFRINGEMENT, ABSENCE OF LATENT OR OTHER DEFECTS,
     ACCURACY, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT
     KNOWN OR DISCOVERABLE. WHERE DISCLAIMERS OF WARRANTIES ARE NOT
     ALLOWED IN FULL OR IN PART, THIS DISCLAIMER MAY NOT APPLY TO YOU.
```

(continues on next page)

(continued from previous page)

```
  b. TO THE EXTENT POSSIBLE, IN NO EVENT WILL THE LICENSOR BE LIABLE
     TO YOU ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION,
     NEGLIGENCE) OR OTHERWISE FOR ANY DIRECT, SPECIAL, INDIRECT,
     INCIDENTAL, CONSEQUENTIAL, PUNITIVE, EXEMPLARY, OR OTHER LOSSES,
     COSTS, EXPENSES, OR DAMAGES ARISING OUT OF THIS PUBLIC LICENSE OR
     USE OF THE LICENSED MATERIAL, EVEN IF THE LICENSOR HAS BEEN
     ADVISED OF THE POSSIBILITY OF SUCH LOSSES, COSTS, EXPENSES, OR
     DAMAGES. WHERE A LIMITATION OF LIABILITY IS NOT ALLOWED IN FULL OR
     IN PART, THIS LIMITATION MAY NOT APPLY TO YOU.

  c. The disclaimer of warranties and limitation of liability provided
     above shall be interpreted in a manner that, to the extent
     possible, most closely approximates an absolute disclaimer and
     waiver of all liability.


Section 6 -- Term and Termination.

  a. This Public License applies for the term of the Copyright and
     Similar Rights licensed here. However, if You fail to comply with
     this Public License, then Your rights under this Public License
     terminate automatically.

  b. Where Your right to use the Licensed Material has terminated under
     Section 6(a), it reinstates:

       1. automatically as of the date the violation is cured, provided
          it is cured within 30 days of Your discovery of the
          violation; or

       2. upon express reinstatement by the Licensor.

     For the avoidance of doubt, this Section 6(b) does not affect any
     right the Licensor may have to seek remedies for Your violations
     of this Public License.

  c. For the avoidance of doubt, the Licensor may also offer the
     Licensed Material under separate terms or conditions or stop
     distributing the Licensed Material at any time; however, doing so
     will not terminate this Public License.

  d. Sections 1, 5, 6, 7, and 8 survive termination of this Public
     License.


Section 7 -- Other Terms and Conditions.

  a. The Licensor shall not be bound by any additional or different
     terms or conditions communicated by You unless expressly agreed.

  b. Any arrangements, understandings, or agreements regarding the
     Licensed Material not stated herein are separate from and
     independent of the terms and conditions of this Public License.


Section 8 -- Interpretation.
```

(continues on next page)

(continued from previous page)

```
  a. For the avoidance of doubt, this Public License does not, and
     shall not be interpreted to, reduce, limit, restrict, or impose
     conditions on any use of the Licensed Material that could lawfully
     be made without permission under this Public License.

  b. To the extent possible, if any provision of this Public License is
     deemed unenforceable, it shall be automatically reformed to the
     minimum extent necessary to make it enforceable. If the provision
     cannot be reformed, it shall be severed from this Public License
     without affecting the enforceability of the remaining terms and
     conditions.

  c. No term or condition of this Public License will be waived and no
     failure to comply consented to unless expressly agreed to by the
     Licensor.

  d. Nothing in this Public License constitutes or may be interpreted
     as a limitation upon, or waiver of, any privileges and immunities
     that apply to the Licensor or You, including from the legal
     processes of any jurisdiction or authority.

=======================================================================

Creative Commons is not a party to its public
licenses. Notwithstanding, Creative Commons may elect to apply one of
its public licenses to material it publishes and in those instances
will be considered the "Licensor." The text of the Creative Commons
public licenses is dedicated to the public domain under the CC0 Public
Domain Dedication. Except for the limited purpose of indicating that
material is shared under a Creative Commons public license or as
otherwise permitted by the Creative Commons policies published at
creativecommons.org/policies, Creative Commons does not authorize the
use of the trademark "Creative Commons" or any other trademark or logo
of Creative Commons without its prior written consent including,
without limitation, in connection with any unauthorized modifications
to any of its public licenses or any other arrangements,
understandings, or agreements concerning use of licensed material. For
the avoidance of doubt, this paragraph does not form part of the
public licenses.

Creative Commons may be contacted at creativecommons.org.
```

3.2 Authors

The HETDEX collaboration:

- Daniel Farrow <dfarrow@mpe.mpg.de>
- Francesco Montesano <montefra@mpe.mpg.de>
- Jan Snigula <snigula@mpe.mpg.de>
- María José Bustamante Rosell <majoburo@gmail.com>

3.3 Virus Data Analysis Tool release notes

3.3.1 Development version @ trunk

3.3.2 Version 0.9.0

- allow manual resizing of task bar (issue #2326) and the ginga fits viewer (issue #2350)
- expose FitsPlanePanel based tab types (issue #2389)
- extract fibers from pre-sky subtracted frames and from master frames (issue #2441, issue #2442)
- add use multiple primary keys (issue #2454)
- add command to create data cubes and tab type to display them (issue #2390, issue #2370)
- add `all_files` primary keyword type and a use example (issue #2443)
- add tab type to display distortion solutions and fiber models (issue #2386, issue #2371)
- fix long standing bug with log widget putting messages in the wrong place (issue #1764)
- add ability to clear the log widget (issue #2444)
- emit the recap log at the end of a command run with the correct level (issue #2547)
- open log files in window (issue #2474)
- allow relative symlinking (issue #2437)
- allow replacing symlinks (issue #2587)
- port VDAT to `qtpy` (issue #2259)
- Collapse "Reduction Browser" at startup if it has too many items (issue #1343) and add menu options to expand/collapse it (issue #2476)

3.3.3 Version 0.8.0

- Add keyword type that uses the fplane file to map between IDs. (issue #2115)
- Exclude IFUs from the gui (issue #2138) and respect empty columns/rows (issue #2139)
- Add new tab type to display text files. issue #2108

3.3.4 Version 0.7.0

- Make multiprocessing work in macox, needs minimum
- Updated pipeline to use library darks and bias frames, update vdat_commands.yml with the correct path, turned on use of the virus_config lines files by default, again, update the path accordingly.
- Modify in order to use the latest feature of pyhetdex 0.12.0

3.3.5 Version 0.6.1

- fix bug with opening/closing database (issue #1895, issue #1897)

3.3.6 Version 0.6.0

- Rework the cloning and removing of directories (issue #1048)
- Add possibility to remove single exposures (issue #1053)
- move the command logging out of the command interpreter via a signal (issue #1492)
- Add versioning to some configuration file and a 'vdat_config compare' command (issue #1493)
- Add option to backup existing configuration files before copying (issue #1500)
- If the files or configuration entries needed to run the reconstruction do not exist, go ahead and disable reconstruction (issue #1501)
- add database versions (issue #1338)
- add vdat_db executable to check and possibly update the meta data (issue #1511)
- update the database tables (issue #1339, issue #1530)
- add offline documentation (issue #1454)
- add splash screen and icons to VDAT and documentation
- properly close VDAT from the menu bar (issue #1568)
- make the reconstructed object only the first time a directory is selected (issue #1534)
- add the interfaces for the plugins and tabs (issue #1601, issue #1602)
- implement the plugin loading mechanism (issue #1598)
- base classes for the ifu and the focal plane tab widgets (issue #1628)
- make sure thread belongs to some object and are properly destroyed (issue #1646)
- don't trigger selection/deselection on double click (issue #1639)
- reimplement Fits file window (issue #1631)
- reimplement the fplane widget for fits files (issue #1629, issue #1632)
- reimplement the fplane widget for quick reconstructin (issue #1633)
- reimplement the fplane widget to combine the above two (issue #1669)
- fix logging-related issues (issue #1317, issue #1765)

3.3.7 Version 0.5.0

- Implement removal of files and/or thumbnails (issue #1330)
- update to use the new fplane format (issue #1461)
- Improved documentation

3.3.8 Version 0.4.0

- completely remove matplotlib and aplpy references
- update command line
- rewrite the GUI as per issue #1155
- update configuration files with the latest reduction steps

- visualize the reconstructed images in the IFU viewer; fixes issue #1407
- fix memory leak, issue #1412
- fix bug with sqlite maximum number of insert/select queries
- resolve issue #1408: the gui starts also when no entry is found in the database
- resolve issue #1465: right-click menu shows up when the mouse is over a selectable directory
- Reduction browser documentation improved
- resolve issue #1410: initialize the reduction browser model in the class itself to ease (re)creation
- connect the progress and the status bar to the command interpreter
- add documentation of the queue and the progress and status bar
- improve documentation on how to install and start VDAT
- add support for twilight shots

3.3.9 Version 0.3.0

- resolve issue #1334: enable multiprocessing
- resolve issue #1335: fix symlinking testing
- resolve issue #1345: bulk insert in the database of already symlinked directories
- fix issue #1381; now command logging mechanism works
- new command line; issue #1337
- support for a directory with "virus" instrument name below the night; issue #1358

3.3.10 Version 0.2.4

- command interpreter: fix bug with file collection in directories containing characters like +
- command interpreter: don't recurse into target directory
- command interpreter: correctly report exceptions happening in the command interpreter

3.3.11 Version 0.2.3

- allow redo symlinking from the gui

3.3.12 Version 0.2.2

- Fix bug with empty OBJECT in science shots
- Allow symlinking different science shots with the same OBJECT by appending a counter
- Added tooltip to the Reduction Browser to show info about the directories
- command interpreter core: log message when starting running a command now prints also the selected directory

3.3.13 Version 0.2.1

- command interpreter types: allow to format header keywords using python string formatting syntax
- print version number from command line and show it in the "help" entry in the menu bar
- add link to the documentation in the menu bar
- fix symlinking documentation.

3.3.14 Version 0.2.0

WARNING: this release has some non-backward compatible changes in some of the configuration files.

- symlinking improved and made more flexible; documentation and testing improved;
- tree view pane: thoroughly tested; basic support for non standard file types (e.g. drk)
- command interpreter signals: renamed and completely rewritten and tested. Documentation improved
- command interpreter types: regex type is now more reliable and produces helpful error messages when the substitutions fails; other types extended and streamlined; documentation and testing improved.
- command interpreter core: some bugfix, execution information emitted via signals.
- unused files removed.

3.3.15 Version 0.1.0

- first version distributed via pypi-like server

3.4 Changelog

```
2018-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump to v0.9.0-post

2018-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/static/VirusDataAnalysisTool.qch: make sure that the offline␣
↪docs
          are up-to-date
    * setup.py: prepare for release v0.9.0
        * ReleaseNotes.rst: same

2018-06-20 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/collapse_tree into ^/trunk
        * ReleaseNotes.rst: add rst links to issues

2018-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: when showing the reduction browser the first
          time, collapse all if the tree is too long; resolves #1343
    * tests/test_gui/test_tree_view.py: try to test it
```

```
    * doc/_source/gui/tree_view.rst: add a note about it
        * ReleaseNotes.rst: update

2018-06-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: better action name
    * doc/_source/_static/menubar_view.png: update screenshot
    * doc/_source/gui/menu_bar.rst: document the change; resolves issue #2476

2018-06-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwindow.py: connect the new menu actions with the tree view
          slots to expand and collapse; second part of issue #2476
    * tests/test_gui/test_mainwindow.py: test it
    * tests/test_gui/conftest.py: move fixtures here
    * tests/test_gui/test_tree_view.py: from here

2018-06-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: add a menu with tree view actions; first part
          of #2476
    * vdat/gui/menubar.py: add it to the menu bar and re-emit the new signals
    * tests/test_gui/test_menus_actions.py: test the changes
    * tests/test_gui/test_menubar.py: same

2018-06-18 Francesco Montesano <montefra@mpe.mpg.de>

   * doc/_source/index.rst: add ReadTheDocs link and logo, add release notes
   * doc/_source/release_notes.rst: same
   * ReleaseNotes.rst: convert to rst

2018-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: makes pyds9 optional as it has always meant to be (blame
          ReadTheDocs)
    * doc/_source/install.rst: update the docs
    * vdat/gui/tabs/ifu_viewer.py: improve the message when ds9 is not␣
↪installed
    * tests/test_gui/test_tabs/test_ifu_viewer.py: update the tests
    * tox.ini: install pyds9
    * requirements_rtd.txt: don't install pyds9 on ReadTheDocs

2018-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * requirements_rtd.txt: install pyds9

2018-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * readthedocs.yml: disable conda
    * requirements_rtd.txt: install pyqt5

2018-06-15 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/qtpy into ^/trunk
```

```
2018-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/install.rst: add info about qtpy and the Qt backends.␣
↪Resolves
          issue #2259
    * doc/_source/contributions.rst: same
    * doc/_source/conf.py: add link to issues

2018-06-13 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: PySide2 is not supported by some dependency
    * vdat/gui/menus_actions.py: add NOTE about pyqtProperty.deleter

2018-06-12 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: pyside crashes, use pyqt4 on py34

2018-06-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: super initializiation
    * vdat/gui/queue.py: invert class inheritance order
    * vdat/gui/tabs/ifu_widget.py: same
    * tests/test_gui/test_queue.py: add sleep to make tests more reliable
    * tests/test_gui/test_tabs/test_ifu_widget.py: relax test
    * tests/test_gui/test_tabs/test_tab_widget.py: work around a test problem

2018-06-11 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: py27 tests PyQt4
    * tests/test_gui/conftest.py: fix problems with PyQt4
    * tests/test_gui/test_help_window.py: same
    * tests/test_gui/test_tabs/test_ifu_viewer.py: same

2018-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_help_window.py: fix import
    * tests/test_gui/test_mainwindow.py: same
    * tests/test_gui/test_tabs/test_ifu_widget.py: same
    * tests/test_gui/test_tree_view.py: same
    * vdat/gui/tabs/ifu_widget.py: same
    * tox.ini: add variable needed to run the tests

2018-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: use qtpy>=1.1
    * vdat/gui/*.py: pyqt{Signal,Slot,Property} lost a ``pyqt``
    * tests/test_gui/test_central.py: same
    * vdat/gui/tabs/ifu_widget.py: same and fix one Q-stuff

2018-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * pytest.ini: remove qt_api
    * tox.ini: same
```

```
    * tests/conftest.py: remove sip calls
    * tests/test_gui/*py: move to qtpy; second part of #2259

2018-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: remove also all sip calls
    * vdat/gui/queue.py: remove non used import

2018-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: get rid of the pyqt4 tester and add qtpy as dependency
        * tox.ini: install pyqt5 and don't symlink pyqt in the virtual␣
↪environment
    * vdat/gui: convert documentation to PyQt5 and use qtpy; first part of
↪#2259
    * vdat/command_interpreter/signals.py: update comments accordingly

2018-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/symlink into ^/trunk

2018-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/trunk into ^/branches/symlink

2018-06-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: if required, replace the symlinks; resolves␣
↪issue
          #2587
    * vdat/config/vdat_setting.cfg: add the corresponding option
    * tests/test_libvdat/test_symlink.py: test the change
    * tests/test_gui/test_mainwidget.py: update the tests
    * doc/_source/dirstruct.rst: update
    * ReleaseNotes.md: udpate

2018-06-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add possibility to do relative symlinking;
          resolves #2437
    * vdat/config/vdat_setting.cfg: add the option
    * vdat/libvdat/vdat.py: and expose it to the command line
    * vdat/config/versions.py: bump the config version
    * doc/_source/dirstruct.rst: document the change
    * tests/test_libvdat/test_symlink.py: test the changes
    * tests/test_gui/test_mainwidget.py: same
    * tests/test_config/test_versions.py: same
    * ReleaseNotes.md: update

2018-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: add module docstring
    * vdat/gui/tabs/ifu_widget.py: fix typo
```

```
2018-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: add configuration option and fixture for the extra
          data files
    * tests/test_ci/test_command_interpreter.py: use the new fixture
    * tests/test_gui/test_tabs/test_ifu_widget.py: same
        * doc/_source/contributions.rst: document the new extra test data␣
↪repository

2018-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/data/raw: remove it to speedup checkouts

2018-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * readthedocs.yml: install vdat as ocd, but in conda
    * requirements_rtd.txt: install pyhetdex from extra index
    * environment.yml: remove pip dependences

2018-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * environment.yml: get matplotlib from conda, use quotes for the pip
          options, cleanup pip packages

2018-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * environment.yml: remove conda dependency

2018-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * environment.yml: remove the prefix at the end of the file

2018-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * environment.yml: conda environment file
    * readthedocs.yml: tell readthedocs to use the above file
    * vdat/gui/tabs/ifu_viewer.py: if pyds9 is not importable, deal with it
    * doc/_source/install.rst: removed non used footnotes

2018-06-05 Francesco Montesano <montefra@mpe.mpg.de>

    * LICENSE_cc: added
    * doc/_source/license.rst: add licence
    * doc/_source/index.rst: add notice
    * vdat/gui/mainwindow.py: PEP8

2018-05-08 Francesco Montesano <montefra@mpe.mpg.de>

    * LICENSE_fdl1.3: added
    * LICENSE_gpl3: added
    * doc/_source/license.rst: add licenses, resolves issue #1757
    * doc/_source/index.rst: add copyright notice
    * vdat/**.py: same
    * doc/_source/command_interpreter.rst: fix typo
```

```
2018-05-08 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/logs into ^/trunk

2018-05-08 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/menu_bar.rst: add section about the log menu. Resolves
          issue #2474
    * doc/_source/_static/menubar_log*.png: add new screenshot
    * doc/_source/_static/menubar_log_window.png
    * doc/_source/_static/menubar_*.png: update existing ones
    * vdat/gui/static/VirusDataAnalysisTool.qch: update
    * vdat/gui/menus_actions.py: update docs and log window button
    * ReleaseNotes.md: update

2018-05-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: add refresh button. Third part of issue #2474
    * tests/test_gui/test_menus_actions.py: test the changes

2018-05-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: open read-only text window to display a log
          file. Second part of issue #2474
    * tests/test_gui/test_menus_actions.py:

2018-05-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: add actions to open log files (callback not␣
↪yet
          implemented). First part of issue #2474
    * vdat/gui/menubar.py: change an import
    * tests/test_gui/test_menus_actions.py: test it
    * tests/test_gui/test_menubar.py: same

2018-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: the level of the message printed out␣
↪at
          the end of the run method depend on the success of the command␣
↪executions;
          resolves #2547
    * tests/test_ci/test_command_interpreter.py: test it
        * ReleaseNotes.md: update

2018-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: create a menu entry for the logs and an␣
↪action
          to clear the log widget; resolves issue #2444
    * vdat/gui/menubar.py: plug it into the menu bar
    * vdat/gui/mainwindow.py: and connect the signals
    * tests/test_gui/test_menubar.py: test the signals
```

```
    * tests/test_gui/test_mainwindow.py: and the connections
        * ReleaseNotes.md: update

2018-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/logger_widget.py: make sure to append the new log message;
          resolves issue #1764; improve docs and cleanup unused code
    * tests/test_gui/test_logger_widget.py: test the widget
    * doc/_source/codedoc/gui/logger_widget.rst: add the module in the docs
    * doc/_source/codedoc/gui/index.rst: and in the index
        * ReleaseNotes.md: update

2018-04-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/static/VirusDataAnalysisTool.qch: update the offline␣
↪documentation
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2018-04-13 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/display_distortion into ^/trunk

2018-04-12 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_tabs/test_ifu_widget.py: makes sure to end the
          QPainter to avoid coredumps

2018-04-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: use the fits_multiext to display fiber model.
          Resolves #2371
    * ReleaseNotes.md: add changes

2018-04-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add extension to tab name
    * doc/_source/gui/main_panel.rst: document the fits_multiext tab type␣
↪(fifth
          part of #2371)
    * doc/_source/gui/ifu_viewer.rst: and the changes to the FITS viewer
    * doc/_source/_static/fits_viewer_ext.png: add screenshot
    * doc/_source/_static/fplane_fits_multiext.png: same

2018-04-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: display in ginga one extension and send the
          same extension to DS9 (fourth part of #2371). Fix bug with badly␣
↪named
          variables
    * tests/test_gui/test_tabs/test_ifu_viewer.py
    * vdat/utilities.py: deal with itertools name changes in python2/3

2018-04-12 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/tabs/entry_points.py: add ``ext`` in the formatting dictionary
          (third part of #2371)
    * tests/test_gui/test_fplane.py: make sure that it works

2018-04-12 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add the fits_multiext tab type
    * vdat/gui/tabs/entry_points.py: its entry point implementation
    * vdat/gui/tabs/tab_widget.py: its tab implementation
    * vdat/gui/tabs/ifu_widget.py: and its ifu implementation
    * tests/test_gui/test_fplane.py: test it
    * tests/test_gui/test_tabs/test_ifu_widget.py: same
    * tests/test_list_plugins.py: update accordingly

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: create thumbnail also for multi extension files␣
↪(first
          part of #2371)
    * tests/data/mastertwi_L.fmod: add a test file
    * tests/test_gui/conftest.py: and a fixture about it
    * tests/test_gui/test_gui_utils.py: and test the changes

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^trunk into ^/branches/display_distortion

2018-03-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add the dist tab (properly resolving #2386)
    * vdat/config/versions.py: bump the task minor version

2018-03-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: fix docstring
    * vdat/gui/tabs/ifu_widget.py: add rescaling of fonts in QImage
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * doc/_source/gui/main_panel.rst: document the new dist tab type. Resolves
          issue #2386
    * doc/_source/gui/ifu_viewer.rst: document the new dist window
    * doc/_source/_static/dist_viewer.png: add screenshot
    * doc/_source/_static/fplane_dist.png: same

2018-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add possibility to send only the region to␣
↪an
          existing ds9 frame
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test the changes

2018-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/utilities.py: add a chunk iterator
    * vdat/gui/tabs/ifu_viewer.py: send a chunk of regions at a time to speed␣
```

```
↪it
          up; add spinning wheel (eighth part of #2386)
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test it
    * tests/test_utilities.py: same

2018-03-20 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_tabs/test_ifu_widget.py: better isolate tests

2018-03-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: the DS9 menu must not add a new frame
    * tests/test_gui/test_tabs/test_ifu_viewer.py: update the tests

2018-03-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: on double click, open a DistWindow (seventh
          part of #2386)
    * vdat/gui/tabs/ifu_viewer.py: small improvements
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the double clicking

2018-03-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add DistWindow (sixth part of #2386)
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test the changes
    * tests/test_gui/conftest.py: move around fixures
    * tests/test_gui/test_tabs/test_ifu_widget.py: same
    * tests/data/mastertwi_L.dist.reg: add an example region file

2018-03-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: extract the ds9 menu from the fits window
          (fifth part of #2386)
    * tests/test_gui/test_tabs/test_ifu_viewer.py: update accordingly

2018-03-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: add an IFU widget to display distortion␣
↪files
          (fourth part of #2386)
    * vdat/gui/tabs/tab_widget.py: use it
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/test_fplane.py: extend a bit the test
    * tests/data/mastertwi_L.dist: add a test distortion file

2018-03-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: fix bug introduced by the new IFU class
    * tests/test_gui/test_tabs/test_ifu_widget.py: add regression test
    * pytest.ini: strict xfait

2018-03-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: modify the IFUFitsWidget using the new␣
```

```
↪class
          (third part of #2386)
    * tests/test_gui/test_tabs/test_ifu_widget.py: modify the tests␣
↪accordingly

2018-03-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: refactor the code to create a grid of␣
↪images
          in the IFU (second part of #2386)

2018-03-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: isolate the thumbnail item class

2018-03-01 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add the new dist tab type (first part of issue #2386)
    * vdat/gui/tabs/entry_points.py: and it's interface
    * vdat/gui/tabs/tab_widget.py: and a first, non working, version to the␣
↪tab
          type
    * tests/test_gui/test_fplane.py: test the changes
    * tests/test_list_plugins.py: same

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: simplify the make cube step definition

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/all_files_primary into ^/trunk

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: add an example command using ``all_
↪files``.
          Resolves #2443.
    * vdat/config/tasks.yml: add a task using the above command
    * doc/_source/command_interpreter.rst: add note about ``all_files`` and
          ``filter_selected``
    * tests/test_libvdat/test_loggers.py: update tests

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: document the ``all_files`` primary
          type (second part of #2443)

2018-04-11 Francesco Montesano <montefra@mpe.mpg.de>

        * setup.py: add the new ``all_files`` primary type (first part of
↪#2443)
    * vdat/command_interpreter/types.py: implement it
    * tests/test_ci/test_types.py: test the changes
```

```
2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/datacube into ^/trunk

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: describe the new tab type (resolves␣
↪issue
          #2370)
    * doc/_source/_static/fplane_fits_cube.png: add a screen shot
        * doc/_source/gui/ifu_viewer.rst: add some small information about␣
↪data
          cubes
    * vdat/gui/tabs/entry_points.py: fix the docs
    * ReleaseNotes.md: update

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: pep8
    * vdat/gui/tabs/ifu_viewer.py: when displaying a 3d fits, set the image to
          the nanmedia of the z_indx (if available) or of the whole range␣
↪(fourth
          part of #2370)
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test the changes

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: add z_indx in the thumbnail name (third␣
↪part
          of #2370)
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * vdat/config/tasks.yml: use the new fits_cube tab type and add some␣
↪example
          of z_indx

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add fits_cube tab type entry point (second part of #2370)
    * vdat/gui/tabs/entry_points.py: and its implementation
    * vdat/gui/tabs/tab_widget.py: the corresponding tab type
    * vdat/gui/tabs/ifu_widget.py: and ifu widget
    * tests/test_gui/test_fplane.py: test the changes
    * tests/test_gui/test_gui_utils.py: same
    * tests/test_gui/test_tabs/test_ifu_widget.py: same
    * tests/test_list_plugins.py: same
    * tests/test_gui/conftest.py: same
    * vdat/exceptions.py: add VDAT exceptions
    * doc/_source/codedoc/exceptions.rst: add them to the documentation
    * doc/_source/codedoc/index.rst: same
    * doc/Makefile: rebuild also if something in vdat/ changes

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/utils.py: add support for data cube thumbnail creation
    * tests/data/CuFeSpdsses_084.fits: add datacube example
    * tests/test_gui/test_gui_utils.py: add tests for datacube

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/datacube

2018-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: add command to make cubes; resolves #2390
    * vdat/config/tasks.yml: add task to run and visualise it
    * vdat/config/versions.py: update the version
    * ReleaseNotes.md: update
    * tests/test_libvdat/test_loggers.py: update the tests

2018-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: fix missing pointer
    * vdat/database/base.py: something changed on peewee/sqlite, threadsafe
          option doesn't work anymore
    * vdat/database/check.py: missing columns now raise an AttributeError
    * vdat/libvdat/symlink.py: same
    * tests/test_database.py: update the tests
    * tests/test_gui/test_queue.py: update according to the primary key␣
↪changes
    * tests/test_gui/test_tree_view.py: make the tooltip change more stable

2018-04-09 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/extract_fibers into ^/trunk

2018-04-09 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update

2018-03-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: use previous commit to define a single
          fiberextract command and add extraction of master files. Resolves␣
↪issue
          #2441
    * vdat/config/tasks.yml: add fiber extraction in the cal tabs

2018-03-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: make primary accept also a list.
          Resolves issue #2454
    * doc/_source/command_interpreter.rst: update the documentation
    * tests/test_ci/test_command_interpreter.py: and the tests
    * tests/conftest.py: set the caplog logger level to DEBUG

2018-03-28 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/vdat_commands.yml: add command to extract fibers from pre-
↪sky
          subtracted images
    * vdat/config/tasks.yml: add the buttons and the tabs in the fiber
          extraction task
    * vdat/config/versions.py: bump the versions

2018-03-05  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: Replace flatfits call for
        mastertwi / masterflat with meanflat

2018-03-02  Jan Snigula  <snigula@mpe.mpg.de>

    * vdat/config/tasks.yml: Use perrow overscan subtraction
    * vdat/config/vdat_commands.yml: Use perrow overscan subtraction

2018-02-27 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/display_distortion

2018-02-27 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/simplefits into ^/trunk

2018-02-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add a ``exp_fits`` and ``fits`` tab type plugin
    * vdat/gui/tabs/entry_points.py: and their implementation; resolves #2389
    * tests/test_gui/test_fplane.py: test it
    * tests/test_list_plugins.py: same
    * doc/_source/gui/main_panel.rst
    * ReleaseNotes.md

2018-02-20 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/resize/ into ^/trunk

2018-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add splitter between the ginga fits viewer
          and the fits header keys list (resolves issue #2350)
    * ReleaseNotes.md: update

2018-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: add splitter between fplane and task list, allow to
          make fplane small (resolves issue #2326)
    * vdat/gui/mainwindow.py: allow slightly smaller VDAT window
        * ReleaseNotes.md: update

2017-12-13 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: use pytest>3.3 and remove pytest-catchlog. Resolves #2257
```

```
    * setup.py: same
    * doc/source/install.rst: update accordingly
    * doc/Makefile: open the browser
    * doc/source/contributions.rst: update accordingly

2017-10-20  Francesco Montesano  <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.8.0-post

2017-10-20  Francesco Montesano  <montefra@mpe.mpg.de>

    * setup.py: set version 0.8.0
    * ReleaseNotes.md: same
    * vdat/gui/static/VirusDataAnalysisTool.qch: update documentation
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-10-20  Francesco Montesano  <montefra@mpe.mpg.de>

    * tests/test_gui/conftest.py: better parametrization of the fplane_file
          fixture
    * tests/test_gui/test_tabs/test_ifu_widget.py: make test pass properly

2017-10-20  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/branches/text_file_tab in ^/trunk

2017-10-20  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add text_file tabs for the detect action

2017-10-18  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: don't set icons, apparently doesn't work on mac...
    * vdat/gui/static/icons: remove the icons
    * vdat/gui/tabs/ifu_viewer.py: don't do line wraps

2017-10-18  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/trunk in ^/branches/text_file_tab

2017-10-18  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add text_file tab for the dither file

2017-10-12  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/static/icons: add a subset of the "macOS Icons" icons
          https://store.kde.org/p/1102582/
    * vdat/gui/__init__.py: if the theme name is not set, set it to the above
          name. Resolves issue #2133

2017-09-27  Francesco Montesano  <montefra@mpe.mpg.de>

    * tests/test_gui/conftest.py: update fixture
```

```
    * tests/test_gui/test_tabs/test_ifu_viewer.py: 100% coverage of the
          ifu_viewer module
    * vdat/gui/tabs/ifu_viewer.py: make some little change to allow testing

2017-09-26  Francesco Montesano  <montefra@mpe.mpg.de>

    * doc/_source/_static/fits_viewer.png: update screenshot
    * doc/_source/_static/text_file_viewer.png: same

2017-09-26  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add the quit and help menu to the ifu␣
↪viewer
          windows

2017-09-25  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/menubar.py: plug the Help menu and get rid of the unnecessary
          signals
        * vdat/gui/mainwindow.py: remove slots and connection incorporated␣
↪into the
          Help menu. Second part of issue #2135
    * vdat/gui/help_window.py: set reasonable size for the help window
    * tests/test_gui/test_mainwindow.py: remove tests

2017-09-25  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: isolate the help menu for better reusability.
          First part of issue #2135
    * tests/test_gui/test_menus_actions.py: test it

2017-09-22  Francesco Montesano  <montefra@mpe.mpg.de>

    * doc/_source/gui/ifu_viewer.rst: describe the new text file window
    * doc/_source/gui/main_panel.rst: finish documentation of the text file␣
↪tab
    * doc/_source/_static/fplane_text_file.png: add screenshot
    * doc/_source/_static/text_file_viewer.png: same

2017-09-21  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: if the file exists, open it in the file
          editor
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it

2017-09-21  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add basic text editor
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test it

2017-09-20  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: create QuitAction
    * vdat/gui/menubar.py: use it
```

```
    * tests/test_gui/test_menus_actions.py: test it

2017-09-18  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: prepare al image with the number of lines
    * vdat/gui/tabs/tab_widget.py: no need to add the cleanup method
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes
    * tests/test_gui/test_tabs/test_tab_widget.py: the fake IFU are not needed

2017-09-13  Francesco Montesano  <montefra@mpe.mpg.de>

        * setup.py: add text_file tab plugin. First part of issue #2108
    * vdat/gui/tabs/entry_points.py: implement the entry point for text_file
    * vdat/gui/tabs/tab_widget.py: implement the tab type (I think that it's
          everything that we need)
    * vdat/gui/tabs/ifu_widget.py: implement the setup method of the ifu␣
↪widget.
          It still doesn't display anything
    * doc/_source/gui/main_panel.rst: begin documenting the changes
    * tests/test_gui/test_fplane.py: test the new plugin
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the initialisation and
          setup of the new ifu widget
    * tests/test_gui/test_tabs/test_tab_widget.py: test the new tab widget
    * tests/test_list_plugins.py: update the number of plugins

2017-10-11  Niv Drory  <drory@astro.as.utexas.edu>

        * config/vdat_commands.yml (subtract_os): remove deceptively evil -s␣
↪flag.

2017-10-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/branches/exclude_ifus into ^/trunk

2017-10-04  Francesco Montesano  <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: add some info about the fplane file

2017-09-29  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: add option to respect empty columns and␣
↪rows
          when displaying the focal plane. Resolves issue #2139
    * vdat/gui/tabs/tab_widget.py: use it
    * vdat/gui/central.py: exclude the IFUs also in the main part of the gui
    * tests/test_gui/conftest.py: adapt the tests
    * tests/test_gui/test_tabs/test_tab_widget.py: same

2017-09-27  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: add option to ignore IFUSLOT. Resolves
          issue #2138
    * vdat/config/versions.py: bump the vdat_setting.cfg version to 1.1.0
    * vdat/gui/tabs/tab_widget.py: propagate the above option to the focal␣
```

```
↪plane
          in the GUI
    * tests/test_config/test_versions.py: update the versions

2017-09-15  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: forgot to copy the updated fplane file (updated
          from fplane20170202.txt in the virus_config repo)

2017-09-15  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/branches/fplane_map into ^/trunk
    * vdat/gui/static/VirusDataAnalysisTool.qch: update
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-09-15  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: use the fplane_map type
    * vdat/config/tasks.yml: update the commands to use it
    * vdat/config/versions.py: update the version

2017-09-14  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: add the fplane_map type. Resolves␣
↪issue
      #2115
    * doc/_source/command_interpreter.rst: document it
    * setup.py: advertise it
    * tests/test_ci/test_types.py: test it

2017-09-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tasks.yml: Fixed mistake which caused some
      tabs to be repeated

2017-09-07  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: Fixed fiberextract mandatories

2017-09-06  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: correct the dither_file executable call
    * vdat/config/vdat_commands.yml: same

2017-09-05  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: fix typo in variable name (modelbalse ->
          modelbase in the mkdither command)
    * vdat/config/tasks.yml: fix the above type, fix error in Fiber extracted
          tab title

2017-08-22  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: Update filename prefixes and
```

```
        pixelflat filenames.
        deformer step creates dists/fmods for flat and twilight.
        flatnorm runs for flat and twilight models.
        * vdat/config/tasks.yml: Moved pixelflat step before ccdcombine
        Added buttons to use twilight dist/fmod for skysubtraction and
        fiberextract

2017-08-21  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: pixflat_lib dir defaults to ./pixel_
↪lib
        * vdat/config/tasks.yml: Use pixelflats by default

2017-08-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/tasks.yml: Use master twilight frames for deformer

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.7.0-post

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: require pyhetdex 0.12.0, bump version to 0.7.0
    * ReleaseNotes.md: update
    * tox.ini: remove devel pyhetdex version dependency
    * vdat/gui/static/VirusDataAnalysisTool.qch: update
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/next_pyhetdex/ into ^/trunk

2017-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_libvdat/test_loggers.py: fix number of log files created

2017-08-11  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/versions.py: Bumped version numbers
        * vdat/config/tasks.yml: More pipeline updates
        * vdat/config/vdat_commands.yml: Same

2017-08-09  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: Use library bias and dark frames
        * vdat/config/tasks.yml: Same
        * vdat/libvdat/vdat.py: Fixed multiprocessing on OS X

2017-04-20 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_gui_utils.py: get rid of warnings
    * tests/test_libvdat/test_symlink.py: same
    * vdat/gui/utils.py: same
```

```
    * tests/test_gui/test_tabs/test_ifu_widget.py: improve float comparison

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: re-add cov-init, use devel version of pyhetdex everywhere

2017-07-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: use new pyhetdex.tools.db_helpers module
    * vdat/libvdat/symlink.py: same
    * tests/test_database.py: remove unnecessary tests
    * vdat/database/__init__.py: remove unnecessary imports

2017-06-28 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: add python 3.6
    * vdat/config/entry_point.py: use pyhetdex functions. Resolves issue #1964
    * tests/test_config/test_entry_point.py: update the tests

2017-06-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: depends on configparser on python 2
    * tox.ini: use devel pyhetdex
    * vdat/config/core.py: use configparser; resolves #1983
    * vdat/config/versions.py: same
    * vdat/gui/menubar.py: same
    * vdat/gui/utils.py: same
    * vdat/libvdat/vdat.py: same
    * tests/test_gui/test_gui_utils.py: same, get rid of warnings
    * tests/test_gui/test_mainwindow.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_loggers.py: same
    * tests/test_libvdat/test_symlink.py: same
    * tests/test_libvdat/test_vdat.py: same

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to v0.6.1-post

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: prepare for v0.6.1 release
    * ReleaseNotes.md: same

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: use pyhetdex override_conf instead of its own
          implementation. Resolves issue #1850
    * tests/test_config/test_core.py: update tests

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/peewee_connect into ^/trunk
```

```
2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_mainwidget.py: wrap every database query/
↪modification
          within a connection. Solved issue #1897
    * tests/test_gui/test_menubar.py: same
    * tests/test_gui/test_menus_actions.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_symlink.py: same
    * vdat/gui/central.py: same
    * vdat/gui/menubar.py: same
    * vdat/gui/menus_actions.py: same
    * vdat/gui/tabs/entry_points.py: same
    * vdat/gui/tabs/tab_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/database/core.py: open and close the database only if it's not on
          memory

2017-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * tox.ini: ignore little-deploy failures
    * vdat/database/core.py: connect only if the database is closed. Fixes␣
↪issue
          #1895.
    * vdat/libvdat/symlink.py: wrap some query into db.connect()

2017-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * scripts/symlink_pyqt.sh: fix bug

2017-02-03 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: prepare v0.6.0 release
    * ReleaseNotes.md: same

2017-02-03 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/branches/debug_logs into ^/trunk

2017-02-03 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk into ^/branches/debug_logs

2017-01-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add option to process events while the thread is
          running
    * tests/test_gui/test_gui_utils.py: test it
    * vdat/gui/mainwidget.py: use it to update the GUI as the symlinking is
          running (fixes #1765)

2017-01-31 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * tests/test_gui/test_mainwidget.py: add tests
    * vdat/gui/mainwidget.py: do only the symlink in the thread

2017-02-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: workaround to fix issue #1782

2017-02-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: resolves issue #1782
    * tests/test_gui/test_tabs/test_ifu_widget.py: adapt mocking in tests

2017-02-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: improve documentation, resolves issue #1779
    * vdat/gui/static/VirusDataAnalysisTool.qch: add the new documenation
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-02-01 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/tabs/ifu_widget.py: uses astropy's zscale. Fixes Issue
↪#1777

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk into ^/branches/debug_logs

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add function to run the FuncQThread with the waiting
          cursor.
    * vdat/gui/mainwindow.py: use it
    * vdat/gui/treeview_model.py: use it
    * vdat/gui/mainwidget.py: use it when running the symlink from the GUI.
          Solves issue #1765
    * tests/test_gui/test_gui_utils.py: add tests
    * tests/test_gui/test_mainwidget.py: added. Need to finish testing the
          widget, but I first need to merge the other branch as it has some␣
↪fixture
          I need

2017-01-27 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: tmp_dir is now a of tmpdir with the possibility to
          remove the directory; add fixtures
    * tests/test_gui/test_init.py: move fixture to conftest
        * tests/test_integration_vdat.py: add regression/integration test for␣
↪the
          bug in #1317
    * vdat/gui/__init__.py: partial refactoring to help testing
    * vdat/libvdat/vdat.py: add argv option to main for testing purposes

2017-01-24 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/__init__.py: don't set the logger level
    * vdat/gui/logger_widget.py: set the handler level to INFO
    * tests/conftest.py: fix sip.setapi to v2 for all the types

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/static/VirusDataAnalysisTool.*: update documentation with the
          latest changes

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/branches/plug_tabs_1533 into ^/trunk

2017-01-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/list_plugins.py: add functionality to list entry points. Solves␣
↪issue
          #1613
    * vdat/gui/fplane.py: move entry point group name to variable
    * tests/test_list_plugins.py: test the new functionality
    * doc/_source/gui/main_panel.rst: add some documentation
    * doc/_source/codedoc/index.rst: same
    * doc/_source/codedoc/list_plugins.rst: same
    * setup.py: add the entry point

2017-01-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: make "Individual" radio button react to
          button release to allow repainting the GUI. Resolved issue #1679
    * doc/_source/gui/main_panel.rst: add some hint about this

2017-01-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: reset individual selection when dropping␣
↪fits
          tabs
    * tests/test_gui/test_tabs/test_tab_widget.py: update the tests

2017-01-10 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: move the new plugin implementation away.
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: move it here and␣
↪update
          the rest

2017-01-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: port all the tasks to the new tab types. Resolves
          #1724

2017-01-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: make sure that the user is not spammed with
          reconstruction error if the reconstruction object is not present.␣
```

```
↪Resolved
          #1725

2016-12-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/interface.py: add the enabled property
    * vdat/gui/fplane.py: set the tab enabled or not
    * tests/test_gui/conftest.py: move fixture here
    * tests/test_gui/test_central.py: from here
    * tests/test_gui/test_fplane.py: test tab enabled

2016-12-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: add overlay if no tab displayed; fixes issue #1603
    * tests/test_gui/test_fplane.py: test it and fix other tests accordingly

2016-12-15 Francesco Montesano <montefra@mpe.mpg.de>

        fixes issue #1720

    * doc/_source/conf.py: add ginga intersphinx
    * vdat/gui/tabs/entry_points.py: remove unused entry points
    * vdat/gui/tabs/ifu_viewer.py: fix documentation and remove unused code
    * vdat/gui/tabs/ifu_widget.py: same

2016-12-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: fix documentation
    * vdat/gui/tabs/interface.py: same
    * vdat/gui/tabs/tab_widget.py: same and remove old code

2016-12-14 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/send_to_ds9.png: added
    * doc/_source/gui/ifu_viewer.rst: add some documentation about the fits
          viewer

2016-12-14 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/ifu_viewer.rst: add screenshot
    * doc/_source/gui/main_panel.rst: add information about the header_keys
          keyword
    * doc/_source/_static/fits_viewer.png: added
    * vdat/config/tasks.yml: add the header_keys keyword
    * vdat/gui/tabs/ifu_viewer.py: remove line with file name

016-12-14 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: finish the tabs documentation
    * doc/_source/gui/ifu_viewer.rst: add basic info
    * doc/_source/gui/menu_bar.rst: add references
        * doc/_source/_static/vdat_main_panel.png: add
        * doc/_source/_static/vdat_screenshot.png: update
```

```
2016-12-13 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: document the reconstruct tab type
    * doc/_source/_static/fplane_reconstruct.png: add screenshot

2016-12-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: create unique file names using md5 hash.
          Fixes issue #1714

2016-12-12 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/tabs/ifu_widget.py: pass title and tooltip to fits window;␣
↪fixes
          issue #1715
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes

2016-12-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add ability to pass custom tab titles and
          tooltip
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test it

2016-12-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: unset basenames
    * vdat/gui/tabs/tab_widget.py: pass basenames to the ifus
    * vdat/gui/tabs/entry_points.py: add the "reconstruct" tab type
    * setup.py: add the entry point
    * vdat/config/tasks.yml: use it
    * doc/_source/gui/main_panel.rst: add the entry point in the documentation
    * tests/test_gui/test_fplane.py: test the new entrypoint
    * tests/test_gui/test_tabs/test_tab_widget.py: test the changes

2016-12-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: add basenames property to allow loop over
          them to combine multiple exposures
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes

2016-12-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: rename combined to fits_combined
    * setup.py: same
    * tests/test_gui/test_fplane.py: same
    * doc/_source/gui/main_panel.rst: add the fits_combined documentation
    * doc/_source/_static/fplane_fits_combined.png: add image
    * vdat/config/tasks.yml: re-add all the cal steps

2016-12-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_central.py: re-enable the empty fplane tab type for
          testing only
    * tests/test_gui/conftest.py: move some fixture here
```

```
    * tests/test_gui/test_fplane.py: modify accordingly
    * vdat/gui/fplane.py: change pkg_resources import to allow mocking
    * vdat/gui/tabs/entry_points.py: make the empty_fplane private

2016-12-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: implement the combined tab type
    * setup.py: add its entry point, remove old tab entry points
    * tests/test_gui/test_fplane.py: test the new entry points, remove old␣
↪entry
          point tests
    * doc/_source/gui/main_panel.rst: add the entry point docstring

2016-12-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_queue.py: disable some more test for python2
    * vdat/config/tasks.yml: add the first three steps of the zro reduction␣
↪with
          the new tab system

2016-12-05 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: add documentation of the exp_combined␣
↪tab
          type
    * doc/_source/_static/fplane_exp_combined.png: added

2016-12-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: add exp_combined entry point␣
↪implementation
    * vdat/gui/tabs/ifu_widget.py: extend the number of ids that is possible␣
↪to
          use the fits file name; fix bug with cleanup in the base class
    * vdat/gui/tabs/interface.py: fix documentation
    * vdat/gui/tabs/tab_widget.py: create the title and, if given, the tool␣
↪tip;
          remove the tooltip on cleanup
    * doc/_source/gui/main_panel.rst: begin the documentation
    * setup.py: add the exp_combined entry point
    * tests/test_gui/test_fplane.py: test the new entry point
    * tests/test_gui/test_tabs/test_tab_widget.py: fix the tests for the title
          and tooltip

2016-12-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: the base class cleanup method set the empty
          image and paint it (issue #1688)
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it

2016-12-01 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * vdat/gui/tabs/tab_widget.py: add the widgets in the setup method only
```

```
    * tests/test_gui/test_tabs/test_tab_widget.py: update tests

2016-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: remove the widgets from the layout when␣
↪doing
          the cleanup and readd them on setup as a workaround for the non-
↪showing
          problem when readded into the fplane
    * tests/test_gui/test_tabs/test_tab_widget.py: update tests
    * tests/test_gui/test_fplane.py: make sure the first widget is selected

2016-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add temporary entry point for the stacked widget
    * vdat/gui/tabs/entry_points.py: add the entry point function for the
          stacked widget
    * vdat/gui/tabs/tab_widget.py: work on the aspect of the stacked widget.␣
↪Use
          a check box in the bar at the bottom of the focal plane
    * vdat/gui/utils.py: move the vertical spacer here
    * tests/test_gui/test_fplane.py: add an integration test
    * tests/test_gui/test_tabs/test_tab_widget.py: test the changes

2016-11-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: disable the switch button if no
          reconstruction object present
    * tests/test_gui/test_tabs/test_tab_widget.py: test it

2016-11-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: fix bug
    * tests/test_gui/test_tabs/test_tab_widget.py: according to tests the␣
↪widget
          works

2016-11-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: a first implementation of issue #1669 is␣
↪done
    * tests/test_gui/test_tabs/test_tab_widget.py: start testing (still all to
          do, though)

2016-11-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: extend clean/update cached thumbnails to␣
↪the
          quick reconstruction widget
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes
    * tests/test_gui/test_queue.py: disable some test for python2

2016-11-25 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/tabs/ifu_widget.py: clean/update cached thumbnails if needed
          (issue #1660)
    * vdat/gui/utils.py: update function
    * tests/test_gui/test_gui_utils.py: update tests
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the new implementation

2016-11-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: make sure that the header keywords are
          correctly reordered (issue #1675)
    * tests/test_gui/test_tabs/test_ifu_viewer.py: add regression test
    * tox.ini: raise the bar for success of the coverage

2016-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: show the reconstructed image in the fits
          viewer window
    * tests/test_gui/test_tabs/test_ifu_widget.py: test some of it

2016-11-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add possibility to show header keywords at
          the beginning
    * vdat/gui/tabs/ifu_widget.py: pass the tab configuration dictionary
    * vdat/gui/tabs/entry_points.py: document
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test the header keyword
          reordering
    * tests/test_gui/test_tabs/test_ifu_widget.py: update tests

2016-11-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: open the fits viewer on double click
    * vdat/gui/tabs/ifu_viewer.py: zoom to fit on first show
    * tests/test_gui/test_tabs/test_ifu_widget.py: add tests

2016-11-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: fix bug introduced with fixing #1646
    * tests/test_gui/test_queue.py: re-enabled some test that would have␣
↪avoided
          this bug

2016-11-17 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: make sure selections are correctly propagated to␣
↪all
          tabs (#1671)
    * vdat/gui/tabs/ifu_widget.py: same
    * vdat/gui/tabs/tab_widget.py: same
    * tests/test_gui/conftest.py: fix fixtures
    * tests/test_gui/test_central.py: test the changes
    * tests/test_gui/test_fplane.py: same

2016-11-16 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/tabs/entry_points.py: add the reconstructed IFU pluging
          (issue #1633)
    * vdat/gui/tabs/ifu_widget.py: implement the IFU widget for it
    * vdat/gui/tabs/tab_widget.py: implement the tab widget for it
    * vdat/gui/utils.py: add the prefix for the reconstructed files
    * setup.py: add the entry point
    * ReleaseNotes.md: update
    * tests/test_gui/conftest.py: new fixture of the reconstructed object
    * tests/test_gui/test_fplane.py: test the new plugin
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the new widget
    * tests/test_gui/test_tabs/test_tab_widget.py: same

2016-11-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: remove all the reconstruction hints from␣
↪the
          FitsFplanePanel class
    * vdat/gui/tabs/ifu_widget.py: store some more info, remove old pieces of
          code
    * vdat/gui/tabs/entry_points.py: cleanup the plugin
    * tests/test_gui/test_tabs/test_tab_widget.py: fix tests accordingly
        * ReleaseNotes.md: updated

2016-11-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: react to changing the scale
    * vdat/gui/tabs/ifu_widget.py: add attributes storing the global scaling␣
↪and
          use them, if available, to paint the fits files
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/test_tabs/test_tab_widget.py: test it

2016-11-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: add zmin/zmax property
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * vdat/gui/tabs/tab_widget.py: get the global zscale for the IFUs and␣
↪store
          it; add needed functionality for this; add cleanup
    * tests/test_gui/test_tabs/test_tab_widget.py: test it

2016-11-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add header information
    * vdat/gui/tabs/ifu_widget.py: ifu_viewer classes name changed
        * ReleaseNotes.md: udpated with issue #1631

2016-11-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: start rewriting the fits viewer. Show the
          files in independent tabs.
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test most of the new
          implementation
```

```
    * tests/test_gui/test_central.py: move fixtures from here ...
    * tests/test_gui/conftest.py: ... to here

2016-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: recreate thumbnail if it's older than the
          fits file; clear cached thumbnail if the fits file does not exist.
    * tests/test_gui/test_tabs/test_ifu_widget.py: test this

2016-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: move the thumbnailing code here
    * tests/test_gui/test_gui_utils.py: test it
    * vdat/gui/tabs/ifu_widget.py: implement prepare_image to produce␣
↪thumbnails
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/conftest.py: add fixture

2016-11-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add base class for the IFU viewers
    * doc/_source/codedoc/gui/tabs/ifu_viewer.rst: add inheritance tree

2016-11-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: update documentation; get type from
          database
    * vdat/gui/tabs/tab_widget.py: add setup method
    * vdat/gui/tabs/ifu_widget.py: add and implement setup method
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/test_fplane.py: add fits_fplane integration test

2016-11-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: improve error messages at plugin loading time (issue
          #1655)
    * tests/test_gui/test_fplane.py: update tests

2016-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: add extra "step_name" parameter to pass to the␣
↪plugins
    * vdat/gui/tabs/entry_points.py: same
    * vdat/gui/tabs/interface.py: same
    * vdat/gui/tabs/tab_widget.py: begin implementing the setup
    * vdat/gui/tabs/ifu_widget.py: same

2016-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: create fits_fplane entry point
    * setup.py: same
    * vdat/gui/tabs/tab_widget.py: add scale buttons to the GUI
    * vdat/gui/tabs/ifu_widget.py: remove unused method
    * tests/test_gui/test_tabs/test_tab_widget.py: test new class init
```

```
2016-10-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: don't trigger single clicks on double click
          (issue #1639)
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: add some info about
          this
    * ReleaseNotes.md: updated with the issue reference

2016-10-25 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: updated
    * vdat/gui/queue.py: remove the command and thread when the command␣
↪finishes
          to run
    * vdat/gui/tabs/tab_widget.py: set parent in the thread

2016-10-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: don't draw the tab
    * vdat/gui/tabs/ifu_widget.py: fix hw_size, cleanup code
    * vdat/gui/tabs/tab_widget.py: use showEvent, properly select/unselect IFU
          to make it reply back
    * tests/test_gui/test_fplane.py: add regression test to make sure that
          selecting/deselecting all IFUs work as expected
    * tests/test_gui/test_tabs/test_ifu_widget.py: update tests
    * tests/test_gui/test_tabs/test_tab_widget.py: same
    * tests/test_gui/test_mainwindow.py: fix weird interaction with other␣
↪tests
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: update with the
          showEvent info
    * doc/_source/gui/main_panel.rst: add link to the above document

2016-10-21 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/codedoc/gui/tabs/entry_points.rst: added
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: add an example on how
          to use and extend the code
    * doc/_source/codedoc/gui/index.rst: add them to index
    * vdat/gui/tabs/entry_points.py: set parent explicitly
    * vdat/gui/tabs/ifu_widget.py: fix documentation

2016-10-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: add empty_fplane tab type using
          BaseFplanePanel
    * vdat/gui/tabs/interface.py: fix bug with error message
    * doc/_source/gui/main_panel.rst: add empty_fplane documentation
    * setup.py: add empty_fplane entry points
    * tests/test_gui/test_fplane.py: test that the new entry point behave as
          expected

2016-10-20 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/tabs/tab_widget.py: fix bugs
    * vdat/gui/tabs/ifu_widget.py: fix cleanup documentation
    * tests/test_gui/conftest.py: update fixture for better use
    * tests/test_gui/test_central.py: use the new fixtures
    * tests/test_gui/test_tabs/test_tab_widget.py: test BaseFplanePanel

2016-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: implement the BaseFplanePanel widget,␣
↪rename
          some of the UpdateIFUTask parts
    * vdat/gui/tabs/ifu_widget.py: add TODO on a slot
    * vdat/gui/tabs/interface.py: update documentation
    * doc/_source/codedoc/gui/tabs/tab_widget.rst: show full inheritance
        * tests/test_gui/test_tabs/test_tab_widget.py: update the␣
↪UpdateIFUTask
          names

2016-10-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: update the UpdateIFUTask to run
    * tests/test_gui/test_tabs/test_tab_widget.py: test it
    * vdat/gui/tabs/ifu_widget.py: update documentation

2016-10-18 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_central.py: move a fixture to the conftest
    * tests/test_gui/conftest.py: update above fixture
    * tests/test_gui/test_tabs/test_ifu_widget.py: added, test the␣
↪BaseIFUWidget
    * vdat/gui/tabs/ifu_widget.py: fix renaming bugs

2016-10-17 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/codedoc/gui/tabs/ifu_widget.rst: add inheritance map
    * vdat/gui/tabs/ifu_widget.py: first iteration of the IFU widget base␣
↪class
    * vdat/gui/tabs/tab_widget.py: fix names according to the above changes

2016-10-14 Francesco Montesano <montefra@mpe.mpg.de>


        * doc/_source/codedoc/gui/index.rst: use the new tabs/* files
    * doc/_source/codedoc/gui/tabs: created
    * doc/_source/codedoc/gui/tabs/ifu_viewer.rst: same
        * doc/_source/codedoc/gui/tabs/ifu_widget.rst: renamed from
          ../ifu_widget.rst
        * doc/_source/codedoc/gui/tabs/interface.rst: renambed from tabs_
↪plugins
    * doc/_source/codedoc/gui/tabs/tab_widget.rst: created
    * doc/_source/conf.py: add inheritance diagram
    * vdat/gui/fplane.py: remove all the fplane panel implementation
    * vdat/gui/tabs/tab_widget.py: move here the fplane panel code
```

```
    * vdat/gui/ifu_viewer.py: move to tabs/ifu_viewer.py
    * vdat/gui/ifu_widget.py: move to tabs/ifu_widget.py

2016-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/index.rst: fix typo
    * doc/_source/gui/main_panel.rst: update the documentation
    * vdat/gui/tasks.py: PEP8 fix
        * doc/_source/conf.py: fix intersphinx link
        * ReleaseNotes.md: update

2016-09-30 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/fplane.py: remove yield_*_ifu methods and get rid of the
          VDATRunControl object
    * vdat/gui/treeview_model.py: don't use it

2016-09-30 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_fplane.py: test the FplaneWidget
    * vdat/gui/fplane.py: fix some bug and adjust edge cases

2016-09-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: Implement the plugin system for the tabs
    * vdat/gui/tabs/interface.py: adjust interfaces
    * doc/_source/codedoc/gui/fplane.rst: add the FplaneWidget to the docs
    * tox.ini: don't run the doc-qt if the qt* executables do not exist

2016-09-27 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * vdat/gui/tabs/interface.py: add the interface for the function
          implementing the plugin
    * doc/_source/gui/main_panel.rst: add a bit of documentation

2016-09-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/__init__.py: added
    * vdat/gui/tabs/interface.py: added; implement the FplaneTabTemplate class
          (issue #1601)
    * doc/_source/codedoc/gui/tabs_plugins.rst: added
    * doc/_source/codedoc/gui/index.rst: add tabs_plugins to the index
    * doc/_source/gui/main_panel.rst: add the FplaneTabTemplate to the
          documentation

2016-09-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: reimplement the FplaneCache as a pure cache without
          much intelligence
    * tests/test_gui/test_fplane.py: test it
    * doc/_source/gui/main_panel.rst: begin documenting the plugin system
    * doc/_source/codedoc/gui/fplane.rst: change the documentation to better
          separate parts
```

```
2016-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: create here cache and call the reconstruction object
          in the appropriate place; fix super calls
    * vdat/gui/central.py: move the cache creation in the fplane module
    * tests/test_gui/test_central.py

2016-08-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: don't pass a reconstruct object to the FplaneCache
          object. Add property that returns it using the function create in␣
↪r440
    * vdat/gui/central.py: remove the reconstruction object creation and adapt
          the code
    * tests/test_gui/test_central.py: remove tests about the reconstruction
          object
    * ReleaseNotes.md: update

2016-08-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add function that creates the QuickReconstruct object
          only once and returns it
    * tests/test_gui/test_gui_utils.py: test it
    * tests/conftest.py: add fixture to deal with it
    * tests/test_gui/conftest.py: move around some fixture
    * tests/test_gui/test_central.py: same and adapt

2016-08-18 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk into ^/branches/plug_tabs_1533

2016-08-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menubar.py: emit a signal instead of closing the VDAT (issue
          #1568)
    * vdat/gui/mainwindow.py: connect the above signal with the mainwindow␣
↪close
          slot; isolate connections with the menu bar into a method
    * tests/test_gui/test_mainwindow.py: test closing via the above signal
        * ReleaseNotes.md: update

2016-08-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/help_window.py: add support for mailto (issue #1579)
    * tests/test_gui/test_help_window.py: test it
    * setup.py: update dependencies to coverage>=4.2
    * tox.ini: same, remove obsolete cov-init environment

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/splash_screen_1562 into ^/trunk

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        * : merge ^/trunk into ^/branches/splash_screen_1562
        * vdat/gui/static/VirusDataAnalysisTool*: updated and renamed

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: remove always on top flag, make surer that the
          splashscreen is shown
    * tests/test_gui/test_init.py: update the tests

2016-08-01 Francesco Montesano <montefra@mpe.mpg.de>

    * AUTHORS: Majo added
    * ReleaseNotes.md: updated
    * doc/_source/_static/vdat_icon.ico: added
    * doc/_source/_static/vdat_name.jpg: added
    * doc/_source/_static/vdat_screenshot.png: updated
    * doc/_source/command_interpreter.rst: fix typo
    * doc/_source/conf.py: add the VDAT logo and name
    * setup.py: fix ginga and pytest-qt to older version
    * tox.ini: same
    * tests/test_gui/test_init.py: added
    * vdat/gui/__init__.py: work around splash screen not showing

2016-08-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: do all the splash screen things here
    * vdat/gui/mainwindow.py: add window icon
    * vdat/gui/static/HETDEX_*.jpg: new images and renamed to VDAT_*.jpg
    * vdat/gui/static/vdat_icon.jpg: added and used as icon
    * vdat/libvdat/vdat.py: just create and show the splash screen

2016-07-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: add splash screen functionality
    * vdat/gui/static/HETDEX_black.jpg: added
    * vdat/gui/static/HETDEX_white.jpg: same
    * vdat/libvdat/vdat.py: show the splash screen

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/qt_help_1454/ into ^/trunk

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>


    * ReleaseNotes.md: update
    * vdat/gui/menubar.py: add about Qt button
    * doc/_source/_static/menubar_help.png: update accordingly
    * doc/_source/gui/menu_bar.rst: same
        * tests/test_gui/test_menubar.py: same
    * vdat/gui/static/VDATVirusDataAnalysisTool.*: add updated documentation

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/mainwindow.py: remove the logging gui handlers when closing␣
↪down
    * vdat/database/core.py: fix connect context manager for in memory␣
↪database
    * tests/test_database.py: test it
    * tests/test_gui/test_mainwindow.py: add and test the whole mainwindow
    * tests/test_gui/test_help_window.py: make the monkeypatch of static␣
↪method
          work on py2
    * tests/test_gui/test_queue.py: temporarily skip some of the offending␣
↪tests

2016-08-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/help_window.py: don't run setup in the __init__ for better␣
↪test.
          Setup the data only after connecting the slot to setup the content␣
↪widget
    * vdat/gui/menubar.py: set help menu object name for easier testing
    * vdat/gui/utils.py: don't cover run method
    * tests/test_gui/test_gui_utils.py: add test for static directory and␣
↪qthelp
          file
    * tests/test_gui/test_help_window.py: added
    * tests/test_gui/test_menubar.py: add test for when no documentation is
          found

2016-08-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/help_window.py: added to address issue #1454
    * vdat/gui/mainwindow.py: add slot to open the help window
    * vdat/gui/menubar.py: add menu item to open the help window
    * vdat/gui/static/VDATVirusDataAnalysisTool.qch: add documentation
    * vdat/gui/static/VDATVirusDataAnalysisTool.qhc: same
    * vdat/gui/utils.py: add functions to get the documentation file
    * doc/Makefile: make qthelp now also creates the qch and qhc files
    * doc/_source/codedoc/gui/help_window.rst: added
    * doc/_source/codedoc/gui/index.rst: add
    * doc/_source/contributions.rst: add information about the qt␣
↪documentation
    * tox.ini: add you environment to create the qt documentation

2016-08-02 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/contributions.rst: update information (issue #1529)

2016-08-02 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/db_version_cleanup into ^/trunk

2016-07-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_widget.py: convert property to pyqtProperty (issue #1532)
```

```
    * vdat/gui/menus_actions.py: same
    * vdat/gui/treeview_model.py: same
    * tests/test_libvdat/test_symlink.py: add more randomization to decrease␣
↪the
          possible file name clashes

2016-08-02 Francesco Montesano <montefra@mpe.mpg.de>

    * pytest.ini: set qt_api=pyqt4
    * setup.py: set pytest-qt>=0.2, skip broken version of ginga
    * tests/test_database.py: fix test
    * tests/test_gui/test_progress.py: same
    * tox.ini: fix pytest-qt version and remove qt_api

2016-07-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/models.py: update VDATExposures according to issue #1530
    * vdat/database/check.py: update the conversion function
    * vdat/libvdat/symlink.py: update the creation of the exposures entries
    * tests/conftest.py: update the current exps dictionary
    * tests/test_database.py: adapt tests
    * tests/test_gui/test_menus_actions.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_symlink.py: same
    * ReleaseNotes.md: update with info about the issue

2016-07-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/models.py: remove virtual and ifus fields as per issue
↪#1339
    * vdat/database/check.py: update the conversion function
    * tests/conftest.py: update the current shot, move here db initialization
          routines
    * tests/test_database.py: adapt the tests
    * tests/test_gui/test_central.py: same
    * ReleaseNotes.md: update with info about the issue

2016-07-01 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/db_version_cleanup

2016-07-01 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * tests/test_database.py: make sure that metadata are moved

2016-07-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/check.py: when updating the meta data, remove superfluous
          entries
    * vdat/libvdat/symlink.py: improve error messages when inserting existing
          meta data
    * tests/test_database.py: test the changes
    * tests/test_libvdat/test_symlink.py: same
```

```
    * tests/conftest.py: shuffle fixtures around

2016-06-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/launching.rst: add some info about vdat_db command
    * vdat/database/check.py: after updating the metadata try to add it to the
          database
    * tests/test_database.py: test it
    * doc/_source/codedoc/utilities.rst: added
    * doc/_source/codedoc/index.rst: added to the index

2016-06-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/check.py: add possibility to repair the database
    * tests/test_database.py: test it

2016-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/check.py: make vdat_db check subcommand
    * tests/test_database.py: test the changes

2016-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add enum34 for python < 3.4 and colorama; add vdat_db entry␣
↪point
    * vdat/database/check.py: added, implement db checks
    * vdat/database/core.py: allow to create db in memory (and don't close it)
    * vdat/utilities.py: function that reads the shot and exps files can also
          return the dir containing the files
    * tests/test_database.py: test the new check module
    * tests/test_utilities.py: update tests

2016-06-17 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: move the function to scan for metadata files to
          vdat/utilities.py and adapt
    * vdat/utilities.py: update the above function implementation
    * tests/test_libvdat/test_symlink.py: adapt the tests
    * tests/test_utilities.py: same

2016-06-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/base.py: add database exception class
    * vdat/database/models.py:
    * vdat/database/__init__.py: make merge_entries more robust against type
          mismatches
    * tests/test_database.py: test the database subpackage
    * tests/test_database: removed

2016-06-30 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/config_version into ^/trunk

2016-06-30 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        * : merge ^/trunk into ^/branches/config_version

2016-06-30 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: fix typo
    * vdat/config/entry_point.py: don't add common parser to the parent one

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/codedoc/database.rst: show class inheritance
    * vdat/database/models.py: add version as a table column with a check
    * vdat/database/__init__.py: explicitly import stuff at the package level
    * vdat/database/core.py: remove __all__
    * vdat/libvdat/symlink.py: add the table versions
    * tests/test_gui/test_menubar.py: same
    * tests/test_gui/test_menus_actions.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_symlink.py: same
    * tests/test_database: added for the future
        * ReleaseNotes.md: updated

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update with info about issue #1501

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: disable the reconstruction checkbox and tab if the
          reconstruction object does not exist
    * vdat/gui/central.py: remove unnecessary method

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_central.py: finish testing the central module
    * vdat/gui/mainwindow.py: fix documentation typo

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: modify imports to allow monkeypatching for the␣
↪tests;
    * tests/test_gui/test_central.py: add first part of the central module␣
↪tests

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: don't fail if the quick reconstruction object
          creation fails; add documentation
    * vdat/config/entry_point.py: fix typo
    * doc/_source/codedoc/gui/central.rst: added
    * doc/_source/codedoc/gui/tasks.rst: added
    * doc/_source/codedoc/gui/index.rst: add them to the index

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * ReleaseNotes.md
    * vdat/config/entry_point.py: add backup option to the vdat_config copy
          command
    * doc/_source/launching.rst: document it
    * tests/test_config/test_entry_point.py: test it

2016-06-13 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/launching.rst: add the compare command to the doc
    * vdat/config/entry_point.py: fix one of the info messages

2016-06-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/versions.py: add the remaining loaders
    * tests/test_config/test_versions.py: test them

2016-06-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/versions.py: add some loader to check if files are␣
↪reasonable
    * vdat/config/entry_point.py: add ``-l/--load`` option
    * tests/test_config/test_entry_point.py: test the above
    * tests/test_config/test_versions.py: same

2016-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: some improvement on the messages
    * tests/test_config/test_entry_point.py: update the tests

2016-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: add diff of files
    * tests/test_config/test_entry_point.py: test it

2016-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add colorama for nice coloured output
    * vdat/config/entry_point.py: add a compare subcommand and check if files
          exist and have the correct version
    * vdat/config/versions.py: add helper functions
    * tests/test_config/test_entry_point.py: test it
    * tests/test_config/test_versions.py: test it

2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/__init__.py: explicitly import attributes
    * vdat/config/core.py: remove __all__
    * vdat/config/versions.py: explicit file names to copy and file versions
    * vdat/config/entry_point.py: rewrite the copy function to use the list of
          names and to replace version numbers
    * vdat/config/tasks.yml: add version
    * vdat/config/vdat_commands.yml: add version
    * vdat/config/vdat_setting.cfg: add version
```

```
    * tests/test_config/test_entry_point.py: update the tests
    * tests/test_config/test_versions.py: add
    * doc/_source/codedoc/config.rst: add the versions module

2016-06-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: improved the documentation␣
↪(hopefully)
    * vdat/gui/ifu_viewer.py: Fixed a small bug created by a previous rename
      of the ihmp attribute in the IFU object to ifuslot.

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/log_sig_1492 into ^/trunk

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/trunk into ^/branches/log_sig_1492

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py: use the command_intepreter command_logger into␣
↪the
          QCommandInterpreter, connecting/disconnecting the default helper␣
↪slot in
          the ``run`` method
    * tests/test_gui/test_queue.py: PEP8
    * tests/test_libvdat/test_symlink.py: remove unused import
    * ReleaseNotes.md: update release notes

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/clone_remove into ^/trunk
    * ReleaseNotes.md: update

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/trunk into ^/branches/clone_remove
    * ReleaseNotes.md: update

2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: re-emit the sig_exposure_removed signal in␣
↪the
          ReductionQTreeView; return the menu created at right-click
    * tests/test_gui/test_tree_view.py: properly test the menu and that the
          signal is re-emitted

2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: add sig_exposure_removed signal to address
        issue #1498
    * tests/test_gui/test_menus_actions.py: test the emission
```

```
2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/tree_view.rst: add Remove exposure description
    * doc/_source/_static/tree_view_clone_remove.png: update screenshot

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: clear the menu before adding new actions
    * tests/test_gui/test_menus_actions.py: test the menus_actions module
    * doc/_source/codedoc/gui/menus_actions.rst: added
    * doc/_source/codedoc/gui/index.rst: add to the index

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: implement a dynamic menu to remove exposures
    * vdat/gui/treeview_model.py: use it

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: clone in a thread and wait for it to finish;
          set the cursor to wait cursor when cloning or removing a directory
    * tests/test_gui/test_tree_view.py: adapt to the changes and test that
          cloning within or without the thread leads to the same result

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: explicitly create actions to clone/remove␣
↪and
          connect them to slots
    * tests/test_gui/test_tree_view.py: update the tests
    * doc/_source/_static/tree_view_clone_remove.png: update image

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: create signal to perform the␣
↪command
          logger
    * vdat/command_interpreter/core.py: use it to communicate the
          success/failure of the single sub-commands
    * vdat/command_interpreter/helpers.py: add default implementation for a␣
↪slot
          for the above signal
    * tests/test_ci/conftest.py: adapt the tests to the changes
    * tests/test_ci/test_command_interpreter.py: same
    * tests/test_ci/test_helpers.py: same
    * tests/test_ci/test_signals.py: same

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/changelog.rst: include changelog verbatim
    * setup.cfg: move coverage configuration here
    * .coveragerc: remove

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * setup.py: bump the version and the required pyhetdex version
    * ReleaseNotes.md: update

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_progress.py: fix failing test

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: updated to the newest format and values

2016-06-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Updated to new fplane format
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/central.py: Same

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menubar.py: add possibility to specialize removal with the type
    * vdat/config/vdat_setting.cfg: add documentation and example on how to do
          this
    * doc/_source/gui/menu_bar.rst: document it
    * tests/test_gui/test_menubar.py: test it

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: fix typo
    * vdat/gui/utils.py: add function to remove files
    * vdat/gui/menubar.py: sig_remove_files are emitted with the error and␣
↪thumb
          files
    * vdat/gui/mainwindow.py: connect the new menu bar signal with a slot and
          document the class
    * tests/test_gui/test_gui_utils.py: test the removal
    * tests/test_gui/test_menubar.py: adapt the tests
    * doc/_source/codedoc/gui/fplane.rst: added
    * doc/_source/codedoc/gui/ifu_widget.rst: added
    * doc/_source/codedoc/gui/index.rst: added
    * doc/_source/codedoc/gui/mainwidget.rst: added
    * doc/_source/codedoc/gui/mainwindow.rst: added
    * doc/_source/gui/menu_bar.rst: update the documentation

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/install.rst: fix documentation warning
        * tests/test_ci/test_ci_utils.py: renamed from test_utils.py to avoid␣
↪pytest
          error
        * tests/test_gui/test_gui_utils.py: renamed from test_utils.py to␣
↪avoid
          pytest error
```

```
2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add a custom QThread that executes one function
    * tests/test_gui/test_utils.py: test the utilities
    * doc/_source/codedoc/gui/index.rst: add the utilities to the␣
↪documentation
    * doc/_source/codedoc/gui/utils.rst: same

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/menubar_file.png: added
    * doc/_source/_static/menubar_help.png: added
    * doc/_source/_static/menubar_select.png: added
    * doc/_source/_static/menubar_view.png: added
    * doc/_source/codedoc/gui/menubar.rst: added
    * doc/_source/codedoc/gui/index.rst: add the above
    * doc/_source/gui/menu_bar.rst: document behaviour with links
    * doc/_source/install.rst: fix link
    * setup.py: bump version
    * tests/test_gui/test_menubar.py: test the menubar module
    * vdat/config/vdat_setting.cfg: add option with name of the files to␣
↪remove
    * vdat/gui/menubar.py: add the delete file actions
    * vdat/gui/utils.py: add the thumbnails directory name

2016-06-03 Danuel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/progress.py: Changed "Running" to "Finished" in the status bar,
          to better reflect the actual state of the job.

    Also changed the wording in some of the documentation:
    * doc/_source/command_interpreter.rst
    * doc/_source/dirstruct.rst
    * doc/_source/gui/index.rst
    * doc/_source/gui/main_panel.rst
    * source/gui/menu_bar.rst
    * doc/_source/gui/progress.rst
    * doc/_source/gui/queue.rst
    * doc/_source/gui/tree_view.rst
    * doc/_source/install.rst
    * doc/_source/launching.rst

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.4.0

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: add hints to use virus_config files
    * vdat/config/tasks.yml: update the commands

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: updated with the latest values
```

```
2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/faq.rst: added
    * doc/_source/index.rst: add it to the index
    * vdat/libvdat/symlink.py: fix typos in error messages

2016-05-31 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * doc/_source/install.rst: solve #1452
    * doc/_source/contributions.rst: update

2016-05-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: add creation of mastertwi or masterflat
          from twilight frames; remove fake pixel flat
    * vdat/config/tasks.yml: add the above commands
    * vdat/libvdat/symlink.py: improve error message
    * vdat/config/entry_point.py: add space when asking to the user
    * tests/test_libvdat/test_loggers.py: adapt the tests to the removal of␣
↪the
          fake pixel flat

2016-05-31 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/symlink.py: add customisation of the types to use for
          calibration
    * vdat/config/vdat_setting.cfg: add/modify the configuration options to
          support the changes above
    * vdat/libvdat/vdat.py: add command line overrides for the above changes
    * tests/test_libvdat/test_symlink.py: test combination of calibration␣
↪types;
          get rid of virus00001 fixture
    * doc/_source/dirstruct.rst: document the changes

2016-05-30 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/launching.rst: update
    * doc/_source/dirstruct.rst: update
    * vdat/libvdat/vdat.py: update command line option name

2016-05-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/executables.rst: removed
    * doc/_source/index.rst: adapted
    * doc/_source/launching.rst: start improving the use information

2016-05-25  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Fix global cutlevel estimation fixing issue1482
        * vdat/gui/ifu_widget.py: Fix bad levels in thumbnails

2016-05-25 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/command_interpreter/core.py: don't emit the number of primaries if
          it's zero
    * vdat/gui/queue.py: properly emit finished when CommandInterpreter.run
          returns
    * tests/test_ci/test_command_interpreter.py: adapt test

2016-05-25 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/vdat.py: skip interpolation error when setting CUREBIN␣
↪in the
      environment
    * tests/test_libvdat/test_vdat.py: add tests for this

2016-05-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/progress.py: renamed from vdat/gui/progress_bar.py; move
          custom status bar here
    * vdat/gui/mainwidget.py: adapt
    * vdat/gui/mainwindow.py: adapt
    * tests/test_gui/test_progress.py: renamed from test_progress_bar.py;␣
↪adapt
          to new name; add status bar tests
    * doc/_source/_static/progress_done.png: added
    * doc/_source/_static/progress_going.png: added
    * doc/_source/codedoc/gui/progress.rst: added
    * doc/_source/codedoc/gui/index.rst: adapt and cleaup
    * doc/_source/gui/progress.rst: add progress and status bar info
    * doc/_source/gui/progress_bar.rst: renamed progress.rst
    * doc/_source/gui/index.rst: adapted

2016-05-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: fix the documentation; make all slots public
    * vdat/gui/treeview_model.py: fixed typos
    * doc/_source/gui/queue.rst: user documentation added
    * doc/_source/_static/queue_screenshot.png: add screenshot
    * doc/_source/_static/queue_tooltip.png: same
    * doc/_source/codedoc/gui/queue.rst: added
    * doc/_source/codedoc/gui/index.rst: add queue to the index
    * doc/_source/codedoc/gui/tree_view.rst: fix title

2016-05-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/progress_bar.py: remove command interpreter signals and mark
          functions as slots
    * vdat/gui/mainwidget.py: re-enable progress bar
    * vdat/gui/mainwindow.py: connect the queue signals to the progress bars
          and the status bar; the isolate the status bar in its own class
    * tests/test_gui/test_progress_bar.py: rewrite the tests

2016-05-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move CommandInterpreter._run into a
```

```
        standalone function to avoid pickling the instance when running in
        multiprocessor mode; get all the signals and attach them to␣
↪attributes,
        then use the attributes in the CommandInterpreter.run method
    * tests/test_ci/test_command_interpreter.py: adapt the tests
    * vdat/gui/queue.py: replace the VDATCommandWoker with a␣
↪QCommandInterpreter
        class, that create a QObject out of the command_interpreter and␣
↪implement
        the signals as pyqtSignals; adapt other parts of the code to those␣
↪changes
    * tests/test_gui/test_queue.py: adapt the tests
    * vdat/gui/__init__.py: don't connect to CommandInterpreter signals
    * vdat/gui/central.py: use QCommandInterpreter
    * vdat/gui/mainwindow.py: add a function to connect the signals emitted by
        the queue
    * vdat/libvdat/vdat.py: don't connect to the CommandInterpreter signals

2016-05-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move all the signal emissions to the
        main process
    * vdat/gui/queue.py: re-emit worker signals in the Queue class; mark old
        worker and thread for deletion on the next command execution
    * tests/test_gui/test_queue.py: adapt the tests

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: connect command interpreter signals with pyqt ones
    * vdat/gui/__init__.py: cleanup imports
    * vdat/command_interpreter/signals.py: fix documentation bugs
    * tests/test_gui/test_queue.py: test the new signals

2016-05-17 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/relay.py: removed
    * vdat/gui/mainwindow.py: set the queue here, so that it can be used when
        creating the queue action; remove all relay.py references
    * vdat/gui/__init__.py: remove the setting of the queue
    * vdat/gui/queue.py: remove relay and use already existing signals
    * tests/conftest.py: remove relays; normalize clear_* fixtures
    * tests/test_ci/conftest.py: use clear_* fixtures
    * tests/test_ci/test_signals.py: use clear_* fixtures
    * tests/test_gui/test_progress_bar.py: use clear_* fixtures
    * tests/test_gui/test_queue.py: finish testing the queue module

2016-05-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: use a thread-worker approach
    * tests/test_gui/test_queue.py: test 89% of the queue module
    * doc/_source/codedoc/gui/index.rst: remove the background

2016-05-24 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        * vdat/gui/treeview_model.py: improve documentation and rename _
↪pressed slot
          to on_press
    * doc/_source/codedoc/database.rst: added
    * doc/_source/codedoc/gui/index.rst: modified
    * doc/_source/codedoc/gui/tree_view.rst: added
    * doc/_source/codedoc/index.rst: add database
    * doc/_source/conf.py: add pyqt4 intersphinx (but it apparently doesn't
          work)
    * doc/_source/launching.rst: fix link

2016-05-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: add cal_dir and zero_dir to the tooltip
        * tests/test_gui/test_tree_view.py: add it to the test
    * doc/_source/_static/tree_view_tooltip.png: update the screenshot
    * doc/_source/gui/tree_view.rst: finish the documentation

2016-05-21 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/tree_view_clone_dialog.png: add screenshot
    * doc/_source/_static/tree_view_clone_remove.png: add screenshot
    * doc/_source/_static/tree_view_remove_dialog.png: add screenshot
    * doc/_source/_static/tree_view_screenshot.png: add screenshot
    * doc/_source/_static/tree_view_tooltip.png: add screenshot
    * doc/_source/dirstruct.rst: added todo
    * doc/_source/gui/index.rst: add information
    * doc/_source/gui/menu_bar.rst: little change
    * doc/_source/gui/tree_view.rst: begin improving the documentation

2016-05-19 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/vdat_screenshot.png: added
    * doc/_source/gui: all the gui related documentation moved here
    * doc/_source/gui/index.rst: short description and link to more detailed
          ones
    * doc/_source/gui/ifu_viewer.rst: added
    * doc/_source/gui/log_panel.rst: added
    * doc/_source/gui/main_panel.rst: added, all info about the main panel␣
↪moved
          here
    * doc/_source/gui/menu_bar.rst: added
    * doc/_source/gui/progress_bar.rst: added
    * doc/_source/gui/queue.rst: moved here
    * doc/_source/gui/tree_view.rst: added, all info about the tree view moved
          here
    * doc/_source/index.rst: updated to point to gui/index

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: show the right menu entry only when clicking
          on a valid item; resolves #1465
    * tests/test_gui/test_tree_view.py: test ReductionQTreeView.option_menu␣
↪slot
```

```
2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: if no night found, return a model with only
          the root directory
    * tests/test_gui/test_tree_view.py: adapt tests and test the above case

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: move the creation of the model and the
          initialization of the view into the ReductionQTreeView class
        * vdat/gui/mainwidget.py: use the ``ReductionQTreeView.set_model``␣
↪method to
          redo the symlink
    * tests/test_gui/test_tree_view.py: adapt the tests

2016-05-20  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/menubar.py: Added Quit menu, renamed symlink to file menu

2016-05-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: fix #1448

2016-05-11  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Reuse the updateTask, removing one of the
        fast click crash bugs

2016-05-11  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed bug occuring when switching
        fplanes fast

2016-05-11 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * vdat/gui/background.py: remove
    * tests/conftest.py: adapt
    * vdat/gui/__init__.py: same
    * vdat/gui/queue.py: cleanup
    * tests/test_gui/test_queue.py: added

2016-05-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: don't use background
    * vdat/gui/background.py: start phasing this out
    * vdat/gui/queue.py: add a run method and a first implementation of the
          thread to run the command

2016-05-09 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/gui/background.py: remove unused immediate thread
     * vdat/gui/mainwindow.py: same
```

```
2016-05-11 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.4.0rc3
    * vdat/libvdat/vdat.py: fix command line interface
    * doc/_source/launching.rst: fix documentation of the above

2016-05-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: update the fplane file with the 9 IFUs
    * vdat/config/tasks.yml: update
    * vdat/config/vdat_commands.yml: update
    * vdat/gui/ifu_widget.py: add SPECID to the tooltip

2016-05-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwidget.py: fix symlinking in gui
    * vdat/gui/utils.py: add wait cursor context manager

2016-05-09  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Made updates more robust
        * vdat/gui/ifu_viewer.py: Add warning if no images selected to
        send to ds9
        * vdat/gui/ifu_widget.py: Make updates more robust

2016-05-09  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Only update ifus during cleanup, if they
        were initialized before

2016-05-06 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * vdat/config/vdat_commands.yml: add arguments to subtract sky
    * vdat/config/tasks.yml: same
    * vdat/gui/__init__.py: read progress report to console
    * vdat/gui/mainwidget.py: disable progress bar

2016-05-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: add virus instrument to the argument parser

2016-05-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove all astropy loggers
    * vdat/libvdat/loggers.py: explicitly pass configuration objects
    * vdat/libvdat/vdat.py: adapt
    * tests/test_libvdat/test_loggers.py: moved into test_libvdat;
          vdat.libvdat.loggers tested at 100%

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/gui/__init__.py: disconnect the indicator printing progress to␣
```

```
↪console
     * vdat/gui/progress_bar.py: set value to zero when setting it up
     * tests/test_gui/test_progress_bar.py: test the latter

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: make sure that x11 can deal with threads

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add n_primaries signal to
          documentation
    * vdat/gui/progress_bar.py: implement the progress bar and connect to
          command_interpreter signals
    * vdat/gui/mainwidget.py: use the object in progress_bar.py module
    * tests/test_ci/conftest.py: move clean_connected to test/conftest.py
    * tests/test_gui: created
    * tests/test_gui/test_progress_bar.py: added
    * tests/test_gui/test_tree_view.py: moved into test_gui
    * tests/test_ci/test_signals.py: make sure to clean the signals

2016-05-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: new n_primaries signal
    * vdat/command_interpreter/helpers.py: helper function for that
    * vdat/command_interpreter/core.py: emit the signal
    * tests/test_ci/test_helpers.py: update tests
    * tests/test_ci/test_signals.py: same

2016-05-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/extra_files/lines_000[LR].dat: removed
    * vdat/config/extra_files/lines_[LR].par: copied from cure
    * vdat/config/vdat_commands.yml: use combinearcs to create masterarcs; use
          the new line files in deformer, fix option
    * vdat/config/tasks.yml: adapt to the above changes; now I can run␣
↪deformer
          on most IFUs

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: update the configuration documentation
    * vdat/config/tasks.yml: add all the remaining reduction steps
    * vdat/config/vdat_setting.cfg: set the default pixel scale to 0.5, to␣
↪speed
          up the GUI startup
    * vdat/gui/fplane.py: don't raise an error if there are no tabs for a task

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwidget.py: cleanup old ways of communication
    * vdat/gui/mainwindow.py: same
    * vdat/gui/relay.py: remove unused signals
    * vdat/gui/treeview_model.py: cleanup and mark method a pyqt slot
```

```
2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Change default viewer size for
        reconstructed images
        * vdat/gui/ifu_widget.py: Fix possible memory leak in creation of
        binned images

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: remove unused imports
    * vdat/gui/menubar.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/gui/fplane.py: use itertools.product for nested loops
    * vdat/gui/gui.py: removed
    * vdat/gui/ifu_viewer.py: PEP8
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/mainwidget.py: same
    * vdat/gui/tasks.py: same
    * doc/_source/codedoc/gui/index.rst: remove vdat.gui.gui

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Show reconstructed images
        * vdat/gui/central.py: Made pixelscale for reconstructed images␣
↪configurable
        * vdat/gui/ifu_widget.py: Show reconstructed images in ifu viewer
        fixes issues 1407 and 1409
        * vdat/config/vdat_setting.cfg: Added pixelscale for reconstructed␣
↪images

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/conf.py: don't check pyhetdex version and use pyhetdex/
↪latest
          for intersphinx

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * README.md: update
    * ReleaseNotes.md: add info about matplotlib removal
    * doc/_source/conf.py: remove matplotlib
    * doc/_source/install.rst: same
    * tests/conftest.py: same

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: get rid of APlpy and matplotlib
    * vdat/libvdat/vdat.py: same

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: add multiprocessing
```

```
2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: require pyhetdex 0.7.0
    * tests/test_config/test_core.py: adapt tests to the new config files
    * doc/_source/codedoc/gui/index.rst: fix typo

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: document dither tab type
    * vdat/config/tasks.yml: add all reduction science steps up to dividing by
          pixel flats
    * vdat/gui/fplane.py: update dither type implementation
    * vdat/gui/ifu_widget.py: fix bug #1405

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add cal reduction steps
    * vdat/gui/fplane.py: fix some bug with typ and fplane_single
    * vdat/gui/tasks.py: remove debugging prints
    * doc/_source/gui.rst: update documentation

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.3.0
    * ReleaseNotes.md: update

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: implement zro reduction steps
        * vdat/config/vdat_commands.yml: update commands to the latest␣
↪reduction
          steps
    * vdat/gui/central.py: fix indentation bug when submitting commands to
      queue; improve string sent to the queue
    * vdat/gui/fplane.py: improve FplaneWidget.change_fplane; add
      documentation, fix bug with matching directory names
    * vdat/gui/gui.py: make documentation happy
    * vdat/gui/tasks.py: accept commands as string or as list
    * vdat/gui/treeview_model.py: on selection changed pass the path of the
      directory
    * doc/_source/codedoc/gui/index.rst: move it from ../gui.rst, add
      placeholder table
    * doc/_source/codedoc/index.rst: adapt
    * doc/_source/codedoc/reduction.rst: fix module name
    * doc/_source/gui.rst: begin documentin tasks.yml file

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: remove debug print
    * vdat/gui/fplane.py: same
    * vdat/gui/ifu_widget.py: fix python3 bug

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/vdat_setting.cfg: remove tabs.yml and buttons.yml
    * vdat/config/core.py: update accordingly
    * vdat/config/entry_point.py: same
    * vdat/config/buttons.yml: remove
    * tests/test_buttons.py: same
    * vdat/config/tabs.yml: same
    * vdat/gui/buttons_menu.py: same

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: add multiprocessing arguments

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: added
    * vdat/config/vdat_setting.cfg: move max_delta_zro into the symlink␣
↪section
          and remove config_dirs
    * vdat/config/core.py: add config_dirs section when loading the config␣
↪file
    * vdat/libvdat/symlink.py: adapt to the above; add warning when no shot␣
↪file
          is found in a night
    * doc/_source/dirstruct.rst: document max_delta_zro
    * tests/test_config/test_core.py: adapt tests
    * tests/test_libvdat/test_symlink.py: add test for the above warning

2016-04-26 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: update documentation
    * doc/_source/launching.rst: same
    * vdat/config/vdat_setting.cfg: rename virus_dir to virus_instrument
    * tests/conftest.py: same
    * tests/test_libvdat/test_symlink.py: same
    * vdat/libvdat/symlink.py: update accordingly
    * vdat/libvdat/vdat.py: improve description

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: modify symlinking to comply to #1358
    * vdat/config/vdat_setting.cfg: add ``virus_dir`` entry to do the above
    * tests/data/raw/20151025/virus: add the virus directory to the test data
    * tests/conftest.py: adapt to the new directory structure
    * tests/test_ci/test_types.py: same
    * tests/test_libvdat/test_symlink.py: same

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add support for multiple raw or night␣
↪directories
    * tests/test_libvdat/test_symlink.py: adapt the tests

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/vdat.py: adjust docstring
    * tests/test_symlink.py: moved to tests/test_libvdat
    * tests/test_libvdat/test_vdat.py: added

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_loggers.py: add some more tests

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: fix bug in the command loggers
    * vdat/libvdat/loggers.py: clean up and fix few bugs
    * tests/test_loggers.py: add testing for the above modules

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: push variables to environment in a function,␣
↪improve
      command line arguments
    * vdat/config/vdat_setting.cfg: add is_rawdir_night option
    * vdat/config/core.py: skip also empty lists when overriding configuration
    * tests/test_config/test_core.py: test it

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: bugfix with ConfigParser file names; use mapping
          interface to insert a value
    * tests/test_config/test_core.py: test the vdat.config.core module
    * tests/conftest.py: adapt to changes in the vdat.config.core module

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: some function moved around, override_conf added
    * vdat/libvdat/vdat.py: first draft of the new command line interface
    * tests/conftest.py: add fixture to clear the internal configuration
          dictionary
    * tests/test_config: created
    * tests/test_config/test_core.py: added
    * tests/test_config.py: moved and renamed tests/test_config/test_entry_
↪point.py

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: I forgot to update the release notes

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: if a 100000 parameters in sql queries are␣
↪allowed,
          returns it, otherwise search for the number

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: remove SQLITE_MAX_COLUMN and add estimate of
```

```
          SQLITE_MAX_VARIABLE_NUMER
    * vdat/libvdat/symlink.py: use the latter when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: estimate SQLITE_MAX_COLUMN
    * vdat/libvdat/symlink.py: use the estimate when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: remove all the multiprocessing stuff and␣
↪improve
          log messages
    * vdat/libvdat/vdat.py: no multiprocessing things happening here
    * tests/test_symlink.py: update tests to the changes

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: make the types instance attribute
    * vdat/command_interpreter/types.py: fix __getattr__ to play nicely with
          pickling

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

     * tests/test_config.py: ignore .svn files
     * tests/conftest.py: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_ci/conftest.py: move here some useful fixture
    * tests/test_ci/test_signals.py: use the fixture
    * tests/test_ci/test_command_interpreter.py: test the core module to 99%
    * vdat/command_interpreter/core.py: if no file is collected return;␣
↪improve
          error from subprocess crashing

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/command_interpreter/core.py: make multiprocessing work
     * tests/test_ci/test_command_interpreter.py: add multiprocessing to the
          tests
     * setup.py: add pytest-xdist dependency for improved coverage
     * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/conf.py: use sphinx.ext.todo instead of
          pyhetdex.doc.sphinxext.tod
        * setup.py: bump sphinx version
        * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: make sure that no error is raised when running␣
```

```
↪the
          symlink of science frames

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: improve testing the symlink and solve issue #1335
    * vdat/config/vdat_setting.cfg: improve regex to match also file names␣
↪with
          decimal seconds

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

        * setup.py: bump required pyhetdex version to 0.6.0
    * vdat/command_interpreter/core.py: use DeferredResult when running jobs␣
↪in
          single processor mode
    * tests/test_ci/test_command_interpreter.py: adapt the tests; test the
          filter_section keyword in action; some other little fix

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move the logging from _run to run
          methods
    * vdat/command_interpreter/exceptions.py: add CISubprocessError
    * tests/test_ci/test_command_interpreter.py: adapt the tests

2016-04-14 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex version to 0.5
    * vdat/command_interpreter/core.py: worker created and removed in
          CommandInterpreter.run method
    * vdat/libvdat/symlink.py: worker created and removed in do_symlink␣
↪function
    * vdat/libvdat/vdat.py: remove workers, as they are handled where they are
          used

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: get and report exceptions when running
          the command
    * ReleaseNotes.md: update
    * setup.py: bump version to 0.2.4

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.2.3-post
    * vdat/command_interpreter/types.py: fix bug that makes file collection
          fails when using regex and directory names containing special␣
↪characters

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

        * ReleaseNotes.md: added; track history and help writing the report␣
↪for the
```

```
          next release

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: renamed, update documentation with
          info about multiprocessing
    * doc/_source/conf.py: use only pyhetdex latest for intersphinx
    * doc/_source/dirstruct.rst: add info about multiprocessing
    * doc/_source/executables.rst: expand a bit
    * doc/_source/launching.rst: same
    * doc/_source/index.rst: reorder some section
    * vdat/command_interpreter/core.py: try to enable multiprocessing, fail
          miserably
    * vdat/gui/buttons_menu.py: pass multiprocessing keywords
    * vdat/libvdat/vdat.py: close also command_interpreter worker

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat: rebased on ^/trunk
    * setup.py: version 0.2.3-post
    * tox.ini: force DISPLAY=:0

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: workaround bug with too many multiple␣
↪insertions; #1345
    * vdat/gui/gui.py: brute force implementation of re-symlink from gui
↪#1333;
          works, but needs review

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: negative error codes exists and are
          failures; fix the bug

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: don't close the multiprocessing worker to allow
          reusing it; set non existing rawdir to empty string; make public two
          functions that might be used from further symlinking
        * vdat/libvdat/vdat.py: wait and close the symlink worker
    * tests/test_symlink.py: update the tests

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged: branches/multiple_objects@169
        * setup.py: version set to 0.2.2
    * vdat/command_interpreter/core.py: log also the target dir then starting␣
↪a
          task

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: properly symlink science frames with empty␣
```

```
↪OBJECT
          fields and add counter to repeated OBJECT from different shots
    * tests/test_symlink.py: change test function name. The above changes has
          been tested only by hand
    * doc/_source/dirstruct.rst: add info about this

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * vdat/gui/treeview_model.py: add tool tip
    * tests/test_tree_view.py: test it
    * doc/_source/gui.rst: add some info about tooltip and right-click␣
↪commands
          available on the tree view

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: advance version
    * vdat/libvdat/symlink.py: fix multiprocessing while symlinking, now it
          works
    * vdat/libvdat/vdat.py: little changes because of the above

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: improve

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add references to zero and cal directories to
          every type

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: add version to command line
    * vdat/libvdat/vdat.py: same
    * vdat/gui/gui.py: add version to "About VDAT" menu; add links to
          documentation
    * vdat/gui/menu.py: pep8

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: version 0.2.1-pre
    * vdat/command_interpreter/types.py: extend the header type to allow
          formatting the values
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: write info log when a command start
          running
    * vdat/command_interpreter/types.py: add ``returns`` option to plain␣
↪primary
          type
```

```
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: create a general section and use it in
          every command; fix biassubtract fits matching to skip thumbnail fits

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: different shots with multiple lamps are now
          collected in the same cal directory
    * vdat/config/vdat_setting.cfg: add config tell the symlinking which␣
↪header
          keyword has the name of the lamps

2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve error message when regex␣
↪match
          does not work; fix bug with header type and multi-word header␣
↪keyword
          parsing
    * tests/test_ci/test_types.py: test this

2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: fix couple of bugs and remove copied files␣
↪if
          cloning fails
    * tests/test_tree_view.py: test 97% of the treeview_model.py (I don't␣
↪think
          I can do more)

2016-03-31 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_tree_view.py: test checkboxes also from the tree view
          perspective

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/symlink on top of trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * merge branches/cmd_interpreter_update into trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/cmd_interpreter_update on to of trunk

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: connect the global logger to the VDAT main logger
```

```
    * vdat/command_interpreter/core.py: fix typo

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: Create a global logger signal
    * vdat/command_interpreter/helpers.py: move the old default implementation
          here
    * vdat/command_interpreter/core.py: use the new global logger
    * tests/test_ci/test_signals.py: test the global logger
    * tests/test_ci/test_helpers.py: same

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: update documentation

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: connect some signal in order to have some␣
↪execution
          feedback

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: select directory also by enter key
    * tests/test_tree_view.py: test it (it's a workaround for the fact that
          there is a bug about testing clicks on tree view)

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: remove ``row`` from ReductionNode, as it's␣
↪not
          used
    * tests/test_tree_view.py: same

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: use pytest-catchlog instead of pytest-capturelog
    * tox.ini: same
    * tests/test_command_interpreter.py: adapt to the change
    * tests/test_tree_view.py: same
    * tests/test_symlink.py: same; do the symlinking in the test function, not
          setup

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: bugfix
    * tests/test_tree_view.py: mark old tests as integration, unit-test 46%␣
↪of the code

2016-03-17 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/database/models.py: add ``path`` attribute to the␣
↪VDATExposures table
    * vdat/libvdat/symlink.py: modify accordingly
```

```
    * vdat/gui/treeview_model.py: correctly propagate VDATExposures␣
↪information
          when cloning and removing directories; fixed bad bug in check/
↪uncheck
    * tests/test_tree_view.py: rewrite tests for the tree view, 40% done

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Renamed Raw to Exp in Tab descriptions

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add CICommandDone signal
    * vdat/command_interpreter/core.py: use it
    * vdat/command_interpreter/helpers.py: add a receiver that prints out␣
↪stuff
    * tests/test_ci/test_signals.py: test the new signal
    * tests/test_ci/test_helpers.py: test the new helper

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged with ^/trunk
        * tests/test_ci/test_types.py: update number of files

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: add @path@ to the @new_file@ type
    * tests/test_ci/test_types.py: add the tests
    * doc/_source/command_intepreter.rst: document it

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/libvdat/vdat.py: Moved first import of gui till after the
        XPA_METHOD was set based on config

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

        * merged ^/trunk

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove [args] section
    * vdat/database/models.py: remove ThumbnailScaling table
    * vdat/database/core.py: adapt
    * vdat/libvdat/callback.py: removed, as is unused
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/show_fits.py: same
    * doc/_source/codedoc/reduction.rst: remove callback

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: save the exposure database on files to be able␣
```

```
↪to
          rebuild the database from already symlinked directories
    * vdat/utilities.py: add utility function for the exposure database dump
    * vdat/database/base.py: use public functions to get the data from the
          model; provides 3 properties to get some of the data
    * vdat/database/models.py: override the data_clean property to skip the
          foreign fields
    * tests/test_symlink.py: adjust the tests to the changes, test the new␣
↪parts
    * tests/test_tree_view.py: little adjustment

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: update the new type names
    * vdat/libvdat/symlink.py: adapt the vdatexposures names
    * vdat/gui/buttons_menu.py: show the empty widget if no button is defined
          for the type
    * vdat/gui/treeview_model.py: get the type names from the database

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: adjust log messages about the type of the
          symlinked shot

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: proper cleanup when the symlinking fails, small
          adjustments.
    * tests/conftest.py: add virus*** related fixtures
    * tests/test_symlink.py: test to 100% the symlinking

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: export the database to the package level
    * vdat/libvdat/symlink.py: make the insertion in the database and the
          symlinking more robust to failures
    * vdat/utilities.py: add new error

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk
    * tests/test_symlink.py: adapt the tests

2016-03-07 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/symlink.py: fix bug in @_find_nearest@, remove optional
          argument from @symlink@ function
    * pytest.ini: add marker for integration tests
    * tests/conftest.py: add night and virus00001 fixtures
    * tests/test_symlink.py: add some unit tests

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: allow unknown/nonstandard object type linking
```

```
    * vdat/config/vdat_setting.cfg: add little explanation about it
    * doc/_source/dirstruct.rst: document it

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-03-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: first part of the new_file type
          implementation
    * tests/test_ci/test_types.py: test the new_file type
    * vdat/command_interpreter/core.py: adapt to the above
    * tests/test_ci/test_command_interpreter.py: same
    * doc/_source/command_intepreter.rst: adjust documentation
        * vdat/config/vdat_commands.yml: deformer 4 is no more

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/__init__.py: import get_signal and
          get_signal_names at the module level
    * vdat/command_interpreter/types.py: fix documentation
    * vdat/command_interpreter/utils.py: fix documentation
    * tests/test_ci/test_utils.py: add test of id_
    * doc/_source/codedoc/command_interpreter/types.rst: fix documentation

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: set xpa_method to local to avoid test hanging
    * vdat/command_interpreter/helpers.py: remove __all__
    * vdat/command_interpreter/signals.py: same
    * vdat/command_interpreter/types.py: import numpy
    * doc/_source/codedoc/command_interpreter/index.rst: split the command
          interpreter documentation
    * doc/_source/codedoc/command_interpreter/*.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/relay.py: renambed signals
    * vdat/command_interpreter/signals.py: rewritten to have an behaviour
          similar to qt signals; CILogger not reimplemented; some signal␣
↪missing
    * vdat/command_interpreter/__init__.py: remove the import of the relay
    * vdat/command_interpreter/core.py: update according to the above changes
    * vdat/command_interpreter/helpers.py: provide some function that can be
          plugged in
    * tests/test_ci/test_helpers.py: added
```

```
    * tests/test_ci/test_signals.py: added
    * doc/_source/command_intepreter.rst: relays -> signals
    * doc/_source/codedoc/command_interpreter.rst: moved to
          command_interpreter/index.rst
    * doc/_source/codedoc/index.rst: index in the codedoc to allow reshaping␣
↪of
          the documentation
    * doc/_source/index.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add a method that returns a range to
          SliceLike
    * vdat/command_interpreter/types.py: use SliceLike in primary_loop; remove
          _to_number function
    * tests/test_ci/test_utils.py: test the range method
    * tests/test_config.py: fixture to fix seed of random for repeatability of
          tests

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: adapt types to use the new
          keyword_regex and do_split = False
    * tests/test_ci/test_types.py: add tests for all the secondary keywords
    * doc/_source/command_intepreter.rst: fix the documentation

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add string and format customization␣
↪to
          SliceLike
    * tests/test_ci/test_utils.py: test it

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

        * merge trunk

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve the keyword_regex␣
↪functionality
    * vdat/command_interpreter/utils.py: homogenize doc
    * vdat/command_interpreter/core.py: move back types to class properties␣
↪to be
          able to run in parallel (this needs investigation)
    * vdat/command_interpreter/exceptions.py: fix typo
    * tests/test_ci/test_types.py: test the keyword_regex functionality
    * doc/_source/command_intepreter.rst: document the new keyword_regex

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: move some fixture to session scope
    * vdat/command_interpreter/core.py: move the types initialisation into the
          __init__
```

```
2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add SliceLike and copy some utils␣
↪from
      types.py
    * vdat/command_interpreter/exceptions.py: import the future and add
      CISliceError
    * setup.cfg: add doctest
    * tests/test_ci/test_utils.py: added
        * tests/test_ci/test_command_interpreter.py: moved
    * tests/test_ci/test_types.py: added and partially implemented
    * doc/_source/codedoc/command_interpreter.rst: add types and utils
      documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
          in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: check if the target directory exists, even␣
```

```
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
↪widget.py
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table
```

```
2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report
```

```
2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking

        use ``vdat_config copy`` to update the configuration files


2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection
```

```
2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
          button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
```

```
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
          to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
          logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
          functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
          failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                  a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>

      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
```

```
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
          changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
          necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
          initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
                          uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
↪avoid an
                                 astropy bug.
```

```
2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
```

```
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
                                the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
                              plane panels. What is displayed as a thumbnail
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                           rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                      selecting a node causes a whole new
                                      central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                     with a regex specifying the file type to
                                     display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list
```

```
2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
↪account for
```

```
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same
```

```
2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
                              try to regenerate
```

```
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                          select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here

2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
```

```
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
          set up
```

```
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
```

```
                                        use. It can show different reduction␣
↪steps in a
                                        branching hierachy.


2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
       * vdat/gui/static/unreduced.png: New image to differentiate
                                        between IFUs with and without input␣
↪files
       * vdat/libvdat/background.py: Uses the queue
       * vdat/libvdat/callback.py: Uses the queue
       * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
       * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

       Updated documentation and installation files:
       * doc/codedoc/gui.rst
       * doc/codedoc/reduction.rst
       * doc/index.rst
       * doc/queue.rst
       * requirements.txt
       * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

        * MANIFEST.in: Added fplane.txt file, so it is also installed!
        * doc/install.rst: Tweaked documentation
        * doc/launching.rst: As above
        * requirements.txt: Added command to install pyhetdex
        * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added Sphinx documentation (under doc/), minor
        modifications to comments

        * AUTHORS
```

```
        * LICENSE
        * README.md: Added new dependencies
        * doc/: Added documentation here
        * vdat/gui/gui.py
        * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

        *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
        *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
        *  vdat/gui/gui.py:    Added a create masterbias button
        *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
        *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
        *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
        *  vdat/libvdat/symlink.py
        *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure



2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
                                                 to pyhetdex:r74
```

```
2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
       * vdat/libvdat/background.py
       * vdat/libvdat/callback.py
       * vdat/libvdat/database.py
       * vdat/libvdat/fits.py
       * vdat/libvdat/handlers.py
       * vdat/libvdat/reduction.py: Added function to subtract overscans
       * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
       * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

     Added an internal sqlite3 database to keep track
     of what files are available. Created a background
     thread with which to run things so they don't lock
     up the GUI when they're running. Implemented
     a simple code which loops through all fits files
     and converts them to PNGs.

     * vdat/gui/__init__.py: Moved call to symlink to here
     * vdat/gui/gui.py:    Added a (currently disabled) progress bar
     * vdat/libvdat/background.py: run jobs in a separate thread
     * vdat/libvdat/callback.py: Added calls to Background
     * vdat/libvdat/database.py: Internal database to keep track of files
     * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
     * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
     * vdat/libvdat/symlink.py: Can read rawdir from config file
     * vdat/libvdat/vdat.py:  Moved symlink from here.
```

```
2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging


2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
```

```
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
          in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
↪widget.py
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
```

```
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
```

```
    * tox.ini: build the documentation and coverage report

2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking

        use ``vdat_config copy`` to update the configuration files


2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
```

```
        improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
        instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
        background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
        substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
        string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
        configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
        CommandInterpreter instances when pushing the buttons and report␣
↪problems
        with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
        buttons when changing directory; use a signal to change the central␣
↪and
        button panels
```

```
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
          to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
          logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
          functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
          failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                    a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>
```

```
      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
          changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
          necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
          initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
                          uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
```

```
↪avoid an
                                astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
```

```
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
                                the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
                              plane panels. What is displayed as a thumbnail
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                           rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                      selecting a node causes a whole new
                                      central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                     with a regex specifying the file type to
                                     display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging
```

```
2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                      to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
```

```
    * doc/_source/install.rst: update dependency list

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same
```

```
2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
```

```
          actions as per #1048, adapt the building of the tree view to␣
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
```

```
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
```

```
                                image of IFU is corrupted, if yes
                                try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                   stamps of FITs images when
                                   a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                   extra thread for it. This is designed
                                   for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                  of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                           select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here

2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
```

```
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
```

```
↪multiprocessing
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
```

```
        * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
                                          use. It can show different reduction␣
↪steps in a
                                          branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/gui/__init__.py: Create a queue
        * vdat/gui/buttons_menu.py: Added comments
        * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                              now there is just one type defined in
                              ifu_widget
        * vdat/gui/gui.py: Added a button
        * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                  methods to update the picture in the IFU
                                  to reflect whether the IFU has input files
                                  or not.
        * vdat/gui/queue.py: A queue window, which keeps track of the
                             commands a user has requested and runs
                             them when they reach the head of the queue. The
                             user can also delete these commands.
        * vdat/gui/static/unreduced.png: New image to differentiate
                                         between IFUs with and without input␣
↪files
        * vdat/libvdat/background.py: Uses the queue
        * vdat/libvdat/callback.py: Uses the queue
        * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
        * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

        Updated documentation and installation files:
        * doc/codedoc/gui.rst
        * doc/codedoc/reduction.rst
        * doc/index.rst
        * doc/queue.rst
        * requirements.txt
        * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

         * MANIFEST.in: Added fplane.txt file, so it is also installed!
         * doc/install.rst: Tweaked documentation
         * doc/launching.rst: As above
         * requirements.txt: Added command to install pyhetdex
         * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

         Added Sphinx documentation (under doc/), minor
         modifications to comments
```

```
        * AUTHORS
        * LICENSE
        * README.md: Added new dependencies
        * doc/: Added documentation here
        * vdat/gui/gui.py
        * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

        *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
        *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
        *  vdat/gui/gui.py:    Added a create masterbias button
        *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
        *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
        *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
        *  vdat/libvdat/symlink.py
        *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure



2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
```

```
                                                    to pyhetdex:r74


2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
       * vdat/libvdat/background.py
       * vdat/libvdat/callback.py
       * vdat/libvdat/database.py
       * vdat/libvdat/fits.py
       * vdat/libvdat/handlers.py
       * vdat/libvdat/reduction.py: Added function to subtract overscans
       * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
       * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

     Added an internal sqlite3 database to keep track
     of what files are available. Created a background
     thread with which to run things so they don't lock
     up the GUI when they're running. Implemented
     a simple code which loops through all fits files
     and converts them to PNGs.

     * vdat/gui/__init__.py: Moved call to symlink to here
     * vdat/gui/gui.py:    Added a (currently disabled) progress bar
     * vdat/libvdat/background.py: run jobs in a separate thread
     * vdat/libvdat/callback.py: Added calls to Background
     * vdat/libvdat/database.py: Internal database to keep track of files
     * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
     * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
     * vdat/libvdat/symlink.py: Can read rawdir from config file
```

```
    * vdat/libvdat/vdat.py:  Moved symlink from here.


2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging


2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
```

```
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: make sure that x11 can deal with threads

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add n_primaries signal to
          documentation
    * vdat/gui/progress_bar.py: implement the progress bar and connect to
          command_interpreter signals
    * vdat/gui/mainwidget.py: use the object in progress_bar.py module
    * tests/test_ci/conftest.py: move clean_connected to test/conftest.py
    * tests/test_gui: created
    * tests/test_gui/test_progress_bar.py: added
    * tests/test_gui/test_tree_view.py: moved into test_gui
    * tests/test_ci/test_signals.py: make sure to clean the signals

2016-05-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: new n_primaries signal
    * vdat/command_interpreter/helpers.py: helper function for that
    * vdat/command_interpreter/core.py: emit the signal
    * tests/test_ci/test_helpers.py: update tests
    * tests/test_ci/test_signals.py: same

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: update the configuration documentation
    * vdat/config/tasks.yml: add all the remaining reduction steps
    * vdat/config/vdat_setting.cfg: set the default pixel scale to 0.5, to␣
↪speed
          up the GUI startup
    * vdat/gui/fplane.py: don't raise an error if there are no tabs for a task

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwidget.py: cleanup old ways of communication
    * vdat/gui/mainwindow.py: same
    * vdat/gui/relay.py: remove unused signals
    * vdat/gui/treeview_model.py: cleanup and mark method a pyqt slot

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Change default viewer size for
        reconstructed images
        * vdat/gui/ifu_widget.py: Fix possible memory leak in creation of
```

```
        binned images

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: remove unused imports
    * vdat/gui/menubar.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/gui/fplane.py: use itertools.product for nested loops
    * vdat/gui/gui.py: removed
    * vdat/gui/ifu_viewer.py: PEP8
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/mainwidget.py: same
    * vdat/gui/tasks.py: same
    * doc/_source/codedoc/gui/index.rst: remove vdat.gui.gui

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Show reconstructed images
        * vdat/gui/central.py: Made pixelscale for reconstructed images␣
↪configurable
        * vdat/gui/ifu_widget.py: Show reconstructed images in ifu viewer
        fixes issues 1407 and 1409
        * vdat/config/vdat_setting.cfg: Added pixelscale for reconstructed␣
↪images

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: add multiprocessing

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: require pyhetdex 0.7.0
    * tests/test_config/test_core.py: adapt tests to the new config files
    * doc/_source/codedoc/gui/index.rst: fix typo

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: document dither tab type
    * vdat/config/tasks.yml: add all reduction science steps up to dividing by
          pixel flats
    * vdat/gui/fplane.py: update dither type implementation
    * vdat/gui/ifu_widget.py: fix bug #1405

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add cal reduction steps
    * vdat/gui/fplane.py: fix some bug with typ and fplane_single
    * vdat/gui/tasks.py: remove debugging prints
    * doc/_source/gui.rst: update documentation

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.3.0
    * ReleaseNotes.md: update
```

```
2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: implement zro reduction steps
        * vdat/config/vdat_commands.yml: update commands to the latest␣
↪reduction
          steps
    * vdat/gui/central.py: fix indentation bug when submitting commands to
      queue; improve string sent to the queue
    * vdat/gui/fplane.py: improve FplaneWidget.change_fplane; add
      documentation, fix bug with matching directory names
    * vdat/gui/gui.py: make documentation happy
    * vdat/gui/tasks.py: accept commands as string or as list
    * vdat/gui/treeview_model.py: on selection changed pass the path of the
      directory
    * doc/_source/codedoc/gui/index.rst: move it from ../gui.rst, add
      placeholder table
    * doc/_source/codedoc/index.rst: adapt
    * doc/_source/codedoc/reduction.rst: fix module name
    * doc/_source/gui.rst: begin documentin tasks.yml file

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: remove debug print
    * vdat/gui/fplane.py: same
    * vdat/gui/ifu_widget.py: fix python3 bug

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove tabs.yml and buttons.yml
    * vdat/config/core.py: update accordingly
    * vdat/config/entry_point.py: same
    * vdat/config/buttons.yml: remove
    * tests/test_buttons.py: same
    * vdat/config/tabs.yml: same
    * vdat/gui/buttons_menu.py: same

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: add multiprocessing arguments

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: added
    * vdat/config/vdat_setting.cfg: move max_delta_zro into the symlink␣
↪section
          and remove config_dirs
    * vdat/config/core.py: add config_dirs section when loading the config␣
↪file
    * vdat/libvdat/symlink.py: adapt to the above; add warning when no shot␣
↪file
          is found in a night
    * doc/_source/dirstruct.rst: document max_delta_zro
    * tests/test_config/test_core.py: adapt tests
```

```
    * tests/test_libvdat/test_symlink.py: add test for the above warning

2016-04-26 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: update documentation
    * doc/_source/launching.rst: same
    * vdat/config/vdat_setting.cfg: rename virus_dir to virus_instrument
    * tests/conftest.py: same
    * tests/test_libvdat/test_symlink.py: same
    * vdat/libvdat/symlink.py: update accordingly
    * vdat/libvdat/vdat.py: improve description

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: modify symlinking to comply to #1358
    * vdat/config/vdat_setting.cfg: add ``virus_dir`` entry to do the above
    * tests/data/raw/20151025/virus: add the virus directory to the test data
    * tests/conftest.py: adapt to the new directory structure
    * tests/test_ci/test_types.py: same
    * tests/test_libvdat/test_symlink.py: same

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add support for multiple raw or night␣
↪directories
    * tests/test_libvdat/test_symlink.py: adapt the tests

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: adjust docstring
    * tests/test_symlink.py: moved to tests/test_libvdat
    * tests/test_libvdat/test_vdat.py: added

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_loggers.py: add some more tests

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: fix bug in the command loggers
    * vdat/libvdat/loggers.py: clean up and fix few bugs
    * tests/test_loggers.py: add testing for the above modules

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: push variables to environment in a function,␣
↪improve
      command line arguments
    * vdat/config/vdat_setting.cfg: add is_rawdir_night option
    * vdat/config/core.py: skip also empty lists when overriding configuration
    * tests/test_config/test_core.py: test it

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/core.py: bugfix with ConfigParser file names; use mapping
          interface to insert a value
    * tests/test_config/test_core.py: test the vdat.config.core module
    * tests/conftest.py: adapt to changes in the vdat.config.core module

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: some function moved around, override_conf added
    * vdat/libvdat/vdat.py: first draft of the new command line interface
    * tests/conftest.py: add fixture to clear the internal configuration
          dictionary
    * tests/test_config: created
    * tests/test_config/test_core.py: added
    * tests/test_config.py: moved and renamed tests/test_config/test_entry_
↪point.py

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: I forgot to update the release notes

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: if a 100000 parameters in sql queries are␣
↪allowed,
          returns it, otherwise search for the number

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: remove SQLITE_MAX_COLUMN and add estimate of
          SQLITE_MAX_VARIABLE_NUMER
    * vdat/libvdat/symlink.py: use the latter when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: estimate SQLITE_MAX_COLUMN
    * vdat/libvdat/symlink.py: use the estimate when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: remove all the multiprocessing stuff and␣
↪improve
          log messages
    * vdat/libvdat/vdat.py: no multiprocessing things happening here
    * tests/test_symlink.py: update tests to the changes

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: make the types instance attribute
    * vdat/command_interpreter/types.py: fix __getattr__ to play nicely with
          pickling

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

     * tests/test_config.py: ignore .svn files
```

```
     * tests/conftest.py: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_ci/conftest.py: move here some useful fixture
    * tests/test_ci/test_signals.py: use the fixture
    * tests/test_ci/test_command_interpreter.py: test the core module to 99%
    * vdat/command_interpreter/core.py: if no file is collected return;␣
↪improve
          error from subprocess crashing

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/command_interpreter/core.py: make multiprocessing work
     * tests/test_ci/test_command_interpreter.py: add multiprocessing to the
          tests
     * setup.py: add pytest-xdist dependency for improved coverage
     * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/conf.py: use sphinx.ext.todo instead of
          pyhetdex.doc.sphinxext.tod
        * setup.py: bump sphinx version
        * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: make sure that no error is raised when running␣
↪the
          symlink of science frames

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: improve testing the symlink and solve issue #1335
    * vdat/config/vdat_setting.cfg: improve regex to match also file names␣
↪with
          decimal seconds

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

        * setup.py: bump required pyhetdex version to 0.6.0
    * vdat/command_interpreter/core.py: use DeferredResult when running jobs␣
↪in
          single processor mode
    * tests/test_ci/test_command_interpreter.py: adapt the tests; test the
          filter_section keyword in action; some other little fix

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move the logging from _run to run
          methods
    * vdat/command_interpreter/exceptions.py: add CISubprocessError
    * tests/test_ci/test_command_interpreter.py: adapt the tests
```

```
2016-04-14 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex version to 0.5
    * vdat/command_interpreter/core.py: worker created and removed in
          CommandInterpreter.run method
    * vdat/libvdat/symlink.py: worker created and removed in do_symlink␣
↪function
    * vdat/libvdat/vdat.py: remove workers, as they are handled where they are
          used

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: get and report exceptions when running
          the command
    * ReleaseNotes.md: update
    * setup.py: bump version to 0.2.4

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.2.3-post
    * vdat/command_interpreter/types.py: fix bug that makes file collection
          fails when using regex and directory names containing special␣
↪characters

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

        * ReleaseNotes.md: added; track history and help writing the report␣
↪for the
          next release

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: renamed, update documentation with
          info about multiprocessing
    * doc/_source/conf.py: use only pyhetdex latest for intersphinx
    * doc/_source/dirstruct.rst: add info about multiprocessing
    * doc/_source/executables.rst: expand a bit
    * doc/_source/launching.rst: same
    * doc/_source/index.rst: reorder some section
    * vdat/command_interpreter/core.py: try to enable multiprocessing, fail
          miserably
    * vdat/gui/buttons_menu.py: pass multiprocessing keywords
    * vdat/libvdat/vdat.py: close also command_interpreter worker

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat: rebased on ^/trunk
    * setup.py: version 0.2.3-post
    * tox.ini: force DISPLAY=:0

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: workaround bug with too many multiple␣
```

```
↪insertions; #1345
    * vdat/gui/gui.py: brute force implementation of re-symlink from gui
↪#1333;
          works, but needs review

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: negative error codes exists and are
          failures; fix the bug

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: don't close the multiprocessing worker to allow
          reusing it; set non existing rawdir to empty string; make public two
          functions that might be used from further symlinking
        * vdat/libvdat/vdat.py: wait and close the symlink worker
    * tests/test_symlink.py: update the tests

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged: branches/multiple_objects@169
        * setup.py: version set to 0.2.2
    * vdat/command_interpreter/core.py: log also the target dir then starting␣
↪a
          task

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: properly symlink science frames with empty␣
↪OBJECT
          fields and add counter to repeated OBJECT from different shots
    * tests/test_symlink.py: change test function name. The above changes has
          been tested only by hand
    * doc/_source/dirstruct.rst: add info about this

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * vdat/gui/treeview_model.py: add tool tip
    * tests/test_tree_view.py: test it
    * doc/_source/gui.rst: add some info about tooltip and right-click␣
↪commands
          available on the tree view

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: advance version
    * vdat/libvdat/symlink.py: fix multiprocessing while symlinking, now it
          works
    * vdat/libvdat/vdat.py: little changes because of the above

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: improve
```

```
2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add references to zero and cal directories to
          every type

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: add version to command line
    * vdat/libvdat/vdat.py: same
    * vdat/gui/gui.py: add version to "About VDAT" menu; add links to
          documentation
    * vdat/gui/menu.py: pep8

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: version 0.2.1-pre
    * vdat/command_interpreter/types.py: extend the header type to allow
          formatting the values
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: write info log when a command start
          running
    * vdat/command_interpreter/types.py: add ``returns`` option to plain␣
↪primary
          type
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: create a general section and use it in
          every command; fix biassubtract fits matching to skip thumbnail fits

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: different shots with multiple lamps are now
          collected in the same cal directory
    * vdat/config/vdat_setting.cfg: add config tell the symlinking which␣
↪header
          keyword has the name of the lamps

2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve error message when regex␣
↪match
          does not work; fix bug with header type and multi-word header␣
↪keyword
          parsing
    * tests/test_ci/test_types.py: test this
```

```
2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: fix couple of bugs and remove copied files␣
↪if
          cloning fails
    * tests/test_tree_view.py: test 97% of the treeview_model.py (I don't␣
↪think
          I can do more)

2016-03-31 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_tree_view.py: test checkboxes also from the tree view
          perspective

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/symlink on top of trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * merge branches/cmd_interpreter_update into trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/cmd_interpreter_update on to of trunk

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: connect the global logger to the VDAT main logger
    * vdat/command_interpreter/core.py: fix typo

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: Create a global logger signal
    * vdat/command_interpreter/helpers.py: move the old default implementation
          here
    * vdat/command_interpreter/core.py: use the new global logger
    * tests/test_ci/test_signals.py: test the global logger
    * tests/test_ci/test_helpers.py: same

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: update documentation

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: connect some signal in order to have some␣
↪execution
          feedback

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: select directory also by enter key
    * tests/test_tree_view.py: test it (it's a workaround for the fact that
```

```
          there is a bug about testing clicks on tree view)

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: remove ``row`` from ReductionNode, as it's␣
↪not
          used
    * tests/test_tree_view.py: same

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: use pytest-catchlog instead of pytest-capturelog
    * tox.ini: same
    * tests/test_command_interpreter.py: adapt to the change
    * tests/test_tree_view.py: same
    * tests/test_symlink.py: same; do the symlinking in the test function, not
          setup

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: bugfix
    * tests/test_tree_view.py: mark old tests as integration, unit-test 46%␣
↪of the code

2016-03-17 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/database/models.py: add ``path`` attribute to the␣
↪VDATExposures table
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/gui/treeview_model.py: correctly propagate VDATExposures␣
↪information
          when cloning and removing directories; fixed bad bug in check/
↪uncheck
    * tests/test_tree_view.py: rewrite tests for the tree view, 40% done

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Renamed Raw to Exp in Tab descriptions

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add CICommandDone signal
    * vdat/command_interpreter/core.py: use it
    * vdat/command_interpreter/helpers.py: add a receiver that prints out␣
↪stuff
    * tests/test_ci/test_signals.py: test the new signal
    * tests/test_ci/test_helpers.py: test the new helper

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged with ^/trunk
        * tests/test_ci/test_types.py: update number of files

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/command_interpreter/types.py: add @path@ to the @new_file@ type
    * tests/test_ci/test_types.py: add the tests
    * doc/_source/command_intepreter.rst: document it

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/libvdat/vdat.py: Moved first import of gui till after the
        XPA_METHOD was set based on config

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

        * merged ^/trunk

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove [args] section
    * vdat/database/models.py: remove ThumbnailScaling table
    * vdat/database/core.py: adapt
    * vdat/libvdat/callback.py: removed, as is unused
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/show_fits.py: same
    * doc/_source/codedoc/reduction.rst: remove callback

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: save the exposure database on files to be able␣
↪to
          rebuild the database from already symlinked directories
    * vdat/utilities.py: add utility function for the exposure database dump
    * vdat/database/base.py: use public functions to get the data from the
          model; provides 3 properties to get some of the data
    * vdat/database/models.py: override the data_clean property to skip the
          foreign fields
    * tests/test_symlink.py: adjust the tests to the changes, test the new␣
↪parts
    * tests/test_tree_view.py: little adjustment

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: update the new type names
    * vdat/libvdat/symlink.py: adapt the vdatexposures names
    * vdat/gui/buttons_menu.py: show the empty widget if no button is defined
          for the type
    * vdat/gui/treeview_model.py: get the type names from the database

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: adjust log messages about the type of the
          symlinked shot

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/symlink.py: proper cleanup when the symlinking fails, small
          adjustments.
    * tests/conftest.py: add virus*** related fixtures
    * tests/test_symlink.py: test to 100% the symlinking

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: export the database to the package level
    * vdat/libvdat/symlink.py: make the insertion in the database and the
          symlinking more robust to failures
    * vdat/utilities.py: add new error

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk
    * tests/test_symlink.py: adapt the tests

2016-03-07 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/symlink.py: fix bug in @_find_nearest@, remove optional
          argument from @symlink@ function
    * pytest.ini: add marker for integration tests
    * tests/conftest.py: add night and virus00001 fixtures
    * tests/test_symlink.py: add some unit tests

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: allow unknown/nonstandard object type linking
    * vdat/config/vdat_setting.cfg: add little explanation about it
    * doc/_source/dirstruct.rst: document it

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-03-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: first part of the new_file type
          implementation
    * tests/test_ci/test_types.py: test the new_file type
    * vdat/command_interpreter/core.py: adapt to the above
    * tests/test_ci/test_command_interpreter.py: same
    * doc/_source/command_intepreter.rst: adjust documentation
        * vdat/config/vdat_commands.yml: deformer 4 is no more

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/command_interpreter/__init__.py: import get_signal and
          get_signal_names at the module level
    * vdat/command_interpreter/types.py: fix documentation
    * vdat/command_interpreter/utils.py: fix documentation
    * tests/test_ci/test_utils.py: add test of id_
    * doc/_source/codedoc/command_interpreter/types.rst: fix documentation

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: set xpa_method to local to avoid test hanging
    * vdat/command_interpreter/helpers.py: remove __all__
    * vdat/command_interpreter/signals.py: same
    * vdat/command_interpreter/types.py: import numpy
    * doc/_source/codedoc/command_interpreter/index.rst: split the command
          interpreter documentation
    * doc/_source/codedoc/command_interpreter/*.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/relay.py: renambed signals
    * vdat/command_interpreter/signals.py: rewritten to have an behaviour
          similar to qt signals; CILogger not reimplemented; some signal␣
↪missing
    * vdat/command_interpreter/__init__.py: remove the import of the relay
    * vdat/command_interpreter/core.py: update according to the above changes
    * vdat/command_interpreter/helpers.py: provide some function that can be
          plugged in
    * tests/test_ci/test_helpers.py: added
    * tests/test_ci/test_signals.py: added
    * doc/_source/command_intepreter.rst: relays -> signals
    * doc/_source/codedoc/command_interpreter.rst: moved to
          command_interpreter/index.rst
    * doc/_source/codedoc/index.rst: index in the codedoc to allow reshaping␣
↪of
          the documentation
    * doc/_source/index.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add a method that returns a range to
          SliceLike
    * vdat/command_interpreter/types.py: use SliceLike in primary_loop; remove
          _to_number function
    * tests/test_ci/test_utils.py: test the range method
    * tests/test_config.py: fixture to fix seed of random for repeatability of
          tests

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: adapt types to use the new
          keyword_regex and do_split = False
    * tests/test_ci/test_types.py: add tests for all the secondary keywords
    * doc/_source/command_intepreter.rst: fix the documentation
```

```
2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add string and format customization␣
↪to
          SliceLike
    * tests/test_ci/test_utils.py: test it

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

        * merge trunk

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve the keyword_regex␣
↪functionality
    * vdat/command_interpreter/utils.py: homogenize doc
    * vdat/command_interpreter/core.py: move back types to class properties␣
↪to be
          able to run in parallel (this needs investigation)
    * vdat/command_interpreter/exceptions.py: fix typo
    * tests/test_ci/test_types.py: test the keyword_regex functionality
    * doc/_source/command_intepreter.rst: document the new keyword_regex

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: move some fixture to session scope
    * vdat/command_interpreter/core.py: move the types initialisation into the
          __init__

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add SliceLike and copy some utils␣
↪from
      types.py
    * vdat/command_interpreter/exceptions.py: import the future and add
      CISliceError
    * setup.cfg: add doctest
    * tests/test_ci/test_utils.py: added
        * tests/test_ci/test_command_interpreter.py: moved
    * tests/test_ci/test_types.py: added and partially implemented
    * doc/_source/codedoc/command_interpreter.rst: add types and utils
      documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
          in py2 and py3
```

```
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
↪widget.py
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
```

```
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report

2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking

        use ``vdat_config copy`` to update the configuration files
```

```
2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
```

```
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
          button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
          to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
```

```
↪and
          logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
          functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
          failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                  a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>

      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
          changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
          necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
          initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
                          uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
```

```
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
↪avoid an
                                 astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
```

```
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
                                the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
                              plane panels. What is displayed as a thumbnail
```

```
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
       * vdat/gui/gui.py: Central panel now generated dynamically
                          rather than at initialization.
       * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
       * vdat/gui/relay.py: Added 'change_centralPanel' signal.
       * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                      selecting a node causes a whole new
                                      central panel to be created
       * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                    with a regex specifying the file type to
                                    display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
```

```
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
```

```
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
```

```
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
```

```
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
                              try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                          select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here

2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>
```

```
    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities
```

```
    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
```

```
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
                                         use. It can show different reduction␣
↪steps in a
                                         branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
       * vdat/gui/static/unreduced.png: New image to differentiate
                                          between IFUs with and without input␣
↪files
```

```
        * vdat/libvdat/background.py: Uses the queue
        * vdat/libvdat/callback.py: Uses the queue
        * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
        * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

        Updated documentation and installation files:
        * doc/codedoc/gui.rst
        * doc/codedoc/reduction.rst
        * doc/index.rst
        * doc/queue.rst
        * requirements.txt
        * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

         * MANIFEST.in: Added fplane.txt file, so it is also installed!
         * doc/install.rst: Tweaked documentation
         * doc/launching.rst: As above
         * requirements.txt: Added command to install pyhetdex
         * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

         Added Sphinx documentation (under doc/), minor
         modifications to comments

         * AUTHORS
         * LICENSE
         * README.md: Added new dependencies
         * doc/: Added documentation here
         * vdat/gui/gui.py
         * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

         *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
         *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
         *  vdat/gui/gui.py:    Added a create masterbias button
         *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
         *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
         *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
         *  vdat/libvdat/symlink.py
         *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure
```

```
2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
                                                 to pyhetdex:r74



2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
       * vdat/libvdat/background.py
       * vdat/libvdat/callback.py
```

```
        * vdat/libvdat/database.py
        * vdat/libvdat/fits.py
        * vdat/libvdat/handlers.py
        * vdat/libvdat/reduction.py: Added function to subtract overscans
        * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
        * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

     Added an internal sqlite3 database to keep track
     of what files are available. Created a background
     thread with which to run things so they don't lock
     up the GUI when they're running. Implemented
     a simple code which loops through all fits files
     and converts them to PNGs.

     * vdat/gui/__init__.py: Moved call to symlink to here
     * vdat/gui/gui.py:    Added a (currently disabled) progress bar
     * vdat/libvdat/background.py: run jobs in a separate thread
     * vdat/libvdat/callback.py: Added calls to Background
     * vdat/libvdat/database.py: Internal database to keep track of files
     * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
     * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
     * vdat/libvdat/symlink.py: Can read rawdir from config file
     * vdat/libvdat/vdat.py:  Moved symlink from here.


2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging


2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>
```

```
        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
```

```
↪same
          in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
↪widget.py
```

```
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session
```

```
2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report

2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking
```

```
        use ``vdat_config copy`` to update the configuration files


2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
```

```
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
          button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
```

```
        to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
        logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
        functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
        failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                  a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>

      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
        changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
        necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
        initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
```

```
                        uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                        focal plane. User can select scaling per fits file,␣
↪or a global
                        scaling (which can be user specified) for the whole␣
↪focal plane.
                        The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
↪avoid an
                                 astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
```

```
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
                                the ifu_viewer and main subsections).
```

```
        * vdat/gui/fplane.py: Added tools to save and generate focal
                              plane panels. What is displayed as a thumbnail
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                           rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                       selecting a node causes a whole new
                                       central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                     with a regex specifying the file type to
                                     display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
```

```
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
```

```
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
```

```
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database
```

```
2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
```

```
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
                              try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                          select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here
```

```
2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        WARNING: this changes break the gui button functionalities

    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
```

```
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
                                          use. It can show different reduction␣
↪steps in a
                                          branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
       * vdat/gui/static/unreduced.png: New image to differentiate
```

```
                                        between IFUs with and without input␣
↪files
      * vdat/libvdat/background.py: Uses the queue
      * vdat/libvdat/callback.py: Uses the queue
      * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
      * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

      Updated documentation and installation files:
      * doc/codedoc/gui.rst
      * doc/codedoc/reduction.rst
      * doc/index.rst
      * doc/queue.rst
      * requirements.txt
      * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

       * MANIFEST.in: Added fplane.txt file, so it is also installed!
       * doc/install.rst: Tweaked documentation
       * doc/launching.rst: As above
       * requirements.txt: Added command to install pyhetdex
       * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

       Added Sphinx documentation (under doc/), minor
       modifications to comments

       * AUTHORS
       * LICENSE
       * README.md: Added new dependencies
       * doc/: Added documentation here
       * vdat/gui/gui.py
       * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

       *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
       *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
       *  vdat/gui/gui.py:    Added a create masterbias button
       *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
       *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
       *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
       *  vdat/libvdat/symlink.py
       *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
```

```
↪specifying the VIRUS filename structure



2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
                                                 to pyhetdex:r74



2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
```

```
      * vdat/libvdat/background.py
      * vdat/libvdat/callback.py
      * vdat/libvdat/database.py
      * vdat/libvdat/fits.py
      * vdat/libvdat/handlers.py
      * vdat/libvdat/reduction.py: Added function to subtract overscans
      * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
      * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

    Added an internal sqlite3 database to keep track
    of what files are available. Created a background
    thread with which to run things so they don't lock
    up the GUI when they're running. Implemented
    a simple code which loops through all fits files
    and converts them to PNGs.

    * vdat/gui/__init__.py: Moved call to symlink to here
    * vdat/gui/gui.py:    Added a (currently disabled) progress bar
    * vdat/libvdat/background.py: run jobs in a separate thread
    * vdat/libvdat/callback.py: Added calls to Background
    * vdat/libvdat/database.py: Internal database to keep track of files
    * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
    * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
    * vdat/libvdat/symlink.py: Can read rawdir from config file
    * vdat/libvdat/vdat.py:  Moved symlink from here.


2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging
```

```
2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2017-10-18  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/trunk in ^/branches/text_file_tab

2017-10-18  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add text_file tab for the dither file

2017-10-12  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/static/icons: add a subset of the "macOS Icons" icons
          https://store.kde.org/p/1102582/
    * vdat/gui/__init__.py: if the theme name is not set, set it to the above
          name. Resolves issue #2133

2017-09-27  Francesco Montesano  <montefra@mpe.mpg.de>
```

```
    * tests/test_gui/conftest.py: update fixture
    * tests/test_gui/test_tabs/test_ifu_viewer.py: 100% coverage of the
          ifu_viewer module
    * vdat/gui/tabs/ifu_viewer.py: make some little change to allow testing

2017-09-26  Francesco Montesano  <montefra@mpe.mpg.de>

    * doc/_source/_static/fits_viewer.png: update screenshot
    * doc/_source/_static/text_file_viewer.png: same

2017-09-26  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add the quit and help menu to the ifu␣
↪viewer
          windows

2017-09-25  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/menubar.py: plug the Help menu and get rid of the unnecessary
          signals
        * vdat/gui/mainwindow.py: remove slots and connection incorporated␣
↪into the
          Help menu. Second part of issue #2135
    * vdat/gui/help_window.py: set reasonable size for the help window
    * tests/test_gui/test_mainwindow.py: remove tests

2017-09-25  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: isolate the help menu for better reusability.
          First part of issue #2135
    * tests/test_gui/test_menus_actions.py: test it

2017-09-22  Francesco Montesano  <montefra@mpe.mpg.de>

    * doc/_source/gui/ifu_viewer.rst: describe the new text file window
    * doc/_source/gui/main_panel.rst: finish documentation of the text file␣
↪tab
    * doc/_source/_static/fplane_text_file.png: add screenshot
    * doc/_source/_static/text_file_viewer.png: same

2017-09-21  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: if the file exists, open it in the file
          editor
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it

2017-09-21  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add basic text editor
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test it

2017-09-20  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: create QuitAction
```

```
    * vdat/gui/menubar.py: use it
    * tests/test_gui/test_menus_actions.py: test it

2017-09-18  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: prepare al image with the number of lines
    * vdat/gui/tabs/tab_widget.py: no need to add the cleanup method
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes
    * tests/test_gui/test_tabs/test_tab_widget.py: the fake IFU are not needed

2017-09-13  Francesco Montesano  <montefra@mpe.mpg.de>

        * setup.py: add text_file tab plugin. First part of issue #2108
    * vdat/gui/tabs/entry_points.py: implement the entry point for text_file
    * vdat/gui/tabs/tab_widget.py: implement the tab type (I think that it's
          everything that we need)
    * vdat/gui/tabs/ifu_widget.py: implement the setup method of the ifu␣
↪widget.
          It still doesn't display anything
    * doc/_source/gui/main_panel.rst: begin documenting the changes
    * tests/test_gui/test_fplane.py: test the new plugin
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the initialisation and
          setup of the new ifu widget
    * tests/test_gui/test_tabs/test_tab_widget.py: test the new tab widget
    * tests/test_list_plugins.py: update the number of plugins

2017-10-11  Niv Drory  <drory@astro.as.utexas.edu>

        * config/vdat_commands.yml (subtract_os): remove deceptively evil -s␣
↪flag.

2017-10-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/branches/exclude_ifus into ^/trunk

2017-10-04  Francesco Montesano  <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: add some info about the fplane file

2017-09-29  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: add option to respect empty columns and␣
↪rows
          when displaying the focal plane. Resolves issue #2139
    * vdat/gui/tabs/tab_widget.py: use it
    * vdat/gui/central.py: exclude the IFUs also in the main part of the gui
    * tests/test_gui/conftest.py: adapt the tests
    * tests/test_gui/test_tabs/test_tab_widget.py: same

2017-09-27  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: add option to ignore IFUSLOT. Resolves
          issue #2138
    * vdat/config/versions.py: bump the vdat_setting.cfg version to 1.1.0
```

```
    * vdat/gui/tabs/tab_widget.py: propagate the above option to the focal␣
↪plane
          in the GUI
    * tests/test_config/test_versions.py: update the versions

2017-09-15  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: forgot to copy the updated fplane file (updated
          from fplane20170202.txt in the virus_config repo)

2017-09-15  Francesco Montesano  <montefra@mpe.mpg.de>

        * : merge ^/branches/fplane_map into ^/trunk
    * vdat/gui/static/VirusDataAnalysisTool.qch: update
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-09-15  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: use the fplane_map type
    * vdat/config/tasks.yml: update the commands to use it
    * vdat/config/versions.py: update the version

2017-09-14  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: add the fplane_map type. Resolves␣
↪issue
      #2115
    * doc/_source/command_interpreter.rst: document it
    * setup.py: advertise it
    * tests/test_ci/test_types.py: test it

2017-09-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tasks.yml: Fixed mistake which caused some
      tabs to be repeated

2017-09-07  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: Fixed fiberextract mandatories

2017-09-06  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: correct the dither_file executable call
    * vdat/config/vdat_commands.yml: same

2017-09-05  Francesco Montesano  <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: fix typo in variable name (modelbalse ->
          modelbase in the mkdither command)
    * vdat/config/tasks.yml: fix the above type, fix error in Fiber extracted
          tab title

2017-08-22  Jan Snigula  <snigula@mpe.mpg.de>
```

```
        * vdat/config/vdat_commands.yml: Update filename prefixes and
        pixelflat filenames.
        deformer step creates dists/fmods for flat and twilight.
        flatnorm runs for flat and twilight models.
        * vdat/config/tasks.yml: Moved pixelflat step before ccdcombine
        Added buttons to use twilight dist/fmod for skysubtraction and
        fiberextract

2017-08-21  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: pixflat_lib dir defaults to ./pixel_
↪lib
        * vdat/config/tasks.yml: Use pixelflats by default

2017-08-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/tasks.yml: Use master twilight frames for deformer

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.7.0-post

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: require pyhetdex 0.12.0, bump version to 0.7.0
    * ReleaseNotes.md: update
    * tox.ini: remove devel pyhetdex version dependency
    * vdat/gui/static/VirusDataAnalysisTool.qch: update
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/next_pyhetdex/ into ^/trunk

2017-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_libvdat/test_loggers.py: fix number of log files created

2017-08-11  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/versions.py: Bumped version numbers
        * vdat/config/tasks.yml: More pipeline updates
        * vdat/config/vdat_commands.yml: Same

2017-08-09  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/config/vdat_commands.yml: Use library bias and dark frames
        * vdat/config/tasks.yml: Same
        * vdat/libvdat/vdat.py: Fixed multiprocessing on OS X

2017-04-20 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_gui_utils.py: get rid of warnings
    * tests/test_libvdat/test_symlink.py: same
```

```
    * vdat/gui/utils.py: same
    * tests/test_gui/test_tabs/test_ifu_widget.py: improve float comparison

2017-08-16 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: re-add cov-init, use devel version of pyhetdex everywhere

2017-07-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: use new pyhetdex.tools.db_helpers module
    * vdat/libvdat/symlink.py: same
    * tests/test_database.py: remove unnecessary tests
    * vdat/database/__init__.py: remove unnecessary imports

2017-06-28 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: add python 3.6
    * vdat/config/entry_point.py: use pyhetdex functions. Resolves issue #1964
    * tests/test_config/test_entry_point.py: update the tests

2017-06-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: depends on configparser on python 2
    * tox.ini: use devel pyhetdex
    * vdat/config/core.py: use configparser; resolves #1983
    * vdat/config/versions.py: same
    * vdat/gui/menubar.py: same
    * vdat/gui/utils.py: same
    * vdat/libvdat/vdat.py: same
    * tests/test_gui/test_gui_utils.py: same, get rid of warnings
    * tests/test_gui/test_mainwindow.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_loggers.py: same
    * tests/test_libvdat/test_symlink.py: same
    * tests/test_libvdat/test_vdat.py: same

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to v0.6.1-post

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: prepare for v0.6.1 release
    * ReleaseNotes.md: same

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: use pyhetdex override_conf instead of its own
          implementation. Resolves issue #1850
    * tests/test_config/test_core.py: update tests

2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/peewee_connect into ^/trunk
```

```
2017-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_mainwidget.py: wrap every database query/
↪modification
          within a connection. Solved issue #1897
    * tests/test_gui/test_menubar.py: same
    * tests/test_gui/test_menus_actions.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_symlink.py: same
    * vdat/gui/central.py: same
    * vdat/gui/menubar.py: same
    * vdat/gui/menus_actions.py: same
    * vdat/gui/tabs/entry_points.py: same
    * vdat/gui/tabs/tab_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/database/core.py: open and close the database only if it's not on
          memory

2017-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * tox.ini: ignore little-deploy failures
    * vdat/database/core.py: connect only if the database is closed. Fixes␣
↪issue
          #1895.
    * vdat/libvdat/symlink.py: wrap some query into db.connect()

2017-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * scripts/symlink_pyqt.sh: fix bug

2017-02-03 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: prepare v0.6.0 release
    * ReleaseNotes.md: same

2017-02-03 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/branches/debug_logs into ^/trunk

2017-02-03 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk into ^/branches/debug_logs

2017-01-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add option to process events while the thread is
          running
    * tests/test_gui/test_gui_utils.py: test it
    * vdat/gui/mainwidget.py: use it to update the GUI as the symlinking is
          running (fixes #1765)

2017-01-31 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * tests/test_gui/test_mainwidget.py: add tests
    * vdat/gui/mainwidget.py: do only the symlink in the thread

2017-02-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: workaround to fix issue #1782

2017-02-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: resolves issue #1782
    * tests/test_gui/test_tabs/test_ifu_widget.py: adapt mocking in tests

2017-02-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: improve documentation, resolves issue #1779
    * vdat/gui/static/VirusDataAnalysisTool.qch: add the new documenation
    * vdat/gui/static/VirusDataAnalysisTool.qhc: same

2017-02-01 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/tabs/ifu_widget.py: uses astropy's zscale. Fixes Issue
↪#1777

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk into ^/branches/debug_logs

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add function to run the FuncQThread with the waiting
          cursor.
    * vdat/gui/mainwindow.py: use it
    * vdat/gui/treeview_model.py: use it
    * vdat/gui/mainwidget.py: use it when running the symlink from the GUI.
          Solves issue #1765
    * tests/test_gui/test_gui_utils.py: add tests
    * tests/test_gui/test_mainwidget.py: added. Need to finish testing the
          widget, but I first need to merge the other branch as it has some␣
↪fixture
          I need

2017-01-27 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: tmp_dir is now a of tmpdir with the possibility to
          remove the directory; add fixtures
    * tests/test_gui/test_init.py: move fixture to conftest
        * tests/test_integration_vdat.py: add regression/integration test for␣
↪the
          bug in #1317
    * vdat/gui/__init__.py: partial refactoring to help testing
    * vdat/libvdat/vdat.py: add argv option to main for testing purposes

2017-01-24 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/__init__.py: don't set the logger level
    * vdat/gui/logger_widget.py: set the handler level to INFO
    * tests/conftest.py: fix sip.setapi to v2 for all the types

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/static/VirusDataAnalysisTool.*: update documentation with the
          latest changes

2017-01-30 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/branches/plug_tabs_1533 into ^/trunk

2017-01-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/list_plugins.py: add functionality to list entry points. Solves␣
↪issue
          #1613
    * vdat/gui/fplane.py: move entry point group name to variable
    * tests/test_list_plugins.py: test the new functionality
    * doc/_source/gui/main_panel.rst: add some documentation
    * doc/_source/codedoc/index.rst: same
    * doc/_source/codedoc/list_plugins.rst: same
    * setup.py: add the entry point

2017-01-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: make "Individual" radio button react to
          button release to allow repainting the GUI. Resolved issue #1679
    * doc/_source/gui/main_panel.rst: add some hint about this

2017-01-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: reset individual selection when dropping␣
↪fits
          tabs
    * tests/test_gui/test_tabs/test_tab_widget.py: update the tests

2017-01-10 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: move the new plugin implementation away.
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: move it here and␣
↪update
          the rest

2017-01-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: port all the tasks to the new tab types. Resolves
          #1724

2017-01-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: make sure that the user is not spammed with
```

```
        reconstruction error if the reconstruction object is not present.␣
↪Resolved
        #1725

2016-12-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/interface.py: add the enabled property
    * vdat/gui/fplane.py: set the tab enabled or not
    * tests/test_gui/conftest.py: move fixture here
    * tests/test_gui/test_central.py: from here
    * tests/test_gui/test_fplane.py: test tab enabled

2016-12-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: add overlay if no tab displayed; fixes issue #1603
    * tests/test_gui/test_fplane.py: test it and fix other tests accordingly

2016-12-15 Francesco Montesano <montefra@mpe.mpg.de>

        fixes issue #1720

    * doc/_source/conf.py: add ginga intersphinx
    * vdat/gui/tabs/entry_points.py: remove unused entry points
    * vdat/gui/tabs/ifu_viewer.py: fix documentation and remove unused code
    * vdat/gui/tabs/ifu_widget.py: same

2016-12-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: fix documentation
    * vdat/gui/tabs/interface.py: same
    * vdat/gui/tabs/tab_widget.py: same and remove old code

2016-12-14 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/send_to_ds9.png: added
    * doc/_source/gui/ifu_viewer.rst: add some documentation about the fits
        viewer

2016-12-14 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/ifu_viewer.rst: add screenshot
    * doc/_source/gui/main_panel.rst: add information about the header_keys
        keyword
    * doc/_source/_static/fits_viewer.png: added
    * vdat/config/tasks.yml: add the header_keys keyword
    * vdat/gui/tabs/ifu_viewer.py: remove line with file name

016-12-14 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: finish the tabs documentation
    * doc/_source/gui/ifu_viewer.rst: add basic info
    * doc/_source/gui/menu_bar.rst: add references
        * doc/_source/_static/vdat_main_panel.png: add
        * doc/_source/_static/vdat_screenshot.png: update
```

```
2016-12-13 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: document the reconstruct tab type
    * doc/_source/_static/fplane_reconstruct.png: add screenshot

2016-12-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: create unique file names using md5 hash.
          Fixes issue #1714

2016-12-12 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/tabs/ifu_widget.py: pass title and tooltip to fits window;␣
↪fixes
          issue #1715
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes

2016-12-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add ability to pass custom tab titles and
          tooltip
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test it

2016-12-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: unset basenames
    * vdat/gui/tabs/tab_widget.py: pass basenames to the ifus
    * vdat/gui/tabs/entry_points.py: add the "reconstruct" tab type
    * setup.py: add the entry point
    * vdat/config/tasks.yml: use it
    * doc/_source/gui/main_panel.rst: add the entry point in the documentation
    * tests/test_gui/test_fplane.py: test the new entrypoint
    * tests/test_gui/test_tabs/test_tab_widget.py: test the changes

2016-12-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: add basenames property to allow loop over
          them to combine multiple exposures
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes

2016-12-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: rename combined to fits_combined
    * setup.py: same
    * tests/test_gui/test_fplane.py: same
    * doc/_source/gui/main_panel.rst: add the fits_combined documentation
    * doc/_source/_static/fplane_fits_combined.png: add image
    * vdat/config/tasks.yml: re-add all the cal steps

2016-12-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_central.py: re-enable the empty fplane tab type for
          testing only
```

```
    * tests/test_gui/conftest.py: move some fixture here
    * tests/test_gui/test_fplane.py: modify accordingly
    * vdat/gui/fplane.py: change pkg_resources import to allow mocking
    * vdat/gui/tabs/entry_points.py: make the empty_fplane private

2016-12-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: implement the combined tab type
    * setup.py: add its entry point, remove old tab entry points
    * tests/test_gui/test_fplane.py: test the new entry points, remove old␣
↪entry
          point tests
    * doc/_source/gui/main_panel.rst: add the entry point docstring

2016-12-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_queue.py: disable some more test for python2
    * vdat/config/tasks.yml: add the first three steps of the zro reduction␣
↪with
          the new tab system

2016-12-05 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/main_panel.rst: add documentation of the exp_combined␣
↪tab
          type
    * doc/_source/_static/fplane_exp_combined.png: added

2016-12-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: add exp_combined entry point␣
↪implementation
    * vdat/gui/tabs/ifu_widget.py: extend the number of ids that is possible␣
↪to
          use the fits file name; fix bug with cleanup in the base class
    * vdat/gui/tabs/interface.py: fix documentation
    * vdat/gui/tabs/tab_widget.py: create the title and, if given, the tool␣
↪tip;
          remove the tooltip on cleanup
    * doc/_source/gui/main_panel.rst: begin the documentation
    * setup.py: add the exp_combined entry point
    * tests/test_gui/test_fplane.py: test the new entry point
    * tests/test_gui/test_tabs/test_tab_widget.py: fix the tests for the title
          and tooltip

2016-12-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: the base class cleanup method set the empty
          image and paint it (issue #1688)
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it

2016-12-01 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
```

```
    * vdat/gui/tabs/tab_widget.py: add the widgets in the setup method only
    * tests/test_gui/test_tabs/test_tab_widget.py: update tests

2016-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: remove the widgets from the layout when␣
↪doing
          the cleanup and readd them on setup as a workaround for the non-
↪showing
          problem when readded into the fplane
    * tests/test_gui/test_tabs/test_tab_widget.py: update tests
    * tests/test_gui/test_fplane.py: make sure the first widget is selected

2016-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add temporary entry point for the stacked widget
    * vdat/gui/tabs/entry_points.py: add the entry point function for the
          stacked widget
    * vdat/gui/tabs/tab_widget.py: work on the aspect of the stacked widget.␣
↪Use
          a check box in the bar at the bottom of the focal plane
    * vdat/gui/utils.py: move the vertical spacer here
    * tests/test_gui/test_fplane.py: add an integration test
    * tests/test_gui/test_tabs/test_tab_widget.py: test the changes

2016-11-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: disable the switch button if no
          reconstruction object present
    * tests/test_gui/test_tabs/test_tab_widget.py: test it

2016-11-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: fix bug
    * tests/test_gui/test_tabs/test_tab_widget.py: according to tests the␣
↪widget
          works

2016-11-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: a first implementation of issue #1669 is␣
↪done
    * tests/test_gui/test_tabs/test_tab_widget.py: start testing (still all to
          do, though)

2016-11-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: extend clean/update cached thumbnails to␣
↪the
          quick reconstruction widget
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the changes
    * tests/test_gui/test_queue.py: disable some test for python2

2016-11-25 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/tabs/ifu_widget.py: clean/update cached thumbnails if needed
          (issue #1660)
    * vdat/gui/utils.py: update function
    * tests/test_gui/test_gui_utils.py: update tests
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the new implementation

2016-11-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: make sure that the header keywords are
          correctly reordered (issue #1675)
    * tests/test_gui/test_tabs/test_ifu_viewer.py: add regression test
    * tox.ini: raise the bar for success of the coverage

2016-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: show the reconstructed image in the fits
          viewer window
    * tests/test_gui/test_tabs/test_ifu_widget.py: test some of it

2016-11-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add possibility to show header keywords at
          the beginning
    * vdat/gui/tabs/ifu_widget.py: pass the tab configuration dictionary
    * vdat/gui/tabs/entry_points.py: document
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test the header keyword
          reordering
    * tests/test_gui/test_tabs/test_ifu_widget.py: update tests

2016-11-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: open the fits viewer on double click
    * vdat/gui/tabs/ifu_viewer.py: zoom to fit on first show
    * tests/test_gui/test_tabs/test_ifu_widget.py: add tests

2016-11-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: fix bug introduced with fixing #1646
    * tests/test_gui/test_queue.py: re-enabled some test that would have␣
↪avoided
          this bug

2016-11-17 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: make sure selections are correctly propagated to␣
↪all
          tabs (#1671)
    * vdat/gui/tabs/ifu_widget.py: same
    * vdat/gui/tabs/tab_widget.py: same
    * tests/test_gui/conftest.py: fix fixtures
    * tests/test_gui/test_central.py: test the changes
    * tests/test_gui/test_fplane.py: same
```

```
2016-11-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: add the reconstructed IFU pluging
          (issue #1633)
    * vdat/gui/tabs/ifu_widget.py: implement the IFU widget for it
    * vdat/gui/tabs/tab_widget.py: implement the tab widget for it
    * vdat/gui/utils.py: add the prefix for the reconstructed files
    * setup.py: add the entry point
    * ReleaseNotes.md: update
    * tests/test_gui/conftest.py: new fixture of the reconstructed object
    * tests/test_gui/test_fplane.py: test the new plugin
    * tests/test_gui/test_tabs/test_ifu_widget.py: test the new widget
    * tests/test_gui/test_tabs/test_tab_widget.py: same

2016-11-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: remove all the reconstruction hints from␣
↪the
          FitsFplanePanel class
    * vdat/gui/tabs/ifu_widget.py: store some more info, remove old pieces of
          code
    * vdat/gui/tabs/entry_points.py: cleanup the plugin
    * tests/test_gui/test_tabs/test_tab_widget.py: fix tests accordingly
        * ReleaseNotes.md: updated

2016-11-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: react to changing the scale
    * vdat/gui/tabs/ifu_widget.py: add attributes storing the global scaling␣
↪and
          use them, if available, to paint the fits files
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/test_tabs/test_tab_widget.py: test it

2016-11-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: add zmin/zmax property
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * vdat/gui/tabs/tab_widget.py: get the global zscale for the IFUs and␣
↪store
          it; add needed functionality for this; add cleanup
    * tests/test_gui/test_tabs/test_tab_widget.py: test it

2016-11-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add header information
    * vdat/gui/tabs/ifu_widget.py: ifu_viewer classes name changed
        * ReleaseNotes.md: udpated with issue #1631

2016-11-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: start rewriting the fits viewer. Show the
          files in independent tabs.
    * tests/test_gui/test_tabs/test_ifu_viewer.py: test most of the new
```

```
        implementation
    * tests/test_gui/test_central.py: move fixtures from here ...
    * tests/test_gui/conftest.py: ... to here

2016-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: recreate thumbnail if it's older than the
        fits file; clear cached thumbnail if the fits file does not exist.
    * tests/test_gui/test_tabs/test_ifu_widget.py: test this

2016-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: move the thumbnailing code here
    * tests/test_gui/test_gui_utils.py: test it
    * vdat/gui/tabs/ifu_widget.py: implement prepare_image to produce␣
↪thumbnails
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/conftest.py: add fixture

2016-11-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_viewer.py: add base class for the IFU viewers
    * doc/_source/codedoc/gui/tabs/ifu_viewer.rst: add inheritance tree

2016-11-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: update documentation; get type from
        database
    * vdat/gui/tabs/tab_widget.py: add setup method
    * vdat/gui/tabs/ifu_widget.py: add and implement setup method
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * tests/test_gui/test_fplane.py: add fits_fplane integration test

2016-11-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: improve error messages at plugin loading time (issue
        #1655)
    * tests/test_gui/test_fplane.py: update tests

2016-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: add extra "step_name" parameter to pass to the␣
↪plugins
    * vdat/gui/tabs/entry_points.py: same
    * vdat/gui/tabs/interface.py: same
    * vdat/gui/tabs/tab_widget.py: begin implementing the setup
    * vdat/gui/tabs/ifu_widget.py: same

2016-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: create fits_fplane entry point
    * setup.py: same
    * vdat/gui/tabs/tab_widget.py: add scale buttons to the GUI
    * vdat/gui/tabs/ifu_widget.py: remove unused method
```

```
    * tests/test_gui/test_tabs/test_tab_widget.py: test new class init

2016-10-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/ifu_widget.py: don't trigger single clicks on double click
          (issue #1639)
    * tests/test_gui/test_tabs/test_ifu_widget.py: test it
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: add some info about
          this
    * ReleaseNotes.md: updated with the issue reference

2016-10-25 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: updated
    * vdat/gui/queue.py: remove the command and thread when the command␣
↪finishes
          to run
    * vdat/gui/tabs/tab_widget.py: set parent in the thread

2016-10-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: don't draw the tab
    * vdat/gui/tabs/ifu_widget.py: fix hw_size, cleanup code
    * vdat/gui/tabs/tab_widget.py: use showEvent, properly select/unselect IFU
          to make it reply back
    * tests/test_gui/test_fplane.py: add regression test to make sure that
          selecting/deselecting all IFUs work as expected
    * tests/test_gui/test_tabs/test_ifu_widget.py: update tests
    * tests/test_gui/test_tabs/test_tab_widget.py: same
    * tests/test_gui/test_mainwindow.py: fix weird interaction with other␣
↪tests
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: update with the
          showEvent info
    * doc/_source/gui/main_panel.rst: add link to the above document

2016-10-21 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/codedoc/gui/tabs/entry_points.rst: added
    * doc/_source/codedoc/gui/tabs/new_type_example.rst: add an example on how
          to use and extend the code
    * doc/_source/codedoc/gui/index.rst: add them to index
    * vdat/gui/tabs/entry_points.py: set parent explicitly
    * vdat/gui/tabs/ifu_widget.py: fix documentation

2016-10-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/entry_points.py: add empty_fplane tab type using
          BaseFplanePanel
    * vdat/gui/tabs/interface.py: fix bug with error message
    * doc/_source/gui/main_panel.rst: add empty_fplane documentation
    * setup.py: add empty_fplane entry points
    * tests/test_gui/test_fplane.py: test that the new entry point behave as
          expected
```

```
2016-10-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: fix bugs
    * vdat/gui/tabs/ifu_widget.py: fix cleanup documentation
    * tests/test_gui/conftest.py: update fixture for better use
    * tests/test_gui/test_central.py: use the new fixtures
    * tests/test_gui/test_tabs/test_tab_widget.py: test BaseFplanePanel

2016-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: implement the BaseFplanePanel widget,␣
↪rename
          some of the UpdateIFUTask parts
    * vdat/gui/tabs/ifu_widget.py: add TODO on a slot
    * vdat/gui/tabs/interface.py: update documentation
    * doc/_source/codedoc/gui/tabs/tab_widget.rst: show full inheritance
        * tests/test_gui/test_tabs/test_tab_widget.py: update the␣
↪UpdateIFUTask
          names

2016-10-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/tab_widget.py: update the UpdateIFUTask to run
    * tests/test_gui/test_tabs/test_tab_widget.py: test it
    * vdat/gui/tabs/ifu_widget.py: update documentation

2016-10-18 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_central.py: move a fixture to the conftest
    * tests/test_gui/conftest.py: update above fixture
    * tests/test_gui/test_tabs/test_ifu_widget.py: added, test the␣
↪BaseIFUWidget
    * vdat/gui/tabs/ifu_widget.py: fix renaming bugs

2016-10-17 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/codedoc/gui/tabs/ifu_widget.rst: add inheritance map
    * vdat/gui/tabs/ifu_widget.py: first iteration of the IFU widget base␣
↪class
    * vdat/gui/tabs/tab_widget.py: fix names according to the above changes

2016-10-14 Francesco Montesano <montefra@mpe.mpg.de>


        * doc/_source/codedoc/gui/index.rst: use the new tabs/* files
    * doc/_source/codedoc/gui/tabs: created
    * doc/_source/codedoc/gui/tabs/ifu_viewer.rst: same
        * doc/_source/codedoc/gui/tabs/ifu_widget.rst: renamed from
          ../ifu_widget.rst
        * doc/_source/codedoc/gui/tabs/interface.rst: renambed from tabs_
↪plugins
    * doc/_source/codedoc/gui/tabs/tab_widget.rst: created
    * doc/_source/conf.py: add inheritance diagram
    * vdat/gui/fplane.py: remove all the fplane panel implementation
```

```
    * vdat/gui/tabs/tab_widget.py: move here the fplane panel code
    * vdat/gui/ifu_viewer.py: move to tabs/ifu_viewer.py
    * vdat/gui/ifu_widget.py: move to tabs/ifu_widget.py

2016-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/index.rst: fix typo
    * doc/_source/gui/main_panel.rst: update the documentation
    * vdat/gui/tasks.py: PEP8 fix
        * doc/_source/conf.py: fix intersphinx link
        * ReleaseNotes.md: update

2016-09-30 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/fplane.py: remove yield_*_ifu methods and get rid of the
          VDATRunControl object
    * vdat/gui/treeview_model.py: don't use it

2016-09-30 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_fplane.py: test the FplaneWidget
    * vdat/gui/fplane.py: fix some bug and adjust edge cases

2016-09-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: Implement the plugin system for the tabs
    * vdat/gui/tabs/interface.py: adjust interfaces
    * doc/_source/codedoc/gui/fplane.rst: add the FplaneWidget to the docs
    * tox.ini: don't run the doc-qt if the qt* executables do not exist

2016-09-27 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * vdat/gui/tabs/interface.py: add the interface for the function
          implementing the plugin
    * doc/_source/gui/main_panel.rst: add a bit of documentation

2016-09-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/tabs/__init__.py: added
    * vdat/gui/tabs/interface.py: added; implement the FplaneTabTemplate class
          (issue #1601)
    * doc/_source/codedoc/gui/tabs_plugins.rst: added
    * doc/_source/codedoc/gui/index.rst: add tabs_plugins to the index
    * doc/_source/gui/main_panel.rst: add the FplaneTabTemplate to the
          documentation

2016-09-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: reimplement the FplaneCache as a pure cache without
          much intelligence
    * tests/test_gui/test_fplane.py: test it
    * doc/_source/gui/main_panel.rst: begin documenting the plugin system
    * doc/_source/codedoc/gui/fplane.rst: change the documentation to better
```

```
          separate parts

2016-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: create here cache and call the reconstruction object
          in the appropriate place; fix super calls
    * vdat/gui/central.py: move the cache creation in the fplane module
    * tests/test_gui/test_central.py

2016-08-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: don't pass a reconstruct object to the FplaneCache
          object. Add property that returns it using the function create in␣
↪r440
    * vdat/gui/central.py: remove the reconstruction object creation and adapt
          the code
    * tests/test_gui/test_central.py: remove tests about the reconstruction
          object
    * ReleaseNotes.md: update

2016-08-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add function that creates the QuickReconstruct object
          only once and returns it
    * tests/test_gui/test_gui_utils.py: test it
    * tests/conftest.py: add fixture to deal with it
    * tests/test_gui/conftest.py: move around some fixture
    * tests/test_gui/test_central.py: same and adapt

2016-08-18 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk into ^/branches/plug_tabs_1533

2016-08-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menubar.py: emit a signal instead of closing the VDAT (issue
          #1568)
    * vdat/gui/mainwindow.py: connect the above signal with the mainwindow␣
↪close
          slot; isolate connections with the menu bar into a method
    * tests/test_gui/test_mainwindow.py: test closing via the above signal
        * ReleaseNotes.md: update

2016-08-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/help_window.py: add support for mailto (issue #1579)
    * tests/test_gui/test_help_window.py: test it
    * setup.py: update dependencies to coverage>=4.2
    * tox.ini: same, remove obsolete cov-init environment

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/splash_screen_1562 into ^/trunk
```

```
2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/splash_screen_1562
        * vdat/gui/static/VirusDataAnalysisTool*: updated and renamed

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: remove always on top flag, make surer that the
          splashscreen is shown
    * tests/test_gui/test_init.py: update the tests

2016-08-01 Francesco Montesano <montefra@mpe.mpg.de>

    * AUTHORS: Majo added
    * ReleaseNotes.md: updated
    * doc/_source/_static/vdat_icon.ico: added
    * doc/_source/_static/vdat_name.jpg: added
    * doc/_source/_static/vdat_screenshot.png: updated
    * doc/_source/command_interpreter.rst: fix typo
    * doc/_source/conf.py: add the VDAT logo and name
    * setup.py: fix ginga and pytest-qt to older version
    * tox.ini: same
    * tests/test_gui/test_init.py: added
    * vdat/gui/__init__.py: work around splash screen not showing

2016-08-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: do all the splash screen things here
    * vdat/gui/mainwindow.py: add window icon
    * vdat/gui/static/HETDEX_*.jpg: new images and renamed to VDAT_*.jpg
    * vdat/gui/static/vdat_icon.jpg: added and used as icon
    * vdat/libvdat/vdat.py: just create and show the splash screen

2016-07-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: add splash screen functionality
    * vdat/gui/static/HETDEX_black.jpg: added
    * vdat/gui/static/HETDEX_white.jpg: same
    * vdat/libvdat/vdat.py: show the splash screen

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/qt_help_1454/ into ^/trunk

2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>


    * ReleaseNotes.md: update
    * vdat/gui/menubar.py: add about Qt button
    * doc/_source/_static/menubar_help.png: update accordingly
    * doc/_source/gui/menu_bar.rst: same
        * tests/test_gui/test_menubar.py: same
    * vdat/gui/static/VDATVirusDataAnalysisTool.*: add updated documentation
```

```
2016-08-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwindow.py: remove the logging gui handlers when closing␣
↪down
    * vdat/database/core.py: fix connect context manager for in memory␣
↪database
    * tests/test_database.py: test it
    * tests/test_gui/test_mainwindow.py: add and test the whole mainwindow
    * tests/test_gui/test_help_window.py: make the monkeypatch of static␣
↪method
          work on py2
    * tests/test_gui/test_queue.py: temporarily skip some of the offending␣
↪tests

2016-08-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/help_window.py: don't run setup in the __init__ for better␣
↪test.
          Setup the data only after connecting the slot to setup the content␣
↪widget
    * vdat/gui/menubar.py: set help menu object name for easier testing
    * vdat/gui/utils.py: don't cover run method
    * tests/test_gui/test_gui_utils.py: add test for static directory and␣
↪qthelp
          file
    * tests/test_gui/test_help_window.py: added
    * tests/test_gui/test_menubar.py: add test for when no documentation is
          found

2016-08-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/help_window.py: added to address issue #1454
    * vdat/gui/mainwindow.py: add slot to open the help window
    * vdat/gui/menubar.py: add menu item to open the help window
    * vdat/gui/static/VDATVirusDataAnalysisTool.qch: add documentation
    * vdat/gui/static/VDATVirusDataAnalysisTool.qhc: same
    * vdat/gui/utils.py: add functions to get the documentation file
    * doc/Makefile: make qthelp now also creates the qch and qhc files
    * doc/_source/codedoc/gui/help_window.rst: added
    * doc/_source/codedoc/gui/index.rst: add
    * doc/_source/contributions.rst: add information about the qt␣
↪documentation
    * tox.ini: add you environment to create the qt documentation

2016-08-02 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/contributions.rst: update information (issue #1529)

2016-08-02 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/db_version_cleanup into ^/trunk

2016-07-04 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/ifu_widget.py: convert property to pyqtProperty (issue #1532)
    * vdat/gui/menus_actions.py: same
    * vdat/gui/treeview_model.py: same
    * tests/test_libvdat/test_symlink.py: add more randomization to decrease␣
↪the
          possible file name clashes

2016-08-02 Francesco Montesano <montefra@mpe.mpg.de>

    * pytest.ini: set qt_api=pyqt4
    * setup.py: set pytest-qt>=0.2, skip broken version of ginga
    * tests/test_database.py: fix test
    * tests/test_gui/test_progress.py: same
    * tox.ini: fix pytest-qt version and remove qt_api

2016-07-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/models.py: update VDATExposures according to issue #1530
    * vdat/database/check.py: update the conversion function
    * vdat/libvdat/symlink.py: update the creation of the exposures entries
    * tests/conftest.py: update the current exps dictionary
    * tests/test_database.py: adapt tests
    * tests/test_gui/test_menus_actions.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_symlink.py: same
    * ReleaseNotes.md: update with info about the issue

2016-07-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/models.py: remove virtual and ifus fields as per issue
↪#1339
    * vdat/database/check.py: update the conversion function
    * tests/conftest.py: update the current shot, move here db initialization
          routines
    * tests/test_database.py: adapt the tests
    * tests/test_gui/test_central.py: same
    * ReleaseNotes.md: update with info about the issue

2016-07-01 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/db_version_cleanup

2016-07-01 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * tests/test_database.py: make sure that metadata are moved

2016-07-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/check.py: when updating the meta data, remove superfluous
          entries
    * vdat/libvdat/symlink.py: improve error messages when inserting existing
          meta data
    * tests/test_database.py: test the changes
```

```
    * tests/test_libvdat/test_symlink.py: same
    * tests/conftest.py: shuffle fixtures around

2016-06-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/launching.rst: add some info about vdat_db command
    * vdat/database/check.py: after updating the metadata try to add it to the
          database
    * tests/test_database.py: test it
    * doc/_source/codedoc/utilities.rst: added
    * doc/_source/codedoc/index.rst: added to the index

2016-06-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/check.py: add possibility to repair the database
    * tests/test_database.py: test it

2016-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/check.py: make vdat_db check subcommand
    * tests/test_database.py: test the changes

2016-06-20 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add enum34 for python < 3.4 and colorama; add vdat_db entry␣
↪point
    * vdat/database/check.py: added, implement db checks
    * vdat/database/core.py: allow to create db in memory (and don't close it)
    * vdat/utilities.py: function that reads the shot and exps files can also
          return the dir containing the files
    * tests/test_database.py: test the new check module
    * tests/test_utilities.py: update tests

2016-06-17 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: move the function to scan for metadata files to
          vdat/utilities.py and adapt
    * vdat/utilities.py: update the above function implementation
    * tests/test_libvdat/test_symlink.py: adapt the tests
    * tests/test_utilities.py: same

2016-06-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/base.py: add database exception class
    * vdat/database/models.py:
    * vdat/database/__init__.py: make merge_entries more robust against type
          mismatches
    * tests/test_database.py: test the database subpackage
    * tests/test_database: removed

2016-06-30 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/branches/config_version into ^/trunk
```

```
2016-06-30 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/config_version

2016-06-30 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: fix typo
    * vdat/config/entry_point.py: don't add common parser to the parent one

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/codedoc/database.rst: show class inheritance
    * vdat/database/models.py: add version as a table column with a check
    * vdat/database/__init__.py: explicitly import stuff at the package level
    * vdat/database/core.py: remove __all__
    * vdat/libvdat/symlink.py: add the table versions
    * tests/test_gui/test_menubar.py: same
    * tests/test_gui/test_menus_actions.py: same
    * tests/test_gui/test_tree_view.py: same
    * tests/test_libvdat/test_symlink.py: same
    * tests/test_database: added for the future
        * ReleaseNotes.md: updated

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update with info about issue #1501

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: disable the reconstruction checkbox and tab if the
          reconstruction object does not exist
    * vdat/gui/central.py: remove unnecessary method

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_central.py: finish testing the central module
    * vdat/gui/mainwindow.py: fix documentation typo

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: modify imports to allow monkeypatching for the␣
↪tests;
    * tests/test_gui/test_central.py: add first part of the central module␣
↪tests

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: don't fail if the quick reconstruction object
          creation fails; add documentation
    * vdat/config/entry_point.py: fix typo
    * doc/_source/codedoc/gui/central.rst: added
    * doc/_source/codedoc/gui/tasks.rst: added
    * doc/_source/codedoc/gui/index.rst: add them to the index
```

```
2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md
    * vdat/config/entry_point.py: add backup option to the vdat_config copy
          command
    * doc/_source/launching.rst: document it
    * tests/test_config/test_entry_point.py: test it

2016-06-13 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/launching.rst: add the compare command to the doc
    * vdat/config/entry_point.py: fix one of the info messages

2016-06-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/versions.py: add the remaining loaders
    * tests/test_config/test_versions.py: test them

2016-06-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/versions.py: add some loader to check if files are␣
↪reasonable
    * vdat/config/entry_point.py: add ``-l/--load`` option
    * tests/test_config/test_entry_point.py: test the above
    * tests/test_config/test_versions.py: same

2016-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: some improvement on the messages
    * tests/test_config/test_entry_point.py: update the tests

2016-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: add diff of files
    * tests/test_config/test_entry_point.py: test it

2016-06-09 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add colorama for nice coloured output
    * vdat/config/entry_point.py: add a compare subcommand and check if files
          exist and have the correct version
    * vdat/config/versions.py: add helper functions
    * tests/test_config/test_entry_point.py: test it
    * tests/test_config/test_versions.py: test it

2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/__init__.py: explicitly import attributes
    * vdat/config/core.py: remove __all__
    * vdat/config/versions.py: explicit file names to copy and file versions
    * vdat/config/entry_point.py: rewrite the copy function to use the list of
          names and to replace version numbers
    * vdat/config/tasks.yml: add version
    * vdat/config/vdat_commands.yml: add version
```

```
    * vdat/config/vdat_setting.cfg: add version
    * tests/test_config/test_entry_point.py: update the tests
    * tests/test_config/test_versions.py: add
    * doc/_source/codedoc/config.rst: add the versions module

2016-06-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: improved the documentation␣
↪(hopefully)
    * vdat/gui/ifu_viewer.py: Fixed a small bug created by a previous rename
      of the ihmp attribute in the IFU object to ifuslot.

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/log_sig_1492 into ^/trunk

2016-06-15 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/trunk into ^/branches/log_sig_1492

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py: use the command_intepreter command_logger into␣
↪the
          QCommandInterpreter, connecting/disconnecting the default helper␣
↪slot in
          the ``run`` method
    * tests/test_gui/test_queue.py: PEP8
    * tests/test_libvdat/test_symlink.py: remove unused import
    * ReleaseNotes.md: update release notes

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/branches/clone_remove into ^/trunk
    * ReleaseNotes.md: update

2016-06-14 Francesco Montesano <montefra@mpe.mpg.de>

    * : merge ^/trunk into ^/branches/clone_remove
    * ReleaseNotes.md: update

2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: re-emit the sig_exposure_removed signal in␣
↪the
          ReductionQTreeView; return the menu created at right-click
    * tests/test_gui/test_tree_view.py: properly test the menu and that the
          signal is re-emitted

2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: add sig_exposure_removed signal to address
        issue #1498
    * tests/test_gui/test_menus_actions.py: test the emission
```

```
2016-06-08 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui/tree_view.rst: add Remove exposure description
    * doc/_source/_static/tree_view_clone_remove.png: update screenshot

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: clear the menu before adding new actions
    * tests/test_gui/test_menus_actions.py: test the menus_actions module
    * doc/_source/codedoc/gui/menus_actions.rst: added
    * doc/_source/codedoc/gui/index.rst: add to the index

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menus_actions.py: implement a dynamic menu to remove exposures
    * vdat/gui/treeview_model.py: use it

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: clone in a thread and wait for it to finish;
          set the cursor to wait cursor when cloning or removing a directory
    * tests/test_gui/test_tree_view.py: adapt to the changes and test that
          cloning within or without the thread leads to the same result

2016-06-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: explicitly create actions to clone/remove␣
↪and
          connect them to slots
    * tests/test_gui/test_tree_view.py: update the tests
    * doc/_source/_static/tree_view_clone_remove.png: update image

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: create signal to perform the␣
↪command
          logger
    * vdat/command_interpreter/core.py: use it to communicate the
          success/failure of the single sub-commands
    * vdat/command_interpreter/helpers.py: add default implementation for a␣
↪slot
          for the above signal
    * tests/test_ci/conftest.py: adapt the tests to the changes
    * tests/test_ci/test_command_interpreter.py: same
    * tests/test_ci/test_helpers.py: same
    * tests/test_ci/test_signals.py: same

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/changelog.rst: include changelog verbatim
    * setup.cfg: move coverage configuration here
    * .coveragerc: remove
```

```
2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump the version and the required pyhetdex version
    * ReleaseNotes.md: update

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_gui/test_progress.py: fix failing test

2016-06-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: updated to the newest format and values

2016-06-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Updated to new fplane format
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/central.py: Same

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/menubar.py: add possibility to specialize removal with the type
    * vdat/config/vdat_setting.cfg: add documentation and example on how to do
          this
    * doc/_source/gui/menu_bar.rst: document it
    * tests/test_gui/test_menubar.py: test it

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: fix typo
    * vdat/gui/utils.py: add function to remove files
    * vdat/gui/menubar.py: sig_remove_files are emitted with the error and␣
↪thumb
          files
    * vdat/gui/mainwindow.py: connect the new menu bar signal with a slot and
          document the class
    * tests/test_gui/test_gui_utils.py: test the removal
    * tests/test_gui/test_menubar.py: adapt the tests
    * doc/_source/codedoc/gui/fplane.rst: added
    * doc/_source/codedoc/gui/ifu_widget.rst: added
    * doc/_source/codedoc/gui/index.rst: added
    * doc/_source/codedoc/gui/mainwidget.rst: added
    * doc/_source/codedoc/gui/mainwindow.rst: added
    * doc/_source/gui/menu_bar.rst: update the documentation

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/install.rst: fix documentation warning
        * tests/test_ci/test_ci_utils.py: renamed from test_utils.py to avoid␣
↪pytest
          error
        * tests/test_gui/test_gui_utils.py: renamed from test_utils.py to␣
↪avoid
          pytest error
```

```
2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/utils.py: add a custom QThread that executes one function
    * tests/test_gui/test_utils.py: test the utilities
    * doc/_source/codedoc/gui/index.rst: add the utilities to the␣
↪documentation
    * doc/_source/codedoc/gui/utils.rst: same

2016-06-03 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/menubar_file.png: added
    * doc/_source/_static/menubar_help.png: added
    * doc/_source/_static/menubar_select.png: added
    * doc/_source/_static/menubar_view.png: added
    * doc/_source/codedoc/gui/menubar.rst: added
    * doc/_source/codedoc/gui/index.rst: add the above
    * doc/_source/gui/menu_bar.rst: document behaviour with links
    * doc/_source/install.rst: fix link
    * setup.py: bump version
    * tests/test_gui/test_menubar.py: test the menubar module
    * vdat/config/vdat_setting.cfg: add option with name of the files to␣
↪remove
    * vdat/gui/menubar.py: add the delete file actions
    * vdat/gui/utils.py: add the thumbnails directory name

2016-06-03 Danuel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/progress.py: Changed "Running" to "Finished" in the status bar,
          to better reflect the actual state of the job.

    Also changed the wording in some of the documentation:
    * doc/_source/command_interpreter.rst
    * doc/_source/dirstruct.rst
    * doc/_source/gui/index.rst
    * doc/_source/gui/main_panel.rst
    * source/gui/menu_bar.rst
    * doc/_source/gui/progress.rst
    * doc/_source/gui/queue.rst
    * doc/_source/gui/tree_view.rst
    * doc/_source/install.rst
    * doc/_source/launching.rst

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.4.0

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: add hints to use virus_config files
    * vdat/config/tasks.yml: update the commands

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/fplane.txt: updated with the latest values

2016-06-01 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/faq.rst: added
    * doc/_source/index.rst: add it to the index
    * vdat/libvdat/symlink.py: fix typos in error messages

2016-05-31 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * doc/_source/install.rst: solve #1452
    * doc/_source/contributions.rst: update

2016-05-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: add creation of mastertwi or masterflat
          from twilight frames; remove fake pixel flat
    * vdat/config/tasks.yml: add the above commands
    * vdat/libvdat/symlink.py: improve error message
    * vdat/config/entry_point.py: add space when asking to the user
    * tests/test_libvdat/test_loggers.py: adapt the tests to the removal of␣
↪the
          fake pixel flat

2016-05-31 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/symlink.py: add customisation of the types to use for
          calibration
    * vdat/config/vdat_setting.cfg: add/modify the configuration options to
          support the changes above
    * vdat/libvdat/vdat.py: add command line overrides for the above changes
    * tests/test_libvdat/test_symlink.py: test combination of calibration␣
↪types;
          get rid of virus00001 fixture
    * doc/_source/dirstruct.rst: document the changes

2016-05-30 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/launching.rst: update
    * doc/_source/dirstruct.rst: update
    * vdat/libvdat/vdat.py: update command line option name

2016-05-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/executables.rst: removed
    * doc/_source/index.rst: adapted
    * doc/_source/launching.rst: start improving the use information

2016-05-25  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Fix global cutlevel estimation fixing issue1482
        * vdat/gui/ifu_widget.py: Fix bad levels in thumbnails
```

```
2016-05-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: don't emit the number of primaries if
          it's zero
    * vdat/gui/queue.py: properly emit finished when CommandInterpreter.run
          returns
    * tests/test_ci/test_command_interpreter.py: adapt test

2016-05-25 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/vdat.py: skip interpolation error when setting CUREBIN␣
↪in the
      environment
    * tests/test_libvdat/test_vdat.py: add tests for this

2016-05-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/progress.py: renamed from vdat/gui/progress_bar.py; move
          custom status bar here
    * vdat/gui/mainwidget.py: adapt
    * vdat/gui/mainwindow.py: adapt
    * tests/test_gui/test_progress.py: renamed from test_progress_bar.py;␣
↪adapt
          to new name; add status bar tests
    * doc/_source/_static/progress_done.png: added
    * doc/_source/_static/progress_going.png: added
    * doc/_source/codedoc/gui/progress.rst: added
    * doc/_source/codedoc/gui/index.rst: adapt and cleaup
    * doc/_source/gui/progress.rst: add progress and status bar info
    * doc/_source/gui/progress_bar.rst: renamed progress.rst
    * doc/_source/gui/index.rst: adapted

2016-05-24 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: fix the documentation; make all slots public
    * vdat/gui/treeview_model.py: fixed typos
    * doc/_source/gui/queue.rst: user documentation added
    * doc/_source/_static/queue_screenshot.png: add screenshot
    * doc/_source/_static/queue_tooltip.png: same
    * doc/_source/codedoc/gui/queue.rst: added
    * doc/_source/codedoc/gui/index.rst: add queue to the index
    * doc/_source/codedoc/gui/tree_view.rst: fix title

2016-05-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/progress_bar.py: remove command interpreter signals and mark
          functions as slots
    * vdat/gui/mainwidget.py: re-enable progress bar
    * vdat/gui/mainwindow.py: connect the queue signals to the progress bars
          and the status bar; the isolate the status bar in its own class
    * tests/test_gui/test_progress_bar.py: rewrite the tests

2016-05-20 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/command_interpreter/core.py: move CommandInterpreter._run into a
          standalone function to avoid pickling the instance when running in
          multiprocessor mode; get all the signals and attach them to␣
↪attributes,
          then use the attributes in the CommandInterpreter.run method
    * tests/test_ci/test_command_interpreter.py: adapt the tests
    * vdat/gui/queue.py: replace the VDATCommandWoker with a␣
↪QCommandInterpreter
          class, that create a QObject out of the command_interpreter and␣
↪implement
          the signals as pyqtSignals; adapt other parts of the code to those␣
↪changes
    * tests/test_gui/test_queue.py: adapt the tests
    * vdat/gui/__init__.py: don't connect to CommandInterpreter signals
    * vdat/gui/central.py: use QCommandInterpreter
    * vdat/gui/mainwindow.py: add a function to connect the signals emitted by
          the queue
    * vdat/libvdat/vdat.py: don't connect to the CommandInterpreter signals

2016-05-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move all the signal emissions to the
          main process
    * vdat/gui/queue.py: re-emit worker signals in the Queue class; mark old
          worker and thread for deletion on the next command execution
    * tests/test_gui/test_queue.py: adapt the tests

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: connect command interpreter signals with pyqt ones
    * vdat/gui/__init__.py: cleanup imports
    * vdat/command_interpreter/signals.py: fix documentation bugs
    * tests/test_gui/test_queue.py: test the new signals

2016-05-17 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/relay.py: removed
    * vdat/gui/mainwindow.py: set the queue here, so that it can be used when
          creating the queue action; remove all relay.py references
    * vdat/gui/__init__.py: remove the setting of the queue
    * vdat/gui/queue.py: remove relay and use already existing signals
    * tests/conftest.py: remove relays; normalize clear_* fixtures
    * tests/test_ci/conftest.py: use clear_* fixtures
    * tests/test_ci/test_signals.py: use clear_* fixtures
    * tests/test_gui/test_progress_bar.py: use clear_* fixtures
    * tests/test_gui/test_queue.py: finish testing the queue module

2016-05-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/queue.py: use a thread-worker approach
    * tests/test_gui/test_queue.py: test 89% of the queue module
    * doc/_source/codedoc/gui/index.rst: remove the background

2016-05-24 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        * vdat/gui/treeview_model.py: improve documentation and rename _
↪pressed slot
          to on_press
    * doc/_source/codedoc/database.rst: added
    * doc/_source/codedoc/gui/index.rst: modified
    * doc/_source/codedoc/gui/tree_view.rst: added
    * doc/_source/codedoc/index.rst: add database
    * doc/_source/conf.py: add pyqt4 intersphinx (but it apparently doesn't
          work)
    * doc/_source/launching.rst: fix link

2016-05-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: add cal_dir and zero_dir to the tooltip
        * tests/test_gui/test_tree_view.py: add it to the test
    * doc/_source/_static/tree_view_tooltip.png: update the screenshot
    * doc/_source/gui/tree_view.rst: finish the documentation

2016-05-21 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/tree_view_clone_dialog.png: add screenshot
    * doc/_source/_static/tree_view_clone_remove.png: add screenshot
    * doc/_source/_static/tree_view_remove_dialog.png: add screenshot
    * doc/_source/_static/tree_view_screenshot.png: add screenshot
    * doc/_source/_static/tree_view_tooltip.png: add screenshot
    * doc/_source/dirstruct.rst: added todo
    * doc/_source/gui/index.rst: add information
    * doc/_source/gui/menu_bar.rst: little change
    * doc/_source/gui/tree_view.rst: begin improving the documentation

2016-05-19 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/_static/vdat_screenshot.png: added
    * doc/_source/gui: all the gui related documentation moved here
    * doc/_source/gui/index.rst: short description and link to more detailed
          ones
    * doc/_source/gui/ifu_viewer.rst: added
    * doc/_source/gui/log_panel.rst: added
    * doc/_source/gui/main_panel.rst: added, all info about the main panel␣
↪moved
          here
    * doc/_source/gui/menu_bar.rst: added
    * doc/_source/gui/progress_bar.rst: added
    * doc/_source/gui/queue.rst: moved here
    * doc/_source/gui/tree_view.rst: added, all info about the tree view moved
          here
    * doc/_source/index.rst: updated to point to gui/index

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: show the right menu entry only when clicking
          on a valid item; resolves #1465
    * tests/test_gui/test_tree_view.py: test ReductionQTreeView.option_menu␣
```

```
↪slot

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: if no night found, return a model with only
          the root directory
    * tests/test_gui/test_tree_view.py: adapt tests and test the above case

2016-05-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: move the creation of the model and the
          initialization of the view into the ReductionQTreeView class
        * vdat/gui/mainwidget.py: use the ``ReductionQTreeView.set_model``␣
↪method to
          redo the symlink
    * tests/test_gui/test_tree_view.py: adapt the tests

2016-05-20  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/menubar.py: Added Quit menu, renamed symlink to file menu

2016-05-12 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: fix #1448

2016-05-11  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Reuse the updateTask, removing one of the
        fast click crash bugs

2016-05-11  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed bug occuring when switching
        fplanes fast

2016-05-11 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * vdat/gui/background.py: remove
    * tests/conftest.py: adapt
    * vdat/gui/__init__.py: same
    * vdat/gui/queue.py: cleanup
    * tests/test_gui/test_queue.py: added

2016-05-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: don't use background
    * vdat/gui/background.py: start phasing this out
    * vdat/gui/queue.py: add a run method and a first implementation of the
          thread to run the command

2016-05-09 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/gui/background.py: remove unused immediate thread
```

```
     * vdat/gui/mainwindow.py: same

2016-05-11 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.4.0rc3
    * vdat/libvdat/vdat.py: fix command line interface
    * doc/_source/launching.rst: fix documentation of the above

2016-05-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/fplane.txt: update the fplane file with the 9 IFUs
    * vdat/config/tasks.yml: update
    * vdat/config/vdat_commands.yml: update
    * vdat/gui/ifu_widget.py: add SPECID to the tooltip

2016-05-09 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwidget.py: fix symlinking in gui
    * vdat/gui/utils.py: add wait cursor context manager

2016-05-09  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Made updates more robust
        * vdat/gui/ifu_viewer.py: Add warning if no images selected to
        send to ds9
        * vdat/gui/ifu_widget.py: Make updates more robust

2016-05-09  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Only update ifus during cleanup, if they
        were initialized before

2016-05-06 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: update
    * vdat/config/vdat_commands.yml: add arguments to subtract sky
    * vdat/config/tasks.yml: same
    * vdat/gui/__init__.py: read progress report to console
    * vdat/gui/mainwidget.py: disable progress bar

2016-05-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: add virus instrument to the argument parser

2016-05-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove all astropy loggers
    * vdat/libvdat/loggers.py: explicitly pass configuration objects
    * vdat/libvdat/vdat.py: adapt
    * tests/test_libvdat/test_loggers.py: moved into test_libvdat;
          vdat.libvdat.loggers tested at 100%

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/gui/__init__.py: disconnect the indicator printing progress to␣
↪console
    * vdat/gui/progress_bar.py: set value to zero when setting it up
    * tests/test_gui/test_progress_bar.py: test the latter

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: make sure that x11 can deal with threads

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add n_primaries signal to
        documentation
    * vdat/gui/progress_bar.py: implement the progress bar and connect to
        command_interpreter signals
    * vdat/gui/mainwidget.py: use the object in progress_bar.py module
    * tests/test_ci/conftest.py: move clean_connected to test/conftest.py
    * tests/test_gui: created
    * tests/test_gui/test_progress_bar.py: added
    * tests/test_gui/test_tree_view.py: moved into test_gui
    * tests/test_ci/test_signals.py: make sure to clean the signals

2016-05-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: new n_primaries signal
    * vdat/command_interpreter/helpers.py: helper function for that
    * vdat/command_interpreter/core.py: emit the signal
    * tests/test_ci/test_helpers.py: update tests
    * tests/test_ci/test_signals.py: same

2016-05-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/extra_files/lines_000[LR].dat: removed
    * vdat/config/extra_files/lines_[LR].par: copied from cure
    * vdat/config/vdat_commands.yml: use combinearcs to create masterarcs; use
        the new line files in deformer, fix option
    * vdat/config/tasks.yml: adapt to the above changes; now I can run␣
↪deformer
        on most IFUs

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: update the configuration documentation
    * vdat/config/tasks.yml: add all the remaining reduction steps
    * vdat/config/vdat_setting.cfg: set the default pixel scale to 0.5, to␣
↪speed
        up the GUI startup
    * vdat/gui/fplane.py: don't raise an error if there are no tabs for a task

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwidget.py: cleanup old ways of communication
    * vdat/gui/mainwindow.py: same
    * vdat/gui/relay.py: remove unused signals
```

```
    * vdat/gui/treeview_model.py: cleanup and mark method a pyqt slot

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Change default viewer size for
        reconstructed images
        * vdat/gui/ifu_widget.py: Fix possible memory leak in creation of
        binned images

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: remove unused imports
    * vdat/gui/menubar.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/gui/fplane.py: use itertools.product for nested loops
    * vdat/gui/gui.py: removed
    * vdat/gui/ifu_viewer.py: PEP8
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/mainwidget.py: same
    * vdat/gui/tasks.py: same
    * doc/_source/codedoc/gui/index.rst: remove vdat.gui.gui

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Show reconstructed images
        * vdat/gui/central.py: Made pixelscale for reconstructed images␣
↪configurable
        * vdat/gui/ifu_widget.py: Show reconstructed images in ifu viewer
        fixes issues 1407 and 1409
        * vdat/config/vdat_setting.cfg: Added pixelscale for reconstructed␣
↪images

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/conf.py: don't check pyhetdex version and use pyhetdex/
↪latest
          for intersphinx

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * README.md: update
    * ReleaseNotes.md: add info about matplotlib removal
    * doc/_source/conf.py: remove matplotlib
    * doc/_source/install.rst: same
    * tests/conftest.py: same

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: get rid of APlpy and matplotlib
    * vdat/libvdat/vdat.py: same

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: add multiprocessing
```

```
2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: require pyhetdex 0.7.0
    * tests/test_config/test_core.py: adapt tests to the new config files
    * doc/_source/codedoc/gui/index.rst: fix typo

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: document dither tab type
    * vdat/config/tasks.yml: add all reduction science steps up to dividing by
         pixel flats
    * vdat/gui/fplane.py: update dither type implementation
    * vdat/gui/ifu_widget.py: fix bug #1405

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add cal reduction steps
    * vdat/gui/fplane.py: fix some bug with typ and fplane_single
    * vdat/gui/tasks.py: remove debugging prints
    * doc/_source/gui.rst: update documentation

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.3.0
    * ReleaseNotes.md: update

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: implement zro reduction steps
        * vdat/config/vdat_commands.yml: update commands to the latest␣
↪reduction
         steps
    * vdat/gui/central.py: fix indentation bug when submitting commands to
      queue; improve string sent to the queue
    * vdat/gui/fplane.py: improve FplaneWidget.change_fplane; add
      documentation, fix bug with matching directory names
    * vdat/gui/gui.py: make documentation happy
    * vdat/gui/tasks.py: accept commands as string or as list
    * vdat/gui/treeview_model.py: on selection changed pass the path of the
      directory
    * doc/_source/codedoc/gui/index.rst: move it from ../gui.rst, add
      placeholder table
    * doc/_source/codedoc/index.rst: adapt
    * doc/_source/codedoc/reduction.rst: fix module name
    * doc/_source/gui.rst: begin documentin tasks.yml file

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: remove debug print
    * vdat/gui/fplane.py: same
    * vdat/gui/ifu_widget.py: fix python3 bug

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/vdat_setting.cfg: remove tabs.yml and buttons.yml
    * vdat/config/core.py: update accordingly
    * vdat/config/entry_point.py: same
    * vdat/config/buttons.yml: remove
    * tests/test_buttons.py: same
    * vdat/config/tabs.yml: same
    * vdat/gui/buttons_menu.py: same

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: add multiprocessing arguments

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: added
    * vdat/config/vdat_setting.cfg: move max_delta_zro into the symlink␣
↪section
          and remove config_dirs
    * vdat/config/core.py: add config_dirs section when loading the config␣
↪file
    * vdat/libvdat/symlink.py: adapt to the above; add warning when no shot␣
↪file
          is found in a night
    * doc/_source/dirstruct.rst: document max_delta_zro
    * tests/test_config/test_core.py: adapt tests
    * tests/test_libvdat/test_symlink.py: add test for the above warning

2016-04-26 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: update documentation
    * doc/_source/launching.rst: same
    * vdat/config/vdat_setting.cfg: rename virus_dir to virus_instrument
    * tests/conftest.py: same
    * tests/test_libvdat/test_symlink.py: same
    * vdat/libvdat/symlink.py: update accordingly
    * vdat/libvdat/vdat.py: improve description

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: modify symlinking to comply to #1358
    * vdat/config/vdat_setting.cfg: add ``virus_dir`` entry to do the above
    * tests/data/raw/20151025/virus: add the virus directory to the test data
    * tests/conftest.py: adapt to the new directory structure
    * tests/test_ci/test_types.py: same
    * tests/test_libvdat/test_symlink.py: same

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add support for multiple raw or night␣
↪directories
    * tests/test_libvdat/test_symlink.py: adapt the tests

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/vdat.py: adjust docstring
    * tests/test_symlink.py: moved to tests/test_libvdat
    * tests/test_libvdat/test_vdat.py: added

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_loggers.py: add some more tests

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: fix bug in the command loggers
    * vdat/libvdat/loggers.py: clean up and fix few bugs
    * tests/test_loggers.py: add testing for the above modules

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: push variables to environment in a function,␣
↪improve
      command line arguments
    * vdat/config/vdat_setting.cfg: add is_rawdir_night option
    * vdat/config/core.py: skip also empty lists when overriding configuration
    * tests/test_config/test_core.py: test it

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: bugfix with ConfigParser file names; use mapping
          interface to insert a value
    * tests/test_config/test_core.py: test the vdat.config.core module
    * tests/conftest.py: adapt to changes in the vdat.config.core module

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: some function moved around, override_conf added
    * vdat/libvdat/vdat.py: first draft of the new command line interface
    * tests/conftest.py: add fixture to clear the internal configuration
          dictionary
    * tests/test_config: created
    * tests/test_config/test_core.py: added
    * tests/test_config.py: moved and renamed tests/test_config/test_entry_
↪point.py

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: I forgot to update the release notes

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: if a 100000 parameters in sql queries are␣
↪allowed,
          returns it, otherwise search for the number

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/database/core.py: remove SQLITE_MAX_COLUMN and add estimate of
          SQLITE_MAX_VARIABLE_NUMER
    * vdat/libvdat/symlink.py: use the latter when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: estimate SQLITE_MAX_COLUMN
    * vdat/libvdat/symlink.py: use the estimate when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: remove all the multiprocessing stuff and␣
↪improve
          log messages
    * vdat/libvdat/vdat.py: no multiprocessing things happening here
    * tests/test_symlink.py: update tests to the changes

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: make the types instance attribute
    * vdat/command_interpreter/types.py: fix __getattr__ to play nicely with
          pickling

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

     * tests/test_config.py: ignore .svn files
     * tests/conftest.py: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_ci/conftest.py: move here some useful fixture
    * tests/test_ci/test_signals.py: use the fixture
    * tests/test_ci/test_command_interpreter.py: test the core module to 99%
    * vdat/command_interpreter/core.py: if no file is collected return;␣
↪improve
          error from subprocess crashing

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/command_interpreter/core.py: make multiprocessing work
     * tests/test_ci/test_command_interpreter.py: add multiprocessing to the
          tests
     * setup.py: add pytest-xdist dependency for improved coverage
     * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/conf.py: use sphinx.ext.todo instead of
          pyhetdex.doc.sphinxext.tod
        * setup.py: bump sphinx version
        * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * tests/test_symlink.py: make sure that no error is raised when running␣
↪the
          symlink of science frames

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: improve testing the symlink and solve issue #1335
    * vdat/config/vdat_setting.cfg: improve regex to match also file names␣
↪with
          decimal seconds

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

        * setup.py: bump required pyhetdex version to 0.6.0
    * vdat/command_interpreter/core.py: use DeferredResult when running jobs␣
↪in
          single processor mode
    * tests/test_ci/test_command_interpreter.py: adapt the tests; test the
          filter_section keyword in action; some other little fix

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move the logging from _run to run
          methods
    * vdat/command_interpreter/exceptions.py: add CISubprocessError
    * tests/test_ci/test_command_interpreter.py: adapt the tests

2016-04-14 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex version to 0.5
    * vdat/command_interpreter/core.py: worker created and removed in
          CommandInterpreter.run method
    * vdat/libvdat/symlink.py: worker created and removed in do_symlink␣
↪function
    * vdat/libvdat/vdat.py: remove workers, as they are handled where they are
          used

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: get and report exceptions when running
          the command
    * ReleaseNotes.md: update
    * setup.py: bump version to 0.2.4

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.2.3-post
    * vdat/command_interpreter/types.py: fix bug that makes file collection
          fails when using regex and directory names containing special␣
↪characters

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

        * ReleaseNotes.md: added; track history and help writing the report␣
```

```
↪for the
          next release

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: renamed, update documentation with
          info about multiprocessing
    * doc/_source/conf.py: use only pyhetdex latest for intersphinx
    * doc/_source/dirstruct.rst: add info about multiprocessing
    * doc/_source/executables.rst: expand a bit
    * doc/_source/launching.rst: same
    * doc/_source/index.rst: reorder some section
    * vdat/command_interpreter/core.py: try to enable multiprocessing, fail
          miserably
    * vdat/gui/buttons_menu.py: pass multiprocessing keywords
    * vdat/libvdat/vdat.py: close also command_interpreter worker

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat: rebased on ^/trunk
    * setup.py: version 0.2.3-post
    * tox.ini: force DISPLAY=:0

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: workaround bug with too many multiple␣
↪insertions; #1345
    * vdat/gui/gui.py: brute force implementation of re-symlink from gui
↪#1333;
          works, but needs review

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: negative error codes exists and are
          failures; fix the bug

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: don't close the multiprocessing worker to allow
          reusing it; set non existing rawdir to empty string; make public two
          functions that might be used from further symlinking
        * vdat/libvdat/vdat.py: wait and close the symlink worker
    * tests/test_symlink.py: update the tests

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged: branches/multiple_objects@169
        * setup.py: version set to 0.2.2
    * vdat/command_interpreter/core.py: log also the target dir then starting␣
↪a
          task

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/symlink.py: properly symlink science frames with empty␣
↪OBJECT
          fields and add counter to repeated OBJECT from different shots
    * tests/test_symlink.py: change test function name. The above changes has
          been tested only by hand
    * doc/_source/dirstruct.rst: add info about this

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * vdat/gui/treeview_model.py: add tool tip
    * tests/test_tree_view.py: test it
    * doc/_source/gui.rst: add some info about tooltip and right-click␣
↪commands
          available on the tree view

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: advance version
    * vdat/libvdat/symlink.py: fix multiprocessing while symlinking, now it
          works
    * vdat/libvdat/vdat.py: little changes because of the above

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: improve

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add references to zero and cal directories to
          every type

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: add version to command line
    * vdat/libvdat/vdat.py: same
    * vdat/gui/gui.py: add version to "About VDAT" menu; add links to
          documentation
    * vdat/gui/menu.py: pep8

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: version 0.2.1-pre
    * vdat/command_interpreter/types.py: extend the header type to allow
          formatting the values
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: write info log when a command start
          running
    * vdat/command_interpreter/types.py: add ``returns`` option to plain␣
↪primary
```

```
          type
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: create a general section and use it in
          every command; fix biassubtract fits matching to skip thumbnail fits

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: different shots with multiple lamps are now
          collected in the same cal directory
    * vdat/config/vdat_setting.cfg: add config tell the symlinking which␣
↪header
          keyword has the name of the lamps

2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve error message when regex␣
↪match
          does not work; fix bug with header type and multi-word header␣
↪keyword
          parsing
    * tests/test_ci/test_types.py: test this

2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: fix couple of bugs and remove copied files␣
↪if
          cloning fails
    * tests/test_tree_view.py: test 97% of the treeview_model.py (I don't␣
↪think
          I can do more)

2016-03-31 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_tree_view.py: test checkboxes also from the tree view
          perspective

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/symlink on top of trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * merge branches/cmd_interpreter_update into trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/cmd_interpreter_update on to of trunk

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/vdat.py: connect the global logger to the VDAT main logger
    * vdat/command_interpreter/core.py: fix typo

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: Create a global logger signal
    * vdat/command_interpreter/helpers.py: move the old default implementation
          here
    * vdat/command_interpreter/core.py: use the new global logger
    * tests/test_ci/test_signals.py: test the global logger
    * tests/test_ci/test_helpers.py: same

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: update documentation

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: connect some signal in order to have some␣
↪execution
          feedback

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: select directory also by enter key
    * tests/test_tree_view.py: test it (it's a workaround for the fact that
          there is a bug about testing clicks on tree view)

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: remove ``row`` from ReductionNode, as it's␣
↪not
          used
    * tests/test_tree_view.py: same

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: use pytest-catchlog instead of pytest-capturelog
    * tox.ini: same
    * tests/test_command_interpreter.py: adapt to the change
    * tests/test_tree_view.py: same
    * tests/test_symlink.py: same; do the symlinking in the test function, not
          setup

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: bugfix
    * tests/test_tree_view.py: mark old tests as integration, unit-test 46%␣
↪of the code

2016-03-17 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/database/models.py: add ``path`` attribute to the␣
↪VDATExposures table
```

```
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/gui/treeview_model.py: correctly propagate VDATExposures␣
↪information
          when cloning and removing directories; fixed bad bug in check/
↪uncheck
    * tests/test_tree_view.py: rewrite tests for the tree view, 40% done

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Renamed Raw to Exp in Tab descriptions

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add CICommandDone signal
    * vdat/command_interpreter/core.py: use it
    * vdat/command_interpreter/helpers.py: add a receiver that prints out␣
↪stuff
    * tests/test_ci/test_signals.py: test the new signal
    * tests/test_ci/test_helpers.py: test the new helper

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged with ^/trunk
        * tests/test_ci/test_types.py: update number of files

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: add @path@ to the @new_file@ type
    * tests/test_ci/test_types.py: add the tests
    * doc/_source/command_intepreter.rst: document it

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/libvdat/vdat.py: Moved first import of gui till after the
        XPA_METHOD was set based on config

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

        * merged ^/trunk

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove [args] section
    * vdat/database/models.py: remove ThumbnailScaling table
    * vdat/database/core.py: adapt
    * vdat/libvdat/callback.py: removed, as is unused
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/show_fits.py: same
    * doc/_source/codedoc/reduction.rst: remove callback

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/symlink.py: save the exposure database on files to be able␣
↪to
          rebuild the database from already symlinked directories
    * vdat/utilities.py: add utility function for the exposure database dump
    * vdat/database/base.py: use public functions to get the data from the
          model; provides 3 properties to get some of the data
    * vdat/database/models.py: override the data_clean property to skip the
          foreign fields
    * tests/test_symlink.py: adjust the tests to the changes, test the new␣
↪parts
    * tests/test_tree_view.py: little adjustment

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: update the new type names
    * vdat/libvdat/symlink.py: adapt the vdatexposures names
    * vdat/gui/buttons_menu.py: show the empty widget if no button is defined
          for the type
    * vdat/gui/treeview_model.py: get the type names from the database

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: adjust log messages about the type of the
          symlinked shot

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: proper cleanup when the symlinking fails, small
          adjustments.
    * tests/conftest.py: add virus*** related fixtures
    * tests/test_symlink.py: test to 100% the symlinking

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: export the database to the package level
    * vdat/libvdat/symlink.py: make the insertion in the database and the
          symlinking more robust to failures
    * vdat/utilities.py: add new error

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk
    * tests/test_symlink.py: adapt the tests

2016-03-07 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/symlink.py: fix bug in @_find_nearest@, remove optional
          argument from @symlink@ function
    * pytest.ini: add marker for integration tests
    * tests/conftest.py: add night and virus00001 fixtures
    * tests/test_symlink.py: add some unit tests

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/symlink.py: allow unknown/nonstandard object type linking
    * vdat/config/vdat_setting.cfg: add little explanation about it
    * doc/_source/dirstruct.rst: document it

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-03-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: first part of the new_file type
          implementation
    * tests/test_ci/test_types.py: test the new_file type
    * vdat/command_interpreter/core.py: adapt to the above
    * tests/test_ci/test_command_interpreter.py: same
    * doc/_source/command_intepreter.rst: adjust documentation
        * vdat/config/vdat_commands.yml: deformer 4 is no more

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/__init__.py: import get_signal and
          get_signal_names at the module level
    * vdat/command_interpreter/types.py: fix documentation
    * vdat/command_interpreter/utils.py: fix documentation
    * tests/test_ci/test_utils.py: add test of id_
    * doc/_source/codedoc/command_interpreter/types.rst: fix documentation

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: set xpa_method to local to avoid test hanging
    * vdat/command_interpreter/helpers.py: remove __all__
    * vdat/command_interpreter/signals.py: same
    * vdat/command_interpreter/types.py: import numpy
    * doc/_source/codedoc/command_interpreter/index.rst: split the command
          interpreter documentation
    * doc/_source/codedoc/command_interpreter/*.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/relay.py: renambed signals
    * vdat/command_interpreter/signals.py: rewritten to have an behaviour
          similar to qt signals; CILogger not reimplemented; some signal␣
↪missing
    * vdat/command_interpreter/__init__.py: remove the import of the relay
    * vdat/command_interpreter/core.py: update according to the above changes
    * vdat/command_interpreter/helpers.py: provide some function that can be
          plugged in
```

```
    * tests/test_ci/test_helpers.py: added
    * tests/test_ci/test_signals.py: added
    * doc/_source/command_intepreter.rst: relays -> signals
    * doc/_source/codedoc/command_interpreter.rst: moved to
          command_interpreter/index.rst
    * doc/_source/codedoc/index.rst: index in the codedoc to allow reshaping␣
↪of
          the documentation
    * doc/_source/index.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add a method that returns a range to
          SliceLike
    * vdat/command_interpreter/types.py: use SliceLike in primary_loop; remove
          _to_number function
    * tests/test_ci/test_utils.py: test the range method
    * tests/test_config.py: fixture to fix seed of random for repeatability of
          tests

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: adapt types to use the new
          keyword_regex and do_split = False
    * tests/test_ci/test_types.py: add tests for all the secondary keywords
    * doc/_source/command_intepreter.rst: fix the documentation

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add string and format customization␣
↪to
          SliceLike
    * tests/test_ci/test_utils.py: test it

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

        * merge trunk

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve the keyword_regex␣
↪functionality
    * vdat/command_interpreter/utils.py: homogenize doc
    * vdat/command_interpreter/core.py: move back types to class properties␣
↪to be
          able to run in parallel (this needs investigation)
    * vdat/command_interpreter/exceptions.py: fix typo
    * tests/test_ci/test_types.py: test the keyword_regex functionality
    * doc/_source/command_intepreter.rst: document the new keyword_regex

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: move some fixture to session scope
    * vdat/command_interpreter/core.py: move the types initialisation into the
```

```
          __init__

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add SliceLike and copy some utils␣
↪from
      types.py
    * vdat/command_interpreter/exceptions.py: import the future and add
      CISliceError
    * setup.cfg: add doctest
    * tests/test_ci/test_utils.py: added
        * tests/test_ci/test_command_interpreter.py: moved
    * tests/test_ci/test_types.py: added and partially implemented
    * doc/_source/codedoc/command_interpreter.rst: add types and utils
      documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
          in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
↪widget.py
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table
```

```
2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report
```

```
2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking

        use ``vdat_config copy`` to update the configuration files


2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
```

```
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
          button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
```

```
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
          to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
          logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
          functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
          failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                   a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>

      * requirements.txt: added pyds9 repo
```

```
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
          changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
          necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
          initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
                          uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
↪avoid an
```

```
                                astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
↪thread.
```

```
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
                                the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
                              plane panels. What is displayed as a thumbnail
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                           rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                      selecting a node causes a whole new
                                      central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                     with a regex specifying the file type to
                                     display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging
```

```
2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list
```

```
2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same
```

```
2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
```

```
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
```

```
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
```

```
                                try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                           select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here

2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
```

```
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
```

```
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
```

```
↪treeview widget to
                                        use. It can show different reduction␣
↪steps in a
                                        branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
       * vdat/gui/static/unreduced.png: New image to differentiate
                                        between IFUs with and without input␣
↪files
       * vdat/libvdat/background.py: Uses the queue
       * vdat/libvdat/callback.py: Uses the queue
       * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
       * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

       Updated documentation and installation files:
       * doc/codedoc/gui.rst
       * doc/codedoc/reduction.rst
       * doc/index.rst
       * doc/queue.rst
       * requirements.txt
       * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

        * MANIFEST.in: Added fplane.txt file, so it is also installed!
        * doc/install.rst: Tweaked documentation
        * doc/launching.rst: As above
        * requirements.txt: Added command to install pyhetdex
        * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added Sphinx documentation (under doc/), minor
        modifications to comments
```

```
        * AUTHORS
        * LICENSE
        * README.md: Added new dependencies
        * doc/: Added documentation here
        * vdat/gui/gui.py
        * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

        *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
        *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
        *  vdat/gui/gui.py:    Added a create masterbias button
        *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
        *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
        *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
        *  vdat/libvdat/symlink.py
        *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure



2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
                                                  to pyhetdex:r74
```

```
2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
       * vdat/libvdat/background.py
       * vdat/libvdat/callback.py
       * vdat/libvdat/database.py
       * vdat/libvdat/fits.py
       * vdat/libvdat/handlers.py
       * vdat/libvdat/reduction.py: Added function to subtract overscans
       * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
       * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

     Added an internal sqlite3 database to keep track
     of what files are available. Created a background
     thread with which to run things so they don't lock
     up the GUI when they're running. Implemented
     a simple code which loops through all fits files
     and converts them to PNGs.

     * vdat/gui/__init__.py: Moved call to symlink to here
     * vdat/gui/gui.py:    Added a (currently disabled) progress bar
     * vdat/libvdat/background.py: run jobs in a separate thread
     * vdat/libvdat/callback.py: Added calls to Background
     * vdat/libvdat/database.py: Internal database to keep track of files
     * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
     * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
     * vdat/libvdat/symlink.py: Can read rawdir from config file
     * vdat/libvdat/vdat.py:  Moved symlink from here.
```

```
2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging


2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
```

```
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
          in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info
```

```
2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
↪widget.py
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
```

```
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report

2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking

        use ``vdat_config copy`` to update the configuration files


2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
```

```
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
```

```
        button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
        correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
        default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
        to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
        logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
        functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
        failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                  a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>
```

```
      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
          changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
          necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
          initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
                          uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
```

```
                                  file object with astropy getdata in order to␣
↪avoid an
                                  astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
```

```
                                    flag is controlled by the Worker in the␣
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                                QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                              for running jobs on the immediate thread to end.␣
↪This
                              stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                    of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                       thumbnails for the IFU widget with␣
↪entries
                                       for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                  the different node types (on the same level as
                                  the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
                                plane panels. What is displayed as a thumbnail
                                is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                                images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                              rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                        selecting a node causes a whole new
                                        central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                       with a regex specifying the file type to
                                       display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
```

```
    * doc/_source/index.rst: add todo about logging

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
```

```
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
```

```
    * vdat/libvdat/symlink.py: same

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
```

```
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
```

```
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
```

```
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
                              try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                          select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here

2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
```

```
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
```

```
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
```

```
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
                                         use. It can show different reduction␣
↪steps in a
                                         branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
       * vdat/gui/static/unreduced.png: New image to differentiate
                                        between IFUs with and without input␣
↪files
       * vdat/libvdat/background.py: Uses the queue
       * vdat/libvdat/callback.py: Uses the queue
       * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
       * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

       Updated documentation and installation files:
       * doc/codedoc/gui.rst
       * doc/codedoc/reduction.rst
       * doc/index.rst
       * doc/queue.rst
       * requirements.txt
       * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

        * MANIFEST.in: Added fplane.txt file, so it is also installed!
        * doc/install.rst: Tweaked documentation
        * doc/launching.rst: As above
        * requirements.txt: Added command to install pyhetdex
        * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added Sphinx documentation (under doc/), minor
```

```
        modifications to comments

        * AUTHORS
        * LICENSE
        * README.md: Added new dependencies
        * doc/: Added documentation here
        * vdat/gui/gui.py
        * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

        *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
        *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
        *  vdat/gui/gui.py:    Added a create masterbias button
        *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
        *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
        *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
        *  vdat/libvdat/symlink.py
        *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure



2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
```

```
↪changed wildcards to conform
                                                to pyhetdex:r74



2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
       * vdat/libvdat/background.py
       * vdat/libvdat/callback.py
       * vdat/libvdat/database.py
       * vdat/libvdat/fits.py
       * vdat/libvdat/handlers.py
       * vdat/libvdat/reduction.py: Added function to subtract overscans
       * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
       * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

     Added an internal sqlite3 database to keep track
     of what files are available. Created a background
     thread with which to run things so they don't lock
     up the GUI when they're running. Implemented
     a simple code which loops through all fits files
     and converts them to PNGs.

     * vdat/gui/__init__.py: Moved call to symlink to here
     * vdat/gui/gui.py:    Added a (currently disabled) progress bar
     * vdat/libvdat/background.py: run jobs in a separate thread
     * vdat/libvdat/callback.py: Added calls to Background
     * vdat/libvdat/database.py: Internal database to keep track of files
     * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
     * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
```

```
     * vdat/libvdat/symlink.py: Can read rawdir from config file
     * vdat/libvdat/vdat.py:  Moved symlink from here.


2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging


2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
```

```
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: make sure that x11 can deal with threads

2016-05-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add n_primaries signal to
          documentation
    * vdat/gui/progress_bar.py: implement the progress bar and connect to
          command_interpreter signals
    * vdat/gui/mainwidget.py: use the object in progress_bar.py module
    * tests/test_ci/conftest.py: move clean_connected to test/conftest.py
    * tests/test_gui: created
    * tests/test_gui/test_progress_bar.py: added
    * tests/test_gui/test_tree_view.py: moved into test_gui
    * tests/test_ci/test_signals.py: make sure to clean the signals

2016-05-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: new n_primaries signal
    * vdat/command_interpreter/helpers.py: helper function for that
    * vdat/command_interpreter/core.py: emit the signal
    * tests/test_ci/test_helpers.py: update tests
    * tests/test_ci/test_signals.py: same

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: update the configuration documentation
    * vdat/config/tasks.yml: add all the remaining reduction steps
    * vdat/config/vdat_setting.cfg: set the default pixel scale to 0.5, to␣
↪speed
          up the GUI startup
    * vdat/gui/fplane.py: don't raise an error if there are no tabs for a task

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/mainwidget.py: cleanup old ways of communication
    * vdat/gui/mainwindow.py: same
    * vdat/gui/relay.py: remove unused signals
    * vdat/gui/treeview_model.py: cleanup and mark method a pyqt slot

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Change default viewer size for
        reconstructed images
```

```
        * vdat/gui/ifu_widget.py: Fix possible memory leak in creation of
        binned images

2016-05-02 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: remove unused imports
    * vdat/gui/menubar.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/gui/fplane.py: use itertools.product for nested loops
    * vdat/gui/gui.py: removed
    * vdat/gui/ifu_viewer.py: PEP8
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/mainwidget.py: same
    * vdat/gui/tasks.py: same
    * doc/_source/codedoc/gui/index.rst: remove vdat.gui.gui

2016-05-02  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Show reconstructed images
        * vdat/gui/central.py: Made pixelscale for reconstructed images␣
↪configurable
        * vdat/gui/ifu_widget.py: Show reconstructed images in ifu viewer
        fixes issues 1407 and 1409
        * vdat/config/vdat_setting.cfg: Added pixelscale for reconstructed␣
↪images

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: add multiprocessing

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: require pyhetdex 0.7.0
    * tests/test_config/test_core.py: adapt tests to the new config files
    * doc/_source/codedoc/gui/index.rst: fix typo

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/gui.rst: document dither tab type
    * vdat/config/tasks.yml: add all reduction science steps up to dividing by
          pixel flats
    * vdat/gui/fplane.py: update dither type implementation
    * vdat/gui/ifu_widget.py: fix bug #1405

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: add cal reduction steps
    * vdat/gui/fplane.py: fix some bug with typ and fplane_single
    * vdat/gui/tasks.py: remove debugging prints
    * doc/_source/gui.rst: update documentation

2016-04-30 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.3.0
```

```
    * ReleaseNotes.md: update

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/tasks.yml: implement zro reduction steps
        * vdat/config/vdat_commands.yml: update commands to the latest␣
↪reduction
          steps
    * vdat/gui/central.py: fix indentation bug when submitting commands to
      queue; improve string sent to the queue
    * vdat/gui/fplane.py: improve FplaneWidget.change_fplane; add
      documentation, fix bug with matching directory names
    * vdat/gui/gui.py: make documentation happy
    * vdat/gui/tasks.py: accept commands as string or as list
    * vdat/gui/treeview_model.py: on selection changed pass the path of the
      directory
    * doc/_source/codedoc/gui/index.rst: move it from ../gui.rst, add
      placeholder table
    * doc/_source/codedoc/index.rst: adapt
    * doc/_source/codedoc/reduction.rst: fix module name
    * doc/_source/gui.rst: begin documentin tasks.yml file

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/central.py: remove debug print
    * vdat/gui/fplane.py: same
    * vdat/gui/ifu_widget.py: fix python3 bug

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove tabs.yml and buttons.yml
    * vdat/config/core.py: update accordingly
    * vdat/config/entry_point.py: same
    * vdat/config/buttons.yml: remove
    * tests/test_buttons.py: same
    * vdat/config/tabs.yml: same
    * vdat/gui/buttons_menu.py: same

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: add multiprocessing arguments

2016-04-27 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: added
    * vdat/config/vdat_setting.cfg: move max_delta_zro into the symlink␣
↪section
          and remove config_dirs
    * vdat/config/core.py: add config_dirs section when loading the config␣
↪file
    * vdat/libvdat/symlink.py: adapt to the above; add warning when no shot␣
↪file
          is found in a night
    * doc/_source/dirstruct.rst: document max_delta_zro
```

```
    * tests/test_config/test_core.py: adapt tests
    * tests/test_libvdat/test_symlink.py: add test for the above warning

2016-04-26 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/dirstruct.rst: update documentation
    * doc/_source/launching.rst: same
    * vdat/config/vdat_setting.cfg: rename virus_dir to virus_instrument
    * tests/conftest.py: same
    * tests/test_libvdat/test_symlink.py: same
    * vdat/libvdat/symlink.py: update accordingly
    * vdat/libvdat/vdat.py: improve description

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: modify symlinking to comply to #1358
    * vdat/config/vdat_setting.cfg: add ``virus_dir`` entry to do the above
    * tests/data/raw/20151025/virus: add the virus directory to the test data
    * tests/conftest.py: adapt to the new directory structure
    * tests/test_ci/test_types.py: same
    * tests/test_libvdat/test_symlink.py: same

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add support for multiple raw or night␣
↪directories
    * tests/test_libvdat/test_symlink.py: adapt the tests

2016-04-25 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: adjust docstring
    * tests/test_symlink.py: moved to tests/test_libvdat
    * tests/test_libvdat/test_vdat.py: added

2016-04-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_loggers.py: add some more tests

2016-04-28 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: fix bug in the command loggers
    * vdat/libvdat/loggers.py: clean up and fix few bugs
    * tests/test_loggers.py: add testing for the above modules

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: push variables to environment in a function,␣
↪improve
      command line arguments
    * vdat/config/vdat_setting.cfg: add is_rawdir_night option
    * vdat/config/core.py: skip also empty lists when overriding configuration
    * tests/test_config/test_core.py: test it

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/core.py: bugfix with ConfigParser file names; use mapping
          interface to insert a value
    * tests/test_config/test_core.py: test the vdat.config.core module
    * tests/conftest.py: adapt to changes in the vdat.config.core module

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: some function moved around, override_conf added
    * vdat/libvdat/vdat.py: first draft of the new command line interface
    * tests/conftest.py: add fixture to clear the internal configuration
          dictionary
    * tests/test_config: created
    * tests/test_config/test_core.py: added
    * tests/test_config.py: moved and renamed tests/test_config/test_entry_
↪point.py

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * ReleaseNotes.md: I forgot to update the release notes

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: if a 100000 parameters in sql queries are␣
↪allowed,
          returns it, otherwise search for the number

2016-04-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: remove SQLITE_MAX_COLUMN and add estimate of
          SQLITE_MAX_VARIABLE_NUMER
    * vdat/libvdat/symlink.py: use the latter when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/core.py: estimate SQLITE_MAX_COLUMN
    * vdat/libvdat/symlink.py: use the estimate when doing a bulk insert

2016-04-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: remove all the multiprocessing stuff and␣
↪improve
          log messages
    * vdat/libvdat/vdat.py: no multiprocessing things happening here
    * tests/test_symlink.py: update tests to the changes

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: make the types instance attribute
    * vdat/command_interpreter/types.py: fix __getattr__ to play nicely with
          pickling

2016-04-20 Francesco Montesano <montefra@mpe.mpg.de>
```

```
     * tests/test_config.py: ignore .svn files
     * tests/conftest.py: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_ci/conftest.py: move here some useful fixture
    * tests/test_ci/test_signals.py: use the fixture
    * tests/test_ci/test_command_interpreter.py: test the core module to 99%
    * vdat/command_interpreter/core.py: if no file is collected return;␣
↪improve
          error from subprocess crashing

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

     * vdat/command_interpreter/core.py: make multiprocessing work
     * tests/test_ci/test_command_interpreter.py: add multiprocessing to the
          tests
     * setup.py: add pytest-xdist dependency for improved coverage
     * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/conf.py: use sphinx.ext.todo instead of
          pyhetdex.doc.sphinxext.tod
        * setup.py: bump sphinx version
        * tox.ini: same

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: make sure that no error is raised when running␣
↪the
          symlink of science frames

2016-04-19 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_symlink.py: improve testing the symlink and solve issue #1335
    * vdat/config/vdat_setting.cfg: improve regex to match also file names␣
↪with
          decimal seconds

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

        * setup.py: bump required pyhetdex version to 0.6.0
    * vdat/command_interpreter/core.py: use DeferredResult when running jobs␣
↪in
          single processor mode
    * tests/test_ci/test_command_interpreter.py: adapt the tests; test the
          filter_section keyword in action; some other little fix

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: move the logging from _run to run
          methods
    * vdat/command_interpreter/exceptions.py: add CISubprocessError
```

```
    * tests/test_ci/test_command_interpreter.py: adapt the tests

2016-04-14 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex version to 0.5
    * vdat/command_interpreter/core.py: worker created and removed in
          CommandInterpreter.run method
    * vdat/libvdat/symlink.py: worker created and removed in do_symlink␣
↪function
    * vdat/libvdat/vdat.py: remove workers, as they are handled where they are
          used

2016-04-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: get and report exceptions when running
          the command
    * ReleaseNotes.md: update
    * setup.py: bump version to 0.2.4

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version to 0.2.3-post
    * vdat/command_interpreter/types.py: fix bug that makes file collection
          fails when using regex and directory names containing special␣
↪characters

2016-04-15 Francesco Montesano <montefra@mpe.mpg.de>

        * ReleaseNotes.md: added; track history and help writing the report␣
↪for the
          next release

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_interpreter.rst: renamed, update documentation with
          info about multiprocessing
    * doc/_source/conf.py: use only pyhetdex latest for intersphinx
    * doc/_source/dirstruct.rst: add info about multiprocessing
    * doc/_source/executables.rst: expand a bit
    * doc/_source/launching.rst: same
    * doc/_source/index.rst: reorder some section
    * vdat/command_interpreter/core.py: try to enable multiprocessing, fail
          miserably
    * vdat/gui/buttons_menu.py: pass multiprocessing keywords
    * vdat/libvdat/vdat.py: close also command_interpreter worker

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat: rebased on ^/trunk
    * setup.py: version 0.2.3-post
    * tox.ini: force DISPLAY=:0

2016-04-11 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/libvdat/symlink.py: workaround bug with too many multiple␣
↪insertions; #1345
    * vdat/gui/gui.py: brute force implementation of re-symlink from gui
↪#1333;
          works, but needs review

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: negative error codes exists and are
          failures; fix the bug

2016-04-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: don't close the multiprocessing worker to allow
          reusing it; set non existing rawdir to empty string; make public two
          functions that might be used from further symlinking
        * vdat/libvdat/vdat.py: wait and close the symlink worker
    * tests/test_symlink.py: update the tests

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged: branches/multiple_objects@169
        * setup.py: version set to 0.2.2
    * vdat/command_interpreter/core.py: log also the target dir then starting␣
↪a
          task

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: properly symlink science frames with empty␣
↪OBJECT
          fields and add counter to repeated OBJECT from different shots
    * tests/test_symlink.py: change test function name. The above changes has
          been tested only by hand
    * doc/_source/dirstruct.rst: add info about this

2016-04-08 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump version
    * vdat/gui/treeview_model.py: add tool tip
    * tests/test_tree_view.py: test it
    * doc/_source/gui.rst: add some info about tooltip and right-click␣
↪commands
          available on the tree view

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: advance version
    * vdat/libvdat/symlink.py: fix multiprocessing while symlinking, now it
          works
    * vdat/libvdat/vdat.py: little changes because of the above

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * doc/_source/dirstruct.rst: improve

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: add references to zero and cal directories to
          every type

2016-04-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: add version to command line
    * vdat/libvdat/vdat.py: same
    * vdat/gui/gui.py: add version to "About VDAT" menu; add links to
          documentation
    * vdat/gui/menu.py: pep8

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: version 0.2.1-pre
    * vdat/command_interpreter/types.py: extend the header type to allow
          formatting the values
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-06 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/core.py: write info log when a command start
          running
    * vdat/command_interpreter/types.py: add ``returns`` option to plain␣
↪primary
          type
    * tests/test_ci/test_types.py: test it
    * doc/_source/command_intepreter.rst: document it

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: create a general section and use it in
          every command; fix biassubtract fits matching to skip thumbnail fits

2016-04-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: different shots with multiple lamps are now
          collected in the same cal directory
    * vdat/config/vdat_setting.cfg: add config tell the symlinking which␣
↪header
          keyword has the name of the lamps

2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve error message when regex␣
↪match
          does not work; fix bug with header type and multi-word header␣
↪keyword
          parsing
    * tests/test_ci/test_types.py: test this
```

```
2016-04-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: fix couple of bugs and remove copied files␣
↪if
          cloning fails
    * tests/test_tree_view.py: test 97% of the treeview_model.py (I don't␣
↪think
          I can do more)

2016-03-31 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_tree_view.py: test checkboxes also from the tree view
          perspective

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/symlink on top of trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * merge branches/cmd_interpreter_update into trunk

2016-03-30 Francesco Montesano <montefra@mpe.mpg.de>

        * rebase branches/cmd_interpreter_update on to of trunk

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: connect the global logger to the VDAT main logger
    * vdat/command_interpreter/core.py: fix typo

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: Create a global logger signal
    * vdat/command_interpreter/helpers.py: move the old default implementation
          here
    * vdat/command_interpreter/core.py: use the new global logger
    * tests/test_ci/test_signals.py: test the global logger
    * tests/test_ci/test_helpers.py: same

2016-03-29 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: update documentation

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: connect some signal in order to have some␣
↪execution
          feedback

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: select directory also by enter key
```

```
    * tests/test_tree_view.py: test it (it's a workaround for the fact that
          there is a bug about testing clicks on tree view)

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: remove ``row`` from ReductionNode, as it's␣
↪not
          used
    * tests/test_tree_view.py: same

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: use pytest-catchlog instead of pytest-capturelog
    * tox.ini: same
    * tests/test_command_interpreter.py: adapt to the change
    * tests/test_tree_view.py: same
    * tests/test_symlink.py: same; do the symlinking in the test function, not
          setup

2016-03-21 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: bugfix
    * tests/test_tree_view.py: mark old tests as integration, unit-test 46%␣
↪of the code

2016-03-17 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/database/models.py: add ``path`` attribute to the␣
↪VDATExposures table
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/gui/treeview_model.py: correctly propagate VDATExposures␣
↪information
          when cloning and removing directories; fixed bad bug in check/
↪uncheck
    * tests/test_tree_view.py: rewrite tests for the tree view, 40% done

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/fplane.py: Renamed Raw to Exp in Tab descriptions

2016-03-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/signals.py: add CICommandDone signal
    * vdat/command_interpreter/core.py: use it
    * vdat/command_interpreter/helpers.py: add a receiver that prints out␣
↪stuff
    * tests/test_ci/test_signals.py: test the new signal
    * tests/test_ci/test_helpers.py: test the new helper

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

        * merged with ^/trunk
        * tests/test_ci/test_types.py: update number of files
```

```
2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: add @path@ to the @new_file@ type
    * tests/test_ci/test_types.py: add the tests
    * doc/_source/command_intepreter.rst: document it

2016-03-16  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/libvdat/vdat.py: Moved first import of gui till after the
        XPA_METHOD was set based on config

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

        * merged ^/trunk

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_setting.cfg: remove [args] section
    * vdat/database/models.py: remove ThumbnailScaling table
    * vdat/database/core.py: adapt
    * vdat/libvdat/callback.py: removed, as is unused
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/show_fits.py: same
    * doc/_source/codedoc/reduction.rst: remove callback

2016-03-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: save the exposure database on files to be able␣
↪to
          rebuild the database from already symlinked directories
    * vdat/utilities.py: add utility function for the exposure database dump
    * vdat/database/base.py: use public functions to get the data from the
          model; provides 3 properties to get some of the data
    * vdat/database/models.py: override the data_clean property to skip the
          foreign fields
    * tests/test_symlink.py: adjust the tests to the changes, test the new␣
↪parts
    * tests/test_tree_view.py: little adjustment

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/fplane.py: update the new type names
    * vdat/libvdat/symlink.py: adapt the vdatexposures names
    * vdat/gui/buttons_menu.py: show the empty widget if no button is defined
          for the type
    * vdat/gui/treeview_model.py: get the type names from the database

2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: adjust log messages about the type of the
          symlinked shot
```

```
2016-03-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: proper cleanup when the symlinking fails, small
          adjustments.
    * tests/conftest.py: add virus*** related fixtures
    * tests/test_symlink.py: test to 100% the symlinking

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: export the database to the package level
    * vdat/libvdat/symlink.py: make the insertion in the database and the
          symlinking more robust to failures
    * vdat/utilities.py: add new error

2016-03-08 Francesco Montesano <montefra@mpe.mpg.de>

    * merge ^/trunk
    * tests/test_symlink.py: adapt the tests

2016-03-07 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/libvdat/symlink.py: fix bug in @_find_nearest@, remove optional
          argument from @symlink@ function
    * pytest.ini: add marker for integration tests
    * tests/conftest.py: add night and virus00001 fixtures
    * tests/test_symlink.py: add some unit tests

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/symlink.py: allow unknown/nonstandard object type linking
    * vdat/config/vdat_setting.cfg: add little explanation about it
    * doc/_source/dirstruct.rst: document it

2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-03-04 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: first part of the new_file type
          implementation
    * tests/test_ci/test_types.py: test the new_file type
    * vdat/command_interpreter/core.py: adapt to the above
    * tests/test_ci/test_command_interpreter.py: same
    * doc/_source/command_intepreter.rst: adjust documentation
        * vdat/config/vdat_commands.yml: deformer 4 is no more
```

```
2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/__init__.py: import get_signal and
          get_signal_names at the module level
    * vdat/command_interpreter/types.py: fix documentation
    * vdat/command_interpreter/utils.py: fix documentation
    * tests/test_ci/test_utils.py: add test of id_
    * doc/_source/codedoc/command_interpreter/types.rst: fix documentation

2016-02-29 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: set xpa_method to local to avoid test hanging
    * vdat/command_interpreter/helpers.py: remove __all__
    * vdat/command_interpreter/signals.py: same
    * vdat/command_interpreter/types.py: import numpy
    * doc/_source/codedoc/command_interpreter/index.rst: split the command
          interpreter documentation
    * doc/_source/codedoc/command_interpreter/*.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/relay.py: renambed signals
    * vdat/command_interpreter/signals.py: rewritten to have an behaviour
          similar to qt signals; CILogger not reimplemented; some signal␣
↪missing
    * vdat/command_interpreter/__init__.py: remove the import of the relay
    * vdat/command_interpreter/core.py: update according to the above changes
    * vdat/command_interpreter/helpers.py: provide some function that can be
          plugged in
    * tests/test_ci/test_helpers.py: added
    * tests/test_ci/test_signals.py: added
    * doc/_source/command_intepreter.rst: relays -> signals
    * doc/_source/codedoc/command_interpreter.rst: moved to
          command_interpreter/index.rst
    * doc/_source/codedoc/index.rst: index in the codedoc to allow reshaping␣
↪of
          the documentation
    * doc/_source/index.rst: same

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add a method that returns a range to
          SliceLike
    * vdat/command_interpreter/types.py: use SliceLike in primary_loop; remove
          _to_number function
    * tests/test_ci/test_utils.py: test the range method
    * tests/test_config.py: fixture to fix seed of random for repeatability of
          tests

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: adapt types to use the new
          keyword_regex and do_split = False
    * tests/test_ci/test_types.py: add tests for all the secondary keywords
```

```
    * doc/_source/command_intepreter.rst: fix the documentation

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add string and format customization␣
↪to
          SliceLike
    * tests/test_ci/test_utils.py: test it

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

        * merge trunk

2016-02-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/types.py: improve the keyword_regex␣
↪functionality
    * vdat/command_interpreter/utils.py: homogenize doc
    * vdat/command_interpreter/core.py: move back types to class properties␣
↪to be
          able to run in parallel (this needs investigation)
    * vdat/command_interpreter/exceptions.py: fix typo
    * tests/test_ci/test_types.py: test the keyword_regex functionality
    * doc/_source/command_intepreter.rst: document the new keyword_regex

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/conftest.py: move some fixture to session scope
    * vdat/command_interpreter/core.py: move the types initialisation into the
          __init__

2016-02-18 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter/utils.py: add SliceLike and copy some utils␣
↪from
      types.py
    * vdat/command_interpreter/exceptions.py: import the future and add
      CISliceError
    * setup.cfg: add doctest
    * tests/test_ci/test_utils.py: added
        * tests/test_ci/test_command_interpreter.py: moved
    * tests/test_ci/test_types.py: added and partially implemented
    * doc/_source/codedoc/command_interpreter.rst: add types and utils
      documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
```

```
        in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
        without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
        to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
        add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
        the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
        parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
        add ability to resize widgets; move logger widget into logger_
↪widget.py
        module
```

```
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>
```

```
    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report

2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking

        use ``vdat_config copy`` to update the configuration files
```

```
2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
```

```
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
          button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
↪Still
          to test if exceptions are handled correctly
```

```
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
          logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
          functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
          failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                    a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                                "ds9" menu is clicked. Only files from the␣
↪currently
                                selected tab are sent to "ds9" when the menu␣
↪item
                                is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>

      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                 astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
          changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
          necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
          initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
                          thumbnail generator looking for an
                          uninitialized fplane
```

```
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
↪avoid an
                                 astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
```

```
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
                                the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
```

```
                              plane panels. What is displayed as a thumbnail
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                           rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                       selecting a node causes a whole new
                                       central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                     with a regex specifying the file type to
                                     display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
```

```
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
```

```
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
```

```
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
```

```
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
                              try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                          select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here

2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>
```

```
    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities
```

```
    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
```

```
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
                                          use. It can show different reduction␣
↪steps in a
                                          branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
       * vdat/gui/static/unreduced.png: New image to differentiate
                                        between IFUs with and without input␣
```

```
↪files
      * vdat/libvdat/background.py: Uses the queue
      * vdat/libvdat/callback.py: Uses the queue
      * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
      * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

      Updated documentation and installation files:
      * doc/codedoc/gui.rst
      * doc/codedoc/reduction.rst
      * doc/index.rst
      * doc/queue.rst
      * requirements.txt
      * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

       * MANIFEST.in: Added fplane.txt file, so it is also installed!
       * doc/install.rst: Tweaked documentation
       * doc/launching.rst: As above
       * requirements.txt: Added command to install pyhetdex
       * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

       Added Sphinx documentation (under doc/), minor
       modifications to comments

       * AUTHORS
       * LICENSE
       * README.md: Added new dependencies
       * doc/: Added documentation here
       * vdat/gui/gui.py
       * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

       *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
       *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
       *  vdat/gui/gui.py:    Added a create masterbias button
       *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
       *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
       *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
       *  vdat/libvdat/symlink.py
       *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure
```

```
2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

        *  setup.py: Added APlpy to list of required Python modules
        *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
        *  vdat/gui/fplane.py: Improved comments
        *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
        *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
        *  vdat/libvdat/background.py
        *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
        *  vdat/libvdat/fits.py: Uses multiprocessing
        *  vdat/libvdat/handlers.py
        *  vdat/libvdat/reduction.py: Uses cure_interface
        *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
        *  vdat/libvdat/vdat.py: Set up log directory
        *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
                                                to pyhetdex:r74



2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
       * vdat/gui/relay.py: Module to relay signals to the GUI
       * vdat/libvdat/background.py
```

```
      * vdat/libvdat/callback.py
      * vdat/libvdat/database.py
      * vdat/libvdat/fits.py
      * vdat/libvdat/handlers.py
      * vdat/libvdat/reduction.py: Added function to subtract overscans
      * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
      * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

    Added an internal sqlite3 database to keep track
    of what files are available. Created a background
    thread with which to run things so they don't lock
    up the GUI when they're running. Implemented
    a simple code which loops through all fits files
    and converts them to PNGs.

    * vdat/gui/__init__.py: Moved call to symlink to here
    * vdat/gui/gui.py:    Added a (currently disabled) progress bar
    * vdat/libvdat/background.py: run jobs in a separate thread
    * vdat/libvdat/callback.py: Added calls to Background
    * vdat/libvdat/database.py: Internal database to keep track of files
    * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
    * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
    * vdat/libvdat/symlink.py: Can read rawdir from config file
    * vdat/libvdat/vdat.py:  Moved symlink from here.


2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging
```

```
2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
2016-02-26 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bump pyhetdex requirement to 0.4
    * vdat/libvdat/symlink.py: get most of the information for the symlinking
          from the file names
    * vdat/config/vdat_setting.cfg: put together most of the options needed␣
↪for
          symlinking
    * vdat/utilities.py: homogenize exceptions used by symlinking
    * doc/_source/dirstruct.rst: update documentation

2016-02-19 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore dist
    * setup.py: fix some packages minimum version, fix version number
    * tox.ini: fix some packages minimum version
    * vdat/command_interpreter/types.py: use Yields in documentation
    * vdat/gui/fplane.py: same
```

```
    * vdat/config/entry_point.py: vdat_config without subcommand behave the␣
↪same
          in py2 and py3
    * vdat/gui/buttons_menu.py: add fplane_widget property
    * vdat/gui/gui.py: mark two methods for possible delection
    * tests/test_buttons.py: monkeypatch CommandButton.fplane_widget to test
          without selected IFUs
    * tests/test_config.py: fix test of empty vdat_config call
    * tests/test_tree_view.py: adapt to the new gui structure
    * doc/_source/conf.py: cleanup, PEP8 and try to guess the pyhetdex version
          to for intersphinx
    * doc/_source/install.rst: change link anchor name

2016-02-17 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: add relevant files to package
    * pytest.ini: move pytest specif configurations here
    * requirements.txt: removed
    * setup.cfg: alias pytest=test command, remove pytest specific options
    * setup.py: use pytest-runner, remove tox from setup
    * tox.ini: remove all spurious dependences that are now reachable with␣
↪pip,
          add extra pypi url
    * vdat/__init__.py: get version from the package configuration
    * doc/_source/_templates/version.html: add version
    * doc/_source/conf.py: add the above version in the side bar
    * doc/_source/index.rst: add version number
    * doc/_source/install.rst: update installation info

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/entry_point.py: check if the target directory exists, even␣
↪if
          the path is not "."
    * tests/test_config.py: add a couple of tests for the new features

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

        * : merge ^/trunk into ^/branches/issue1178

2016-01-29 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: temporary disable tox_requires to avoid installation issues
    * vdat/config/entry_point.py: fix #1178, improve output info and argument
          parser

2016-01-27  Jan Snigula  <snigula@mpe.mpg.de>

        * tests/test_buttons.py: Adapt do changes made to setup_buttons

2016-01-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/gui.py: isolate the FplaneWidget and buttons; isolate the menu;
          add ability to resize widgets; move logger widget into logger_
```

```
↪widget.py
          module
    * vdat/libvdat/handlers.py: moved to vdat/gui/logger_widget.py
    * vdat/gui/logger_widget.py: add logger widget
    * vdat/gui/treeview_model.py: same
    * vdat/gui/buttons_menu.py: remove size constraints
    * vdat/gui/background.py: typo fixed

2016-01-19  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed missing X for missing IFUs
        * vdat/gui/gui.py: Pass fplane widget along
        * vdat/gui/buttons_menu.py: Same


2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_viewer.py: Pass basename through
        * vdat/gui/ifu_widget.py: Same
        * vdat/gui/fplane.py: Same

2016-01-18  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/ifu_widget.py: Fixed double click
        * vdat/gui/fplane.py: New thumbnails work now, zscaling mostly as
        well

2016-01-18 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add qimage2ndarray dependence
    * vdat/libvdat/symlink.py: use directory name into vdat exposure table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/gui/treeview_model.py: Changed a signal
        * vdat/gui/menu.py: Moved code to gui
        * vdat/database/core.py: Added new database table
        * vdat/config/tabs.yml: Updated regexes
        * vdat/gui/fplane.py: Restructured
        * vdat/gui/ifu_widget.py: Moved to direct fits file loading
        * vdat/database/models.py: Added new database table
        * vdat/gui/gui.py: Restructured
        * vdat/gui/buttons_menu.py: Changed yield behaviour
        * vdat/libvdat/symlink.py: Added new database table

2015-12-17  Jan Snigula  <snigula@mpe.mpg.de>

        * vdat/__init__.py: Bumped version to 0.1.0

2015-11-30 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/ifu_viewer.py: load it also if pyds9 fails to import; add
          notification about the import failure; add error box if
          pyds9 fails to connect to a ds9 session
```

```
2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added starextract
    * vdat/config/buttons.yml: same

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/command_interpreter/types.py: add is_regex key to primary␣
↪keywords;
          when getting file names match add the full path to the regex/
↪wildcard
    * doc/_source/command_intepreter.rst: document is_regex
    * vdat/config/vdat_commands.yml: add detection step; fix file names and
          regex in various commands; streamline some keyword values
    * vdat/config/extra_files/IFUcen_HETDEX.txt: added
    * vdat/config/buttons.yml: add

2015-11-26 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .coverage files, but no .coveragerc
    * .coveragerc: added
    * doc/_source/contributions.rst: add more info about tox
    * doc/_source/index.rst: add link to coverage report
    * requirements.txt: remove numpy
    * scripts/remove_empty_coverage.sh: added
    * scripts/symlink_pyqt.sh: call the python script with the full path to
          ``scripts`` directory
    * setup.cfg: remove coverage configurations
    * tests/test_buttons.py: fix test bug when ``commands`` is a string
    * tox.ini: build the documentation and coverage report

2015-11-24 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: aesthetic change
    * vdat/command_interpreter/core.py: raise an CIRunError when the return
          value is not null
    * vdat/command_interpreter/types.py: add possibility to manipulate the
          return value of the ``loop`` primary key
    * doc/_source/command_intepreter.rst: document it
    * vdat/command_interpreter/utils.py: added
    * vdat/config/buttons.yml: add the button to create the dither file
    * vdat/config/extra_files/dither_positions.txt: added
    * vdat/config/vdat_commands.yml: add the instruction to create the dither
          files
    * vdat/gui/buttons_menu.py: fix documentation typo

2015-11-08 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: change master* names to values compatible
          with cure's DitherEnvironment, add symlink command to create better␣
↪file
          names for the science frames
    * vdat/config/buttons.yml: add command for the symlinking
```

```
        use ``vdat_config copy`` to update the configuration files


2015-10-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/vdat.py: set xpa_method in the environment to local by
          default
    * vdat/config/vdat_setting.cfg: add option to modify the xpa_method and
          kdescription

2015-10-23 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: Added new tabs to display the products
                            of the new reduction buttons

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * doc/_source/command_intepreter.rst: more info about the selected IFU␣
↪given
    * tests/data/raw/20120301: replaced with new simulations
    * tests/test_command_interpreter.py: adapt to it
    * tests/test_symlink.py: same

2015-10-23 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: pass the selected ifus to the command
          interpreter
    * vdat/gui/fplane.py: PEP8
    * vdat/config/vdat_commands.yml: add the ``filter_selected`` keyword;
          improve match only fits filename starting with number
    * tests/test_buttons.py: test ifu selection

2015-10-22 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: put commands on the queue
    * vdat/gui/queue.py: adapt the queue to accept and return␣
↪CommandInterpreter
          instances; create set/get_queue functions
    * vdat/gui/background.py: set/get_background functions; adapt the
          background object to the above; fix bugs
    * vdat/gui/__init__.py: adapt to the above, remove callback
    * vdat/gui/relay.py: log also exception
    * vdat/gui/gui.py: fix some docstring
    * vdat/command_interpreter/core.py: fix a bug with template and exe
          substitution
    * vdat/command_interpreter/types.py: match the file name at the end of a
          string
    * vdat/libvdat/loggers.py: setup the loggers for the commands
    * vdat/libvdat/vdat.py: use it
    * vdat/config/core.py: better error handling when getting configurations
    * vdat/config/extra_files/*: added
    * vdat/config/entry_point.py: copy also the extra files
    * vdat/config/vdat_commands.yml: fix bugs and adjust paths
```

```
    * vdat/config/vdat_setting.cfg: fix the command logger configuration␣
↪entries
    * tests/conftest.py: force copying the configuration to avoid troubles
    * tests/test_buttons.py: finish the testing of the buttons
    * tests/test_command_interpreter.py: test alias replacing
    * tests/test_config.py: adapt the tests to the changes due to extra
          configuration files in subdirectories

2015-10-19 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: move button to custom class, create
          CommandInterpreter instances when pushing the buttons and report␣
↪problems
          with a dialog
    * vdat/gui/treeview_model.py: pep8
    * vdat/command_interpreter/exceptions.py: fix bug with CIExeError
    * vdat/config/vdat_commands.yml: masterarc needs an alias
    * vdat/config/vdat_setting.cfg: add comments about redux_dirs
    * vdat/database/models.py: PEP8
    * vdat/libvdat/vdat.py: inject CUREBIN into the path
    * tests/test_buttons.py: add a test clicking the buttons

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/buttons_menu.py: rewrite the creation of the buttons
    * vdat/gui/gui.py: use the new button menu
    * vdat/gui/treeview_model.py: connect the button menu to switch set of
          buttons when changing directory; use a signal to change the central␣
↪and
          button panels
    * vdat/config/buttons.yml: configuration file driving the button creation
    * vdat/config/vdat_setting.cfg: add it
    * vdat/config/core.py: add it to the files to load
    * vdat/config/entry_point.py: add it to the files to copy
    * vdat/config/vdat_commands.yml: little formatting
    * tests/test_buttons.py: test the button widget; for now test that is
          correctly created and that the switching happens correctly
    * tests/test_command_interpreter.py: make sure to get a file for the ifu␣
↪34

2015-10-16 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/vdat_commands.yml: added
    * vdat/config/vdat_setting.cfg: add the above file
    * vdat/config/core.py: load vdat_commands.yml
    * vdat/config/entry_point.py: copy it; don't overwrite existing files by
          default
    * tests/test_config.py: test the vdat_config command

2015-10-14 Francesco Montesano <montefra@mpe.mpg.de>

        Tests run for python 2.7, 3.4 and 3.5

    * tests/test_command_interpreter.py: test also part of the run method.␣
```

```
↪Still
         to test if exceptions are handled correctly
    * vdat/command_interpreter/core.py: fix bugs and improve error handling␣
↪and
         logging
    * vdat/command_interpreter/relay.py: fix bugs with progress relay
    * vdat/command_interpreter/types.py: fix bugs and don't cover template
         functions
    * MANIFEST.in: add readme and requirement file to avoid tox building
         failures
    * requirements.txt: add numpy to avoid scipy building failures
    * setup.py: add new_file entry point

2015-10-14 Daniel Farrow <dfarrow@mpe.mpg.de>

     * vdat/gui/fplane.py: Aligned the scale combobox left to make it prettier
     * vdat/libvdat/show_fits.py: replaced another call to astropy getdata␣
↪with
                                   a ``with open(fn, 'rb')`` to avoid the␣
↪astropy bug
     * vdat/gui/ifu_viewer.py: "Send to ds9" menu now generated dynamically␣
↪when the
                               "ds9" menu is clicked. Only files from the␣
↪currently
                               selected tab are sent to "ds9" when the menu␣
↪item
                               is selected.

2015-10-13 Daniel Farrow <dfarrow@mpe.mpg.de>

      * requirements.txt: added pyds9 repo
      * setup.py: added pyds9 repo
      * vdat/gui/ifu_viewer.py: wrapped get_header in with open(f) to avoid␣
↪the
                                astropy bug of not closing files.

2015-10-13 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: group together entripoints
    * vdat/command_interpreter/core.py: some bug fix, use execute types, some
         changes with the exception handling
    * vdat/command_interpreter/exceptions.py: rename some exception
    * vdat/command_interpreter/types.py: add execute type and implement all␣
↪the
         necessary types
    * tests/test_command_interpreter.py: test most of the command interpreter
         initialisation
    * doc/_source/command_intepreter.rst: extend documentation
    * vdat/config/ci_documentation.yml: removed

2015-10-12 Daniel Farrow <dfarrow@mpe.mpg.de>:

    * vdat/gui/fplane.py: moved update IFUs from init to
                          change_focal_plane, to avoid the
```

```
                          thumbnail generator looking for an
                          uninitialized fplane
    * vdat/gui/ifu_viewer.py: Added option to select frames
                              and send them to a new or existing
                              ds9 session
    * setup.py: Added pyds9 to install requires

2015-10-09 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/database/core.py: added a table to store image brightness scaling␣
↪parameters
    * vdat/database/models.py: as above
    * vdat/gui/fplane.py: Added a section to control the brightness scaling␣
↪of the thumbnails in the
                          focal plane. User can select scaling per fits file,␣
↪or a global
                          scaling (which can be user specified) for the whole␣
↪focal plane.
                          The Fplane class in now its own QWidget.
    * vdat/gui/gui.py: Added comments
    * vdat/gui/ifu_viewer.py: Suppresses warnings from Ginga ;-)
    * vdat/gui/ifu_widget.py: IFU viewers are parented to the main window, so
                              they can persist when the user changes fplane
    * vdat/libvdat/show_fits.py: Casts the number of rows to an integer␣
↪explicitly. Connects
                                 to a database to find, or set, global␣
↪brightness
                                 scaling parameters when required for the␣
↪thumbnails. Uses a
                                 file object with astropy getdata in order to␣
↪avoid an
                                 astropy bug.

2015-10-09 Francesco Montesano <montefra@mpe.mpg.de>

    * tests/test_command_interpreter.py: first tests added
    * vdat/command_interpreter/core.py: better exceptions
    * vdat/command_interpreter/exceptions.py: same

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: bootstrap setuptools if it's not installed
    * ez_setup.py: bootstrap module

2015-10-07 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config/core.py: separate the loading from the getting of the
          configurations: allow more homogeneous handling of the␣
↪configuration files
    * vdat/config/vdat_setting.cfg: comment a bit more
    * vdat/config/entry_point.py: move here the implementation of the
          ``vdat_config`` executable; use pkg_resources to get copy the
          configuration files
    * setup.py: update the entry point
```

```
    * vdat/gui/fplane.py: use the new configuration interface; PEP8
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_viewer.py: same
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/queue.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * tests/conftest.py: same
    * doc/Makefile(livehtm): add vdat/config to the tracked directories
    * doc/_source/codedoc/config.rst: add code documentation
    * doc/_source/index.rst: same

2015-10-07 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/tabs.yml: Added new configuration for the upgraded␣
↪thumbnail
                                creation (see below)
        * vdat/gui/background.py: Immediate background thread waits for last␣
↪job to stop
                                  before running the next job. Toggle system
                                  for the isImmRunning flag removed as it␣
↪depended
                                  on the main thread being available. Now the
                                  immediate background thread controls the␣
↪isImmRunning
                                  flag is controlled by the Worker in the␣
↪thread.
        * vdat/gui/fplane.py: Waits for jobs on immediate thread to stop,␣
↪stops
                              QObjects still in use from being deleted.
        * vdat/gui/gui.py: Handles the uses clicking the close button, now␣
↪waits
                           for running jobs on the immediate thread to end.␣
↪This
                           stops seg faults from a sudden close.
        * vdat/gui/ifu_widget.py: Fixes a bug by removing the auto-
↪regeneration
                                  of corrupted thumbnails. Simply dumps them␣
↪instead.
        * vdat/libvdat/show_fits.py: Based on options in tabs.yml, create a␣
↪grid of
                                     thumbnails for the IFU widget with␣
↪entries
                                     for the different channels, amps.

2015-10-05 Daniel Farrow <dfarrow@mpe.mpg.de>

        * vdat/config/core.py: Added load_yaml
        * vdat/config/tabs.yml: Moved tabs subsections to be direcly under
                                the different node types (on the same level as
```

```
                                the ifu_viewer and main subsections).
        * vdat/gui/fplane.py: Added tools to save and generate focal
                              plane panels. What is displayed as a thumbnail
                              is decided by the user via a combo box (i.e.␣
↪the raw fits, fibre-collapsed
                              images, arcs, flats etc.). Defaults are set in␣
↪the tabs.yml
        * vdat/gui/gui.py: Central panel now generated dynamically
                           rather than at initialization.
        * vdat/gui/ifu_viewer.py: Moved load_yaml to vdat.config
        * vdat/gui/relay.py: Added 'change_centralPanel' signal.
        * vdat/gui/treeview_model.py: Rather than prompting an update of the␣
↪IFUs,
                                       selecting a node causes a whole new
                                       central panel to be created
        * vdat/libvdat/show_fits.py: Now show_thumbnails takes a config object
                                     with a regex specifying the file type to
                                     display

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add yaml
    * vdat/libvdat/loggers.py: reorganize the loggers code to remove␣
↪repetitions
    * vdat/config/vdat_setting.cfg: adapt the configuration to this
    * vdat/libvdat/vdat.py: create appropriate ginga logger
    * vdat/gui/ifu_viewer.py: PEP8 frenzy; use ginga logger
    * doc/_source/codedoc/reduction.rst: add logging documentation
    * doc/_source/gui.rst: fix warning
    * doc/_source/index.rst: add todo about logging

2015-10-05 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/command_interpreter: added
    * vdat/command_interpreter/__init__.py: import interface at module level
    * vdat/command_interpreter/core.py: implement the interpreter
    * vdat/command_interpreter/exceptions.py: define custom exceptions
    * vdat/command_interpreter/helpers.py: will contain some helper function
    * vdat/command_interpreter/relay.py: relay-like interface for␣
↪communication
          between the interpreter and the world
    * vdat/command_interpreter/types.py: define classes to deal with types
    * vdat/config/ci_documentation.yml: very wordy yaml file to use for
          documentation purposes
    * doc/Makefile: add command_interpreter for auto-compilation
    * doc/_source/codedoc/command_interpreter.rst: added
    * doc/_source/command_interpreter.rst: added
    * doc/_source/index.rst: add the above documents
    * doc/_source/codedoc/reduction.rst: remove reduction module
    * vdat/libvdat/callback.py: get logger in method

2015-09-30 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/config/tabs.yml: configuration file that decides what is
```

```
                            displayed in different panels
    * vdat/config/vdat_setting.cfg: Add tabs.yml to config file
    * vdat/gui/background.py: Worker now passes **kwargs and *args
    * vdat/gui/ifu_viewer.py: Read in tabs.yml, creates tabs in the
                              viewer based on it.
    * vdat/gui/ifu_widget.py: When doduble clicked and no
                              directory selected, ask the user to select
                              one
    * vdat/gui/treeview_model.py: Passes the type of directory selected
                                  to show_fits
    * vdat/libvdat/loggers.py: Added a generic logger class to store
                               Ginga loggers
    * vdat/libvdat/reduction.py: ifuid -> ihmpid when deriving filenames
    * vdat/libvdat/show_fits.py: Saves the type of directory selected in the␣
↪IFU object
                                 this might not be ideal
    * doc/_source/gui.rst: Added some GUI documentation
    * vdat/config/core.py: Added tabs.yml to CONFIG_FILES

2015-09-23 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore build and .eggs directories
    * setup.cfg: same
    * setup.py: create setuptools command @tox@ to fetch tox, if necessary,␣
↪and
          run tox
    * scripts/symlink_pyqt.sh: don't print error if pyqt4 is not symlinked
    * doc/_source/contributions.rst: added; describe testing via tox and py.
↪test
    * doc/_source/index.rst: add the above
    * doc/_source/install.rst: update dependency list

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: make the gui tests succeed on tox too

2015-09-22 Francesco Montesano <montefra@mpe.mpg.de>

    * tox.ini: added
    * setup.cfg: ignore .tox when discovering tests
    * svn:ignore: add .tox directory
    * MANIFEST.in: fix config directory name change
    * scripts/symlink_pyqt.{sh,py}: symlink pyqt4 and sip into the tox virtual
          enviroments
    * vdat/libvdat/symlink.py: do not try to commit if the redux directory is
          empty
    * tests/conftest.py: initialise the main logger

2015-09-21 Francesco Montesano <montefra@mpe.mpg.de>

    * svn:ignore: ignore .cache directory
    * setup.py: minimum pytest-qt version; fix console_scripts module name
    * tests/conftest.py: no need to get for fixtures to get the configuration
          and to start the database
```

```
    * tests/test_symlink.py: clean the loggers
    * tests/test_tree_view.py: no need to start database;
    * vdat/config/vdat_setting.cfg: disable multiprocessing by default; use␣
↪only
          one max delta time for calibration
    * vdat/database/base.py: property to get data as dictionary
    * vdat/database/core.py: init get directory where the database should go;
          fix bug with @connect@
    * vdat/database/models.py: new table columns, method to create the path␣
↪and
          merge multiple rows into one
    * vdat/libvdat/symlink.py: initialize, fill and update the database when␣
↪doing the
          symlinking
    * vdat/gui/treeview_model.py: build the view from the database
    * vdat/libvdat/vdat.py: don't initialize the database
    * vdat/utilities.py: merge dictionaries function added; modify some errors

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

        * *py: use the __future__
    * vdat/database/__init__.py: split into sub modules and import only the
          "public" interface
    * vdat/database/base.py: define the database and the base model
    * vdat/database/core.py: initialise the database and deal with the
          connection
    * vdat/database/models.py: custom models are implemented here
    * vdat/database/old_database.py: removed
    * vdat/gui/treeview_model.py: use floor with datetime.timedelta
    * vdat/libvdat/symlink.py: same

2015-09-15 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/config: renamed from vdat/vdat_config
    * vdat/config/__init__.py: import only "public" interface
    * vdat/config/core.py: renamed from vdat/libvdat/config.py and adapted
    * setup.py: add ginga, adapt ``vdat_config`` entry point to new
      directories
    * vdat/gui/fplane.py: use new config subpackage
    * vdat/gui/ifu_widget.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/loggers.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/gui/relay.py: instantiate ``SignalClass`` inside a function and␣
↪save
          in a local list to allow for testing
    * vdat/gui/__init__.py: use the new implementation
    * vdat/gui/ifu_viewer.py: same (plus PEP8)
    * vdat/gui/gui.py: same and config subpackage
    * vdat/gui/queue.py: same
    * vdat/libvdat/fits.py: same
    * vdat/utilities.py (config_directory): moved to vdat/config/core.py
```

```
    * tests/conftest.py: adapt to the above changes, use pyqt4 v2 api, add
          fixtures to start the database and to clear lists and dictionaries␣
↪at the
          end of a test to allow reuse
    * tests/test_tree_view.py: use new fixtures

2015-09-14 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: dialog confirming deletion; fix bug with
          indexing

2015-09-11 Francesco Montesano <montefra@mpe.mpg.de>

    * MANIFEST.in: corrected
        * doc/_source: created; conf.py, the _template and _static␣
↪directories and
          all the rst files has been moved into this directory
    * doc/Makefile: adapted to the changes
        * doc/_source/*: small improvements
    * setup.py: add vdat_config entry point
    * vdat/libvdat/config.py: implement ``vdat_config copy`` command
    * vdat/utilities.py: returns the configuration directory
    * vdat/libvdat/callback.py: make the documentation happy

2015-09-10 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/database/__init__.py: add extra fields in preparation for issues
↪#1048
          #1049 and #1053
    * vdat/gui/treeview_model.py: add context menu and handle clone and remove
          actions as per #1048, adapt the building of the tree view to␣
↪account for
          this
    * vdat/libvdat/symlink.py: add ``is_clone`` entry to the shot_file and
          ignore cloned directories when re-symlinking
    * vdat/utilities.py(write_to_shot_file): possible to chose between write
          and append mode when writing
        * vdat/gui/background.py(Background): rename ``cls`` to ``self`` for
          consistency

2015-09-07 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add peewee dependency
    * vdat/libvdat/database.py: moved to vdat/database/__init__.py
    * vdat/database/__init__.py: implement the database table associated
          with the entries in the tree view
    * vdat/database/old_database.py: keep it for reference, it will be
      eventually removed
    * vdat/gui/treeview_model.py: populate the database
        * vdat/utilities.py: move here from libvdat/symlink.py the functions␣
↪to
          read and write the shot files
    * vdat/libvdat/symlink.py: modify accordingly
    * vdat/libvdat/vdat.py: initialise the database
```

```
2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

        * vdat/gui/queue.py(ModifyableListWidget.keyPressEvent): for keys␣
↪other than
          the selected one, call the parent class implementation; no return
    * vdat/gui/gui.py: move the buttons setup to buttons_menu module
    * vdat/gui/buttons_menu.py: same, set buttons max size to 400
    * vdat/gui/fplane.py: the layout is an attribute, no need for a function
    * vdat/gui/treeview_model.py: set max width for the panel to 400

2015-09-03 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: save ticked directories into the␣
↪configuration
    * vdat/libvdat/reduction.py: adapt to the new directory structure
    * vdat/libvdat/loggers.py: set up the cure task loggers
    * vdat/libvdat/cure_interface.py: move the logger setting up to loggers.py
    * vdat/vdat_config/vdat_setting.cfg: add cure task loggers options

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/background.py: moved into vdat/gui as it uses all qt stuff
    * vdat/gui/background.py(Background): make it a proper class, initialising
          the threads with a parent to get rid of qt warnings about objects␣
↪not
          owned by anything
    * vdat/gui/background.py(get_background): create and/or return a␣
↪Background
          instance; once created it returns always the same instance
    * vdat/gui/__init__.py: use get_background
    * vdat/gui/treeview_model.py: same

2015-08-31 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: PEP8
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/relay.py: same
    * vdat/gui/treeview_model.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/background.py: same
    * vdat/libvdat/show_fits.py: same
    * vdat/gui/ifu_widget.py: same, plus variable names fixed
    * vdat/gui/menu.py: PEP8, move the action for the queue and all␣
↪connections
          to queue.py
    * vdat/gui/queue.py: implement here the queue action and connect the␣
↪signals
          properly

2015-08-28 Daniel Farrow <dfarrow@mpe.mpg.de>

    * setup.py: Added ginga to requires
```

```
    * vdat/gui/__init__.py: set the QString and QVariant types for ginga␣
↪compatibility
    * vdat/gui/ifu_viewer.py: Tells ginga to use pyqt4
    * vdat/libvdat/callback.py: import show_fits instead of create_thumbnails␣
↪(bug 1037)
    * vdat/libvdat/show_fits.py: Checks if any files are found before␣
↪creating thumbnail

2015-08-25 Francesco Montesano <montefra@mpe.mpg.de>

    * doc: ignore build directory
    * doc/codedoc/gui.rst: move the treeview model here
    * doc/codedoc/reduction.rst: remove the treeview model
    * doc/conf.py: set matplotlib backend to agg to avoid pyqt4/5 conflicts

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>
    * vdat/gui/ifu_viewer.py: A Ginja based panel that
                              displays a zoomable, pan-able
                              colourscale-able image of a FITs file,
                              with an added display for the header
    * vdat/gui/ifu_widget.py: Launches and IFUViewer on double-click

2015-08-25 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/fplane.py: Added yield all IFUs function,
                          added a flag that when set stops
                          looping over IFUs (to stop
                          jobs more cleanly)
    * vdat/gui/gui.py: Added import to flag above (for later)
    * vdat/gui/ifu_widget.py: Test to see if a thumbnail
                              image of IFU is corrupted, if yes
                              try to regenerate
    * vdat/gui/relay.py: Added parent argument ot initialisation
    * vdat/gui/treeview_model.py: Calls function to show postage
                                  stamps of FITs images when
                                  a directory is selected.
    * vdat/libvdat/background.py: Added a run_now function, and an
                                  extra thread for it. This is designed
                                  for important tasks to jump the queue.
    * vdat/libvdat/callback.py: Added a comment
    * vdat/libvdat/show_fits.py: New module which generates PNG images
                                 of the detector FITs files

2015-08-20 Daniel Farrow <dfarrow@mpe.mpg.de>

    * doc/command_line_tool.rst: Draft specifiation for command line tool
    * doc/index.rst: Added link to above
    * vdat/gui/fplane.py: Moved 'yield_selected_ifus' here, added select all␣
↪and
                          select none functions
    * vdat/gui/ifu_widget.py: Exists and selected are now properties
    * vdat/gui/menu.py: Add a selection menu with 'select all' and 'select␣
↪none'
    * vdat/libvdat/reduction.py: Removed 'yield_selected_ifus' from here
```

```
2015-08-14 Daniel Farrow <dfarrow@mpe.mpg.de>

    * vdat/gui/__init__.py: Now sets the parent of the signal relay
    * vdat/gui/gui.py: Renamed MainWindow -> mainWindow as it's not a class
    * vdat/gui/menu.py: Sets up the new menu bar at the top of the GUI
    * vdat/gui/queue.py: Queue window can be hidden and revealed from the new␣
↪menu bar
    * vdat/gui/relay.py: Uses dictionaries to store signals

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/__init__.py: the main frame must be saved in a variable, even␣
↪if
          it's not used, in the qt app to work properly

2015-08-13 Francesco Montesano <montefra@mpe.mpg.de>

        As now it's not possible to run more than one test running the gui at␣
↪a
        time, as it crashes. This is very likely due to the fact that there␣
↪are qt
        objects around without a parent, and this confuses the qtbot

    * setup.py: add pytest-qt dependency
    * tests/conftest.py: use matplotlib agg backend to avoid pyqt4/5 clashes.
          Add fixtures and move some common code away from test_symlink
    * tests/test_symlink.py: adapt to the above
    * tests/test_tree_view.py: test 93% of the tree view
    * vdat/gui/__init__.py: isolate the code making the main and queue window
          to allow setting up tests
    * vdat/libvdat/handlers.py: add parent widget in the handler
    * vdat/gui/gui.py: adapt to the above
    * vdat/gui/treeview_model.py: set the ReductionTreeviewModel as child of␣
↪the
          ReductionQTreeView
    * vdat/libvdat/background.py: add a todo

2015-08-11 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/gui/treeview_model.py: moved to gui
    * vdat/libvdat/treeview_model.py: create the tree view from the redux
          directory structure, make only directory containing the fits file
          selectable, make calibration directories checkable to allow select
          specific calibrations during reduction.
    * vdat/gui/buttons_menu.py: add temporary button to test the tree view
          model. Will be removed once the other buttons will be reimplemented
    * vdat/gui/gui.py: move the creation of the tree view to the proper␣
↪module;
          add the above button
        * vdat/libvdat/reduction.py: fixed bug with missing configuration␣
↪section

2015-08-04 Francesco Montesano <montefra@mpe.mpg.de>
```

```
        WARNING: this changes break the gui button functionalities

    * .: ignore coverage output files and directories
    * setup.py: convert to pytest
    * setup.cfg: same
    * vdat/libvdat/symlink.py: make rerun symlink more robust and write a file
          "SHOT_FILE" with all the relevant informations of the symlinked␣
↪shot as a
          json
    * vdat/utilities.py: add json serialisation and de-serialisation of␣
↪datetime
          instances
    * vdat/vdat_config/vdat_setting.cfg: add max_delta_zro option
    * vdat/gui/__init__.py: don't import symlink module
    * tests: add tests
    * tests/data/raw: add fits files for testing: zro, sci, flt, arc shots, 3
          IFUs and 3 exposures each
    * tests/conftest.py: add fixtures
    * tests/test_symlink.py: test the symlinking (edge cases still missing)

2015-07-30 Francesco Montesano <montefra@mpe.mpg.de>

        WARNING: this changes break the gui button functionalities

    * vdat/libvdat/symlink.py: almost completely rewritten; data symlinked at
          the shot level; calibration frames divided in subdirectories; flat␣
↪and arc
          collected in the same 'cal' directory
    * vdat/libvdat/vdat.py: symlink done before calling the gui;␣
↪multiprocessing
          set up
    * vdat/utilities.py: custom exceptions added
    * vdat/vdat_config/vdat_setting.cfg: add raw directory, add␣
↪multiprocessing,
          add maximum time delta to use when grouping flat and arc frames
    * vdat/libvdat/loggers.py: set logger level to debug
    * vdat/gui/__init__.py: don't do the symlink here

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * vdat/libvdat/loggers.py: created moving code out of vdat.py and
          reorganizing it
    * vdat/libvdat/vdat.py: updated according to the above
    * vdat/vdat_config/vdat_setting.cfg: more logging configuration given

2015-07-27 Francesco Montesano <montefra@mpe.mpg.de>

    * setup.py: add six dependency
    * vdat/gui/__init__.py: PEP8
    * vdat/gui/buttons_menu.py: PEP8 and documentation fixes
    * vdat/gui/fplane.py: same
    * vdat/gui/gui.py: same
    * vdat/gui/ifu_widget.py: same
```

```
    * vdat/gui/relay.py: same
    * vdat/gui/queue.py: same, plus using self instead of parent class method
    * vdat/libvdat/background.py: same
    * vdat/libvdat/callback.py: same
    * vdat/libvdat/config.py: same
    * vdat/libvdat/cure_interface.py: same
    * vdat/libvdat/database.py: same
    * vdat/libvdat/fits.py: same
    * vdat/libvdat/handlers.py: same
    * vdat/libvdat/reduction.py: same
    * vdat/libvdat/symlink.py: same
    * vdat/libvdat/treeview_model.py: same
    * vdat/libvdat/vdat.py: same
    * vdat/utilities.py: same

2015-07-02 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/libvdat/reduction.py: Added routine for creating error files␣
↪with photon
                                    noise, extracting the data region of the␣
↪files
                                    and joining the amplifiers
       * vdat/vdat_config/vdat_setting.cfg: Added options for the new commands
       * vdat/gui/gui.py:     Added buttons for the new routines

2015-07-01 Daniel Farrow <dfarrow@mpe.mpg.de>
       * vdat/gui/gui.py: Switched from file browser to a custom model in the␣
↪treeview widget. Currently
                          it just gives a hard-coded example of the new␣
↪custom model's
                          capabilities.
       * vdat/libvdat/treeview_model.py: Added a customisable model for the␣
↪treeview widget to
                                         use. It can show different reduction␣
↪steps in a
                                         branching hierachy.



2015-06-16 Daniel Farrow <dfarrow@mpe.mpg.de>

       * vdat/gui/__init__.py: Create a queue
       * vdat/gui/buttons_menu.py: Added comments
       * vdat/gui/fplane.py: Got rid of the unneccessary extra IFU type
                             now there is just one type defined in
                             ifu_widget
       * vdat/gui/gui.py: Added a button
       * vdat/gui/ifu_widget.py: Turned into a pyhetdex IFU type, added
                                 methods to update the picture in the IFU
                                 to reflect whether the IFU has input files
                                 or not.
       * vdat/gui/queue.py: A queue window, which keeps track of the
                            commands a user has requested and runs
                            them when they reach the head of the queue. The
                            user can also delete these commands.
```

```
        * vdat/gui/static/unreduced.png: New image to differentiate
                                          between IFUs with and without input␣
↪files
        * vdat/libvdat/background.py: Uses the queue
        * vdat/libvdat/callback.py: Uses the queue
        * vdat/libvdat/reduction.py: New function the subtract masterbias and␣
↪overscan from files
        * vdat/libvdat/symlink.py: Tells the IFU object it exists if it finds␣
↪FITs files from it

        Updated documentation and installation files:
        * doc/codedoc/gui.rst
        * doc/codedoc/reduction.rst
        * doc/index.rst
        * doc/queue.rst
        * requirements.txt
        * MANIFEST.in

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

         * MANIFEST.in: Added fplane.txt file, so it is also installed!
         * doc/install.rst: Tweaked documentation
         * doc/launching.rst: As above
         * requirements.txt: Added command to install pyhetdex
         * vdat/libvdat/vdat.py: Added check to see if config file exists

2015-06-12 Daniel Farrow <dfarrow@mpe.mpg.de>

         Added Sphinx documentation (under doc/), minor
         modifications to comments

         * AUTHORS
         * LICENSE
         * README.md: Added new dependencies
         * doc/: Added documentation here
         * vdat/gui/gui.py
         * vdat/libvdat/reduction.py



2015-06-11 Daniel Farrow <dfarrow@mpe.mpg.de>

         *  vdat/gui/buttons_menu.py: Fixed python3 compatibility by using␣
↪String instead of QString
         *  vdat/gui/fplane.py: Added a custom IFU class with a variable␣
↪indicating if the IFU is selected
         *  vdat/gui/gui.py:    Added a create masterbias button
         *  vdat/gui/ifu_widget.py: Made the widget selectable, add blue frame␣
↪when not selected
         *  vdat/libvdat/cure_interface.py: Now tells the worker to clear jobs,
↪ so the progress bar is refreshed
         *  vdat/libvdat/reduction.py: Added create master bias function,␣
↪subtract overscan now only works on selected IFUs
         *  vdat/libvdat/symlink.py
```

```
        *  vdat/vdat_config/vdat_setting.cfg: Added a format statement␣
↪specifying the VIRUS filename structure



2015-06-01 Daniel Farrow <dfarrow@mpe.mpg.de>

          Started using the multiprocessing tools
          from pyhetdex to run jobs in parallel.
          Implemented a progress bar to check how
          far a job has gone. Moved logs to a
          user specified log directory. A few improvements
          in commenting and other minor things.

         *  setup.py: Added APlpy to list of required Python modules
         *  vdat/gui/buttons_menu.py: Now supports displaying a tooltip
         *  vdat/gui/fplane.py: Improved comments
         *  vdat/gui/gui.py: Got rid of silly buttons like "Make Coffee"
         *  vdat/gui/relay.py: A module to send signals to the GUI (i.e.␣
↪update progress bar etc)
         *  vdat/libvdat/background.py
         *  vdat/libvdat/cure_interface.py: Functions to wrap around CURE,␣
↪runs in parallel
         *  vdat/libvdat/fits.py: Uses multiprocessing
         *  vdat/libvdat/handlers.py
         *  vdat/libvdat/reduction.py: Uses cure_interface
         *  vdat/libvdat/symlink.py: Tells the user when symlinking is done
         *  vdat/libvdat/vdat.py: Set up log directory
         *  vdat/vdat_config/vdat_setting.cfg: Added log directory and␣
↪changed wildcards to conform
                                                 to pyhetdex:r74



2015-05-29 Francesco Montesano <montefra@mpe.mpg.de>

          * vdat/libvdat/symlink.py: update ``scan_dirs`` after pyhetdex:r74.␣
↪PEP8
                and numpydoc compliant

2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

       A few minor modifications to style based on
       Francesco's comments. Added a subtract overscan
       routine. Switched to using file names rather
       than a database when running commands. Added
       a module to make it easier for the code
       to signal the GUI.


       * vdat/gui/buttons_menu.py
       * vdat/gui/fplane.py
       * vdat/gui/gui.py
       * vdat/gui/ifu_widget.py
```

```
      * vdat/gui/relay.py: Module to relay signals to the GUI
      * vdat/libvdat/background.py
      * vdat/libvdat/callback.py
      * vdat/libvdat/database.py
      * vdat/libvdat/fits.py
      * vdat/libvdat/handlers.py
      * vdat/libvdat/reduction.py: Added function to subtract overscans
      * vdat/libvdat/symlink.py: Tells GUI to update file browser panel when␣
↪symlink done
      * vdat/vdat_config/vdat_setting.cfg: Added some wildcards to find files


2015-05-21 Daniel Farrow <dfarrow@mpe.mpg.de>

    Added an internal sqlite3 database to keep track
    of what files are available. Created a background
    thread with which to run things so they don't lock
    up the GUI when they're running. Implemented
    a simple code which loops through all fits files
    and converts them to PNGs.

    * vdat/gui/__init__.py: Moved call to symlink to here
    * vdat/gui/gui.py:    Added a (currently disabled) progress bar
    * vdat/libvdat/background.py: run jobs in a separate thread
    * vdat/libvdat/callback.py: Added calls to Background
    * vdat/libvdat/database.py: Internal database to keep track of files
    * vdat/libvdat/fits.py:   Implements a simple fits -> PNG conversion
    * vdat/libvdat/handlers.py: Now uses signals to interface with GUI to be␣
↪thread safe
    * vdat/libvdat/symlink.py: Can read rawdir from config file
    * vdat/libvdat/vdat.py:  Moved symlink from here.


2015-05-18 Daniel Farrow <dfarrow@mpe.mpg.de>

        Switched to using PyQt4 and fixed python 2.7
        compatibility. Added symlink function as
        described by issue #821

      * vdat/gui/__init__.py:   ... switched to PyQt4
      * vdat/gui/buttons_menu.py:  PyQt4
      * vdat/gui/fplane.py:  PyQt4
      * vdat/gui/gui.py: PyQt4
      * vdat/gui/ifu_widget.py: PyQt4
      * vdat/libvdat/callback.py: Function factory to return functions to␣
↪connect to
        button clicks. Currently just returns a function that prints "Not␣
↪implemented"
      * vdat/libvdat/config.py: Read options to do with logging
      * vdat/libvdat/handlers.py: PyQt4
      * vdat/libvdat/symlink.py: symlinks files from raw to redux directory␣
↪(issue 821)
      * vdat/libvdat/vdat.py: Sets up logging, switched to PyQt4
      * vdat/vdat_config/vdat_setting.cfg: Added options to do with logging
```

```
2015-05-14 Daniel Farrow <dfarrow@mpe.mpg.de>

        Added a new handler for the logger which
        prints colour-coded messaged to the text
        panel of the VDAT GUI

        * libvdat/handler.py: Created a new Handler for logging
        * gui/gui.py:  Attached the QTextEdit panel to the Handler
        * gui/__init__: Prints a welcome message using the new logger


2015-05-05  Daniel Farrow <dfarrow@mpe.mpg.de>

        * setup.py: Modified to point to vdat.py:main()
        * libvdat/__init__.py: added (empty file)
        * libvdat/vdat.py: added, reads in config file, starts GUI
        * vdat_config/vdat_settings.cfg: added
        * vdat_config/fplane.txt: added
        * gui/fplane.py:  Reads in fplane.txt and displays it
        * gui/ifu_widget.py: Added. Derives QLabel, shows the IFU
        * gui/ifu_widget.py: Includes a custom handler for resize events
        * gui/resources/empty.png: Copied from Quicklook
        * MANINFEST.in: Read by pip to tell it to install the empty.png file

2015-05-04  Francesco Montesano  <montefra@mpe.mpg.de>

        * gui: moved to vdat/gui
        * README.md: some basic installation info added
        * setup.py: install vdat package and create ``vdat`` executable
        * setup.cfg: setup configuration
        * vdat/__init__.py: version number
        * vdat/gui/buttons_menu.py: absolute import, some PEP8
        * vdat/gui/fplane.py: absolute import, some PEP8
        * vdat/gui/gui.py: absolute import, some PEP8
        * vdat/gui/__init__.py: same, isolate main function
        * svn:ignore: egg dir added
```

3.5 TODO

Todo: Replicating code is in general a bad idea, but I haven't found yet a better solutions. This method implementation needs some re-thinking.

original entry

Todo: describe this

original entry

Todo: Add some instruction about the loggers settings

original entry

CHAPTER 4

Copyright notice

CHAPTER 5

Artwork copyright

The artwork consists of:

- the logo with the `VDAT` text shown in upper left corner of the documentation;
- the two splash screens visible when starting VDAT;
- the VDAT window icon.

CHAPTER 6

Links

- genindex
- modindex
- search

Python Module Index

V

Index

Symbols

A

B

C

D

E

F

G

H

M

N

O

S

T

U

V

W

Y

Z