Taurus Community and Collaboration Model

Guifré Cuní, Emilio Morales, Miquel Navarro and Zbigniew Reszela
on behalf of ALBA Controls Section

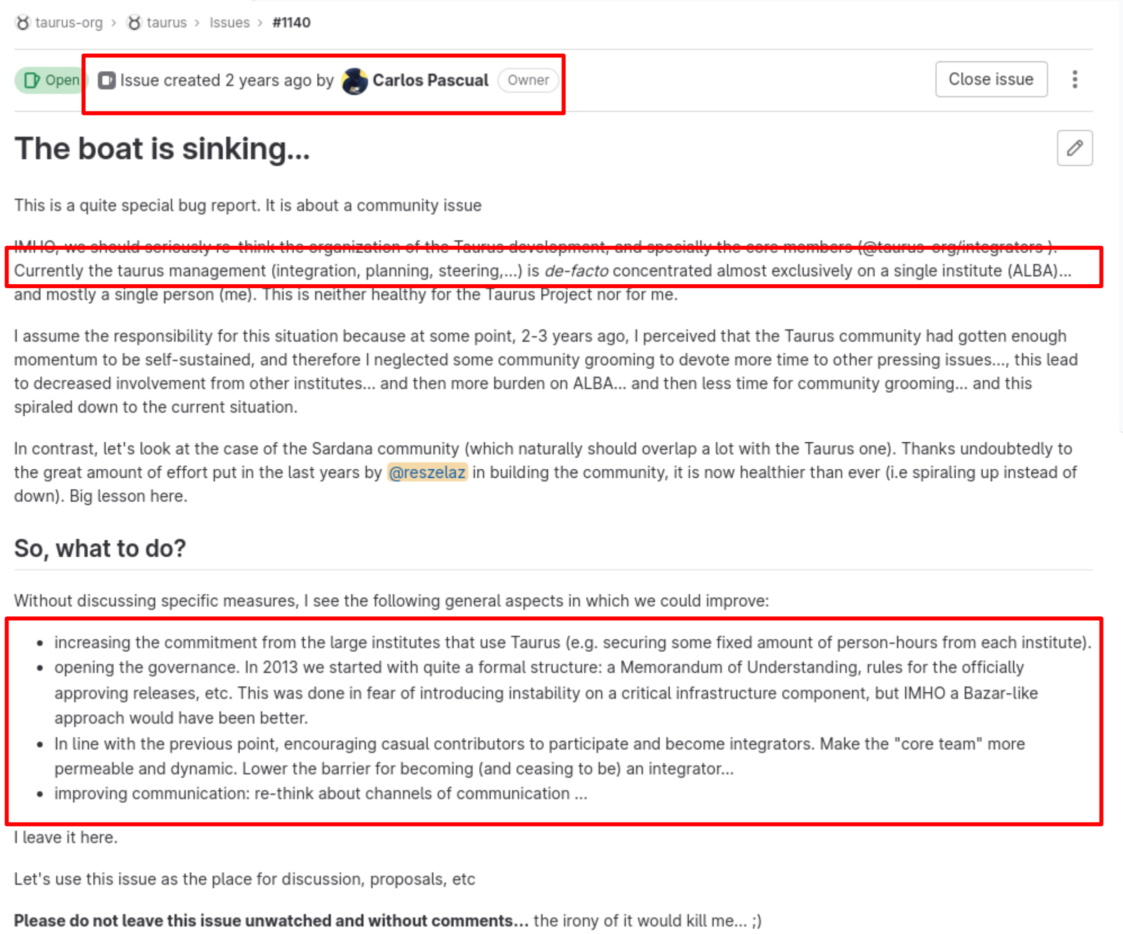
14-15/03/2023

Taurus Workshop; ESRF

Agenda

- Current Situation
- Good things Taurus has to start collaborating
- Learn from successful communities

Current situation



taurus-org › taurus › Issues › #1140

Open | Issue created 2 years ago by Carlos Pascual Owner | Close issue

The boat is sinking...

This is a quite special bug report. It is about a community issue

IMHO, we should seriously re-think the organization of the Taurus development, and specially the core members (@taurus-org/integrators). Currently the taurus management (integration, planning, steering,...) is *de-facto* concentrated almost exclusively on a single institute (ALBA)... and mostly a single person (me). This is neither healthy for the Taurus Project nor for me.

I assume the responsibility for this situation because at some point, 2-3 years ago, I perceived that the Taurus community had gotten enough momentum to be self-sustained, and therefore I neglected some community grooming to devote more time to other pressing issues..., this lead to decreased involvement from other institutes... and then more burden on ALBA... and then less time for community grooming... and this spiraled down to the current situation.

In contrast, let's look at the case of the Sardana community (which naturally should overlap a lot with the Taurus one). Thanks undoubtedly to the great amount of effort put in the last years by @reszelaz in building the community, it is now healthier than ever (i.e spiraling up instead of down). Big lesson here.

So, what to do?

Without discussing specific measures, I see the following general aspects in which we could improve:

- increasing the commitment from the large institutes that use Taurus (e.g. securing some fixed amount of person-hours from each institute).
- opening the governance. In 2013 we started with quite a formal structure: a Memorandum of Understanding, rules for the officially approving releases, etc. This was done in fear of introducing instability on a critical infrastructure component, but IMHO a Bazar-like approach would have been better.
- In line with the previous point, encouraging casual contributors to participate and become integrators. Make the "core team" more permeable and dynamic. Lower the barrier for becoming (and ceasing to be) an integrator...
- improving communication: re-think about channels of communication ...

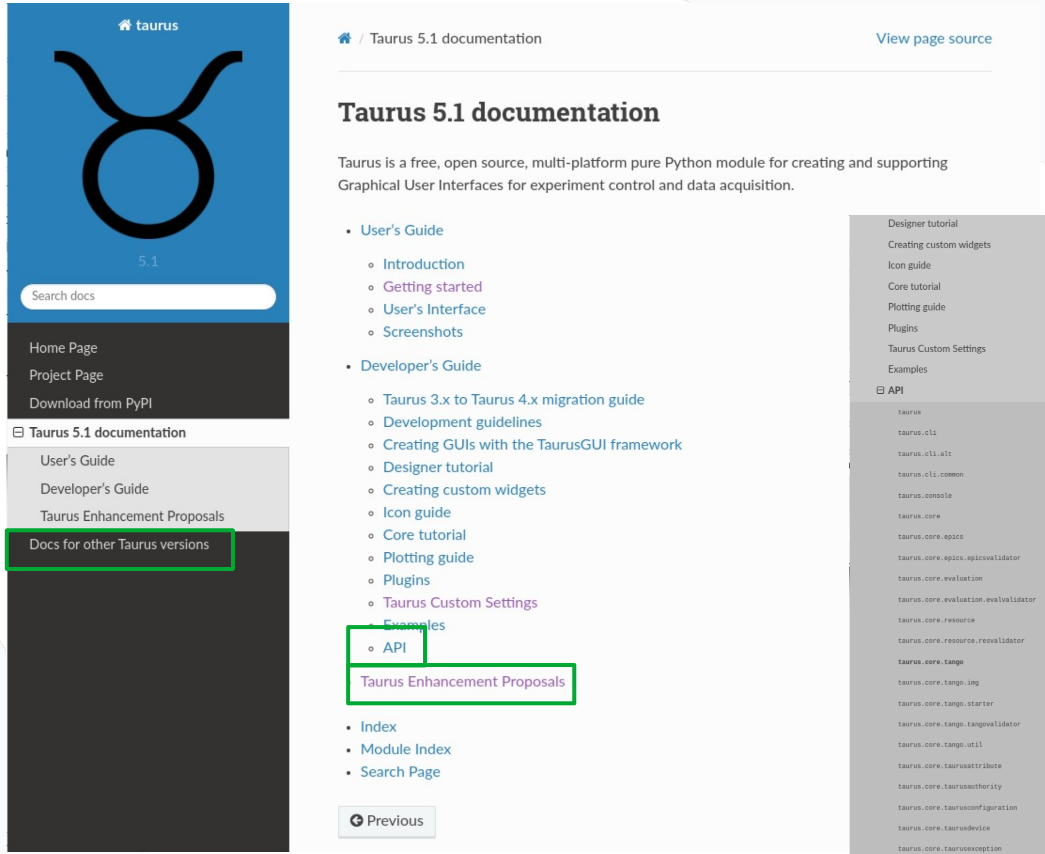
I leave it here.

Let's use this issue as the place for discussion, proposals, etc

Please do not leave this issue unwatched and without comments... the irony of it would kill me... ;)

Good things Taurus already has to start collaborating

Good documentation



taurus

5.1

Search docs

Home Page
Project Page
Download from PyPI

⊟ Taurus 5.1 documentation

User's Guide
Developer's Guide
Taurus Enhancement Proposals

Docs for other Taurus versions

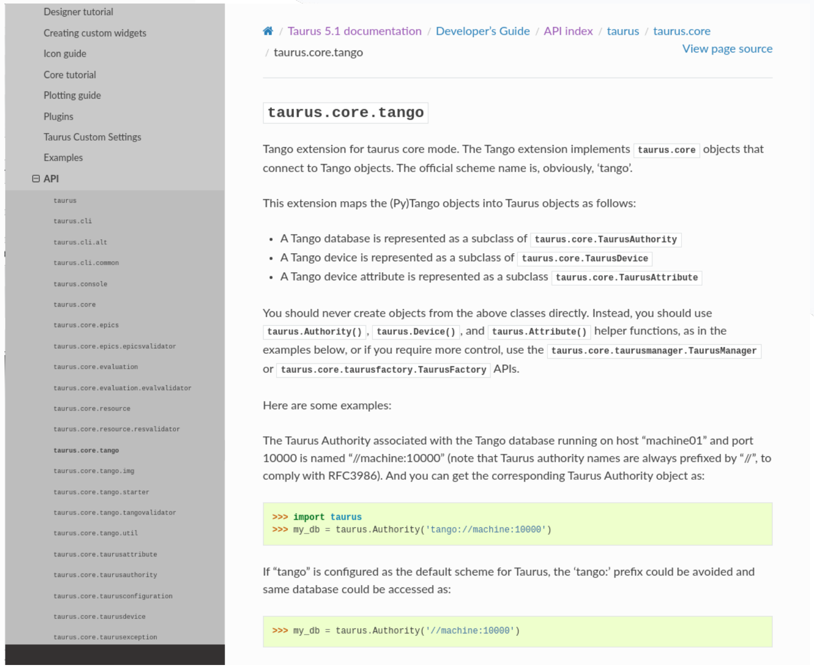
Taurus 5.1 documentation — View page source

Taurus 5.1 documentation

Taurus is a free, open source, multi-platform pure Python module for creating and supporting Graphical User Interfaces for experiment control and data acquisition.

- User's Guide
  - Introduction
  - Getting started
  - User's Interface
  - Screenshots
- Developer's Guide
  - Taurus 3.x to Taurus 4.x migration guide
  - Development guidelines
  - Creating GUIs with the TaurusGUI framework
  - Designer tutorial
  - Creating custom widgets
  - Icon guide
  - Core tutorial
  - Plotting guide
  - Plugins
  - Taurus Custom Settings
  - Examples
  - API
- Taurus Enhancement Proposals
- Index
- Module Index
- Search Page

Previous



Designer tutorial
Creating custom widgets
Icon guide
Core tutorial
Plotting guide
Plugins
Taurus Custom Settings
Examples
⊟ API
taurus
taurus.cli
taurus.cli.alt
taurus.cli.common
taurus.console
taurus.core
taurus.core.epics
taurus.core.epics.epicsvalidator
taurus.core.evaluation
taurus.core.evaluation.evalvalidator
taurus.core.resource
taurus.core.resource.resvalidator
taurus.core.tango
taurus.core.tango.img
taurus.core.tango.starter
taurus.core.tango.tangovalidator
taurus.core.tango.util
taurus.core.taurusattribute
taurus.core.taurusauthority
taurus.core.taurusconfiguration
taurus.core.taurusdevice
taurus.core.taurusexception

Taurus 5.1 documentation / Developer's Guide / API index / taurus / taurus.core / taurus.core.tango — View page source

`taurus.core.tango`

Tango extension for taurus core mode. The Tango extension implements `taurus.core` objects that connect to Tango objects. The official scheme name is, obviously, 'tango'.

This extension maps the (Py)Tango objects into Taurus objects as follows:

- A Tango database is represented as a subclass of `taurus.core.TaurusAuthority`
- A Tango device is represented as a subclass of `taurus.core.TaurusDevice`
- A Tango device attribute is represented as a subclass `taurus.core.TaurusAttribute`

You should never create objects from the above classes directly. Instead, you should use `taurus.Authority()`, `taurus.Device()`, and `taurus.Attribute()` helper functions, as in the examples below, or if you require more control, use the `taurus.core.taurusmanager.TaurusManager` or `taurus.core.taurusfactory.TaurusFactory` APIs.

Here are some examples:

The Taurus Authority associated with the Tango database running on host "machine01" and port 10000 is named "//machine:10000" (note that Taurus authority names are always prefixed by "//", to comply with RFC3986). And you can get the corresponding Taurus Authority object as:

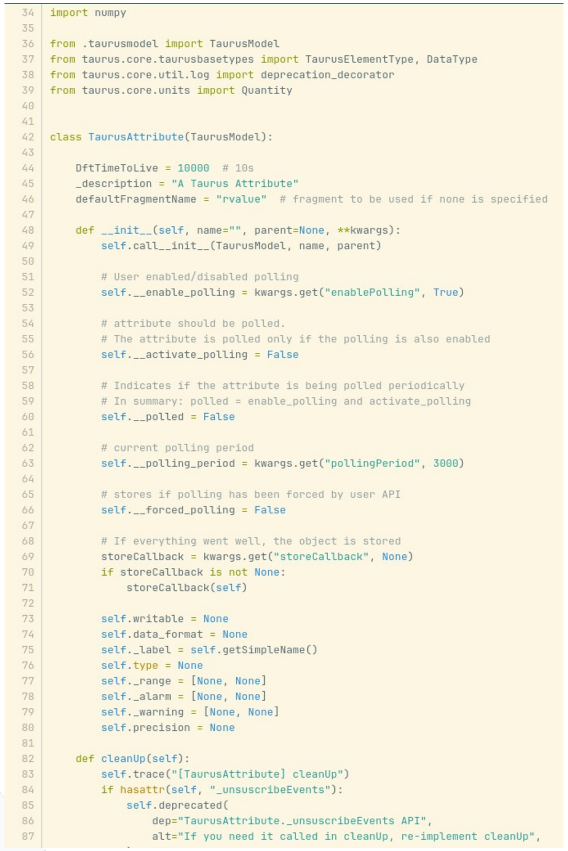
```
>>> import taurus
>>> my_db = taurus.Authority('tango://machine:10000')
```

If "tango" is configured as the default scheme for Taurus, the 'tango:' prefix could be avoided and same database could be accessed as:

```
>>> my_db = taurus.Authority('//machine:10000')
```

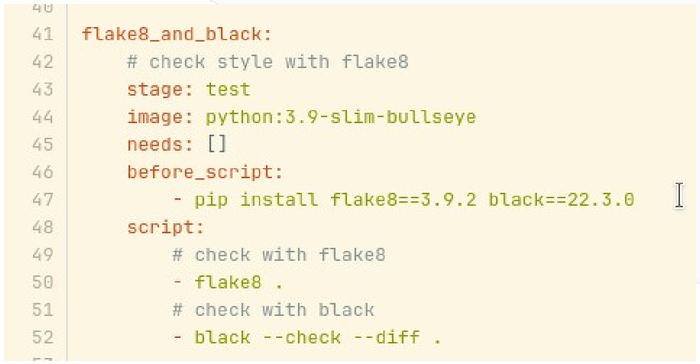
Well maintained and standardized code

Intuitive API



```python
import numpy

from .taurusmodel import TaurusModel
from taurus.core.taurusbasetypes import TaurusElementType, DataType
from taurus.core.util.log import deprecation_decorator
from taurus.core.units import Quantity


class TaurusAttribute(TaurusModel):

    DftTimeToLive = 10000  # 10s
    _description = "A Taurus Attribute"
    defaultFragmentName = "rvalue"  # fragment to be used if none is specified

    def __init__(self, name="", parent=None, **kwargs):
        self.call__init__(TaurusModel, name, parent)

        # User enabled/disabled polling
        self.__enable_polling = kwargs.get("enablePolling", True)

        # attribute should be polled.
        # The attribute is polled only if the polling is also enabled
        self.__activate_polling = False

        # Indicates if the attribute is being polled periodically
        # In summary: polled = enable_polling and activate_polling
        self.__polled = False

        # current polling period
        self.__polling_period = kwargs.get("pollingPeriod", 3000)

        # stores if polling has been forced by user API
        self.__forced_polling = False

        # If everything went well, the object is stored
        storeCallback = kwargs.get("storeCallback", None)
        if storeCallback is not None:
            storeCallback(self)

        self.writable = None
        self.data_format = None
        self._label = self.getSimpleName()
        self.type = None
        self._range = [None, None]
        self._alarm = [None, None]
        self._warning = [None, None]
        self.precision = None

    def cleanUp(self):
        self.trace("[TaurusAttribute] cleanUp")
        if hasattr(self, "_unsuscribeEvents"):
            self.deprecated(
                dep="TaurusAttribute._unsuscribeEvents API",
                alt="If you need it called in cleanUp, re-implement cleanUp",
```

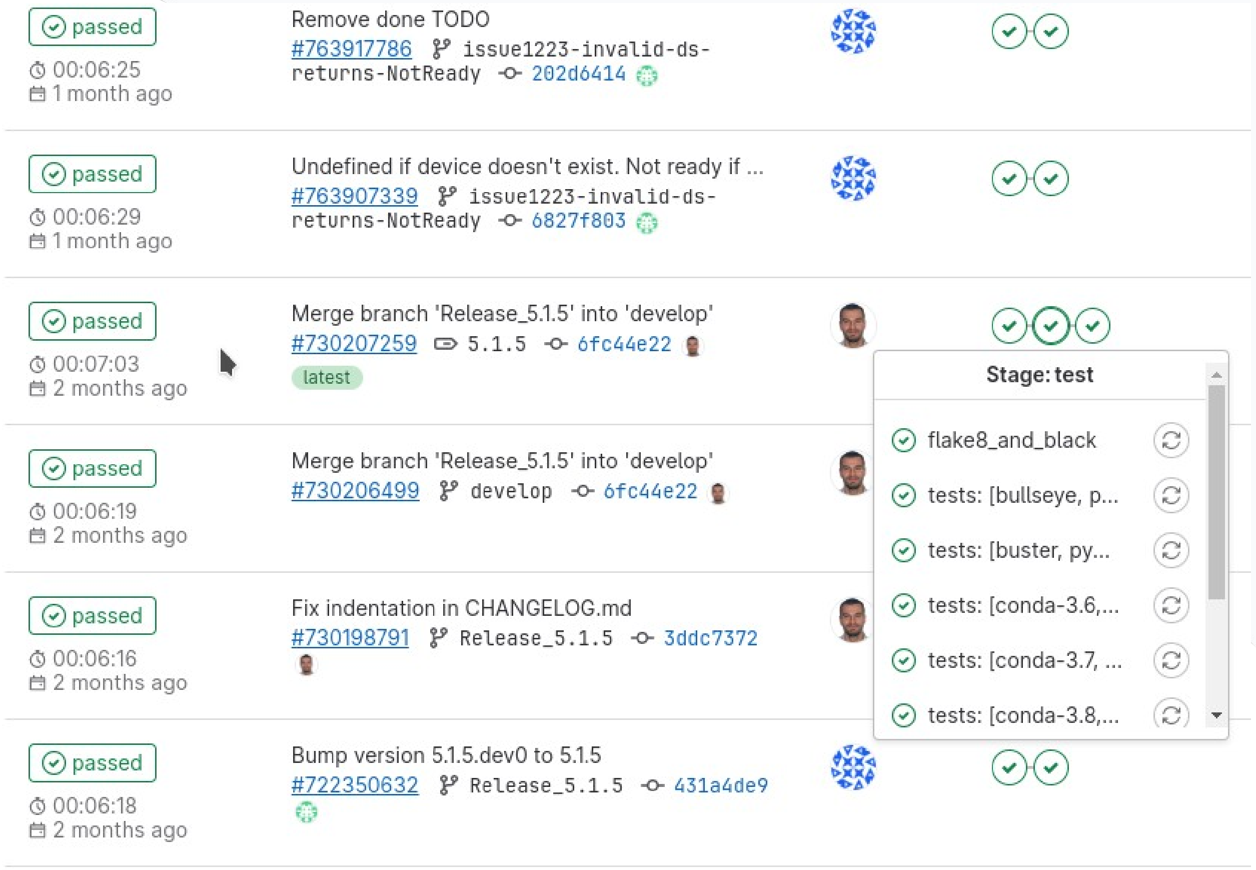
Flake8 and Black



```yaml
flake8_and_black:
    # check style with flake8
    stage: test
    image: python:3.9-slim-bullseye
    needs: []
    before_script:
        - pip install flake8==3.9.2 black==22.3.0
    script:
        # check with flake8
        - flake8 .
        # check with black
        - black --check --diff .
```

Good test coverage (core) and CI pipelines

- Tests:
  - Unittest and PyTest
  - DeviceTestContext
- Docker images
- Testing on different Python versions using Conda
- Upload to PyPI



passed
00:06:25
1 month ago
Remove done TODO
#763917786 issue1223-invalid-ds-returns-NotReady 202d6414

passed
00:06:29
1 month ago
Undefined if device doesn't exist. Not ready if ...
#763907339 issue1223-invalid-ds-returns-NotReady 6827f803

passed
00:07:03
2 months ago
Merge branch 'Release_5.1.5' into 'develop'
#730207259 5.1.5 6fc44e22
latest

passed
00:06:19
2 months ago
Merge branch 'Release_5.1.5' into 'develop'
#730206499 develop 6fc44e22

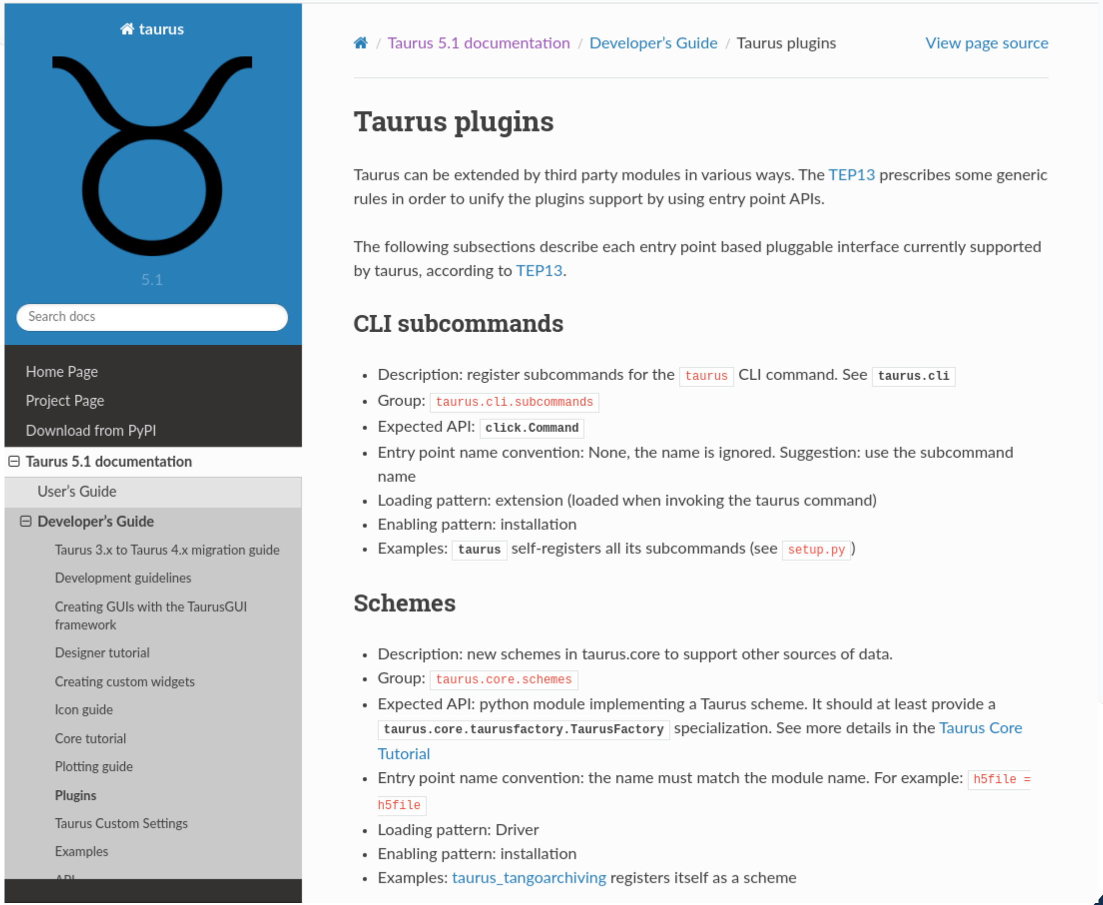
passed
00:06:16
2 months ago
Fix indentation in CHANGELOG.md
#730198791 Release_5.1.5 3ddc7372

passed
00:06:18
2 months ago
Bump version 5.1.5.dev0 to 5.1.5
#722350632 Release_5.1.5 431a4de9

Stage: test
- flake8_and_black
- tests: [bullseye, p...
- tests: [buster, py...
- tests: [conda-3.6,...
- tests: [conda-3.7, ...
- tests: [conda-3.8,...

Plugin system

- Using setuptools entry-points
- CLI subcommands
- Schemes
- Formatters
- Model Chooser Selectors
- Widget Alternatives
- Form Factories
- qtgui submodules



Taurus plugins

Taurus can be extended by third party modules in various ways. The TEP13 prescribes some generic rules in order to unify the plugins support by using entry point APIs.

The following subsections describe each entry point based pluggable interface currently supported by taurus, according to TEP13.

CLI subcommands

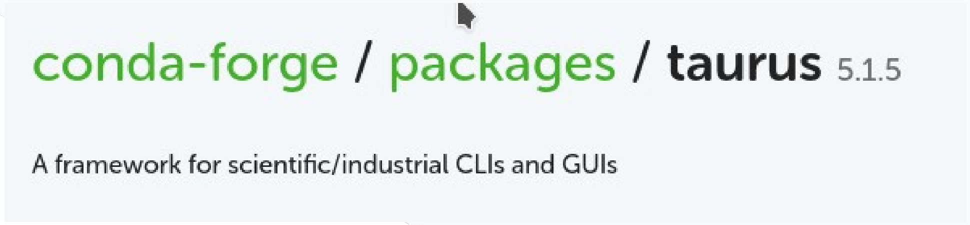
- Description: register subcommands for the `taurus` CLI command. See `taurus.cli`
- Group: `taurus.cli.subcommands`
- Expected API: `click.Command`
- Entry point name convention: None, the name is ignored. Suggestion: use the subcommand name
- Loading pattern: extension (loaded when invoking the taurus command)
- Enabling pattern: installation
- Examples: `taurus` self-registers all its subcommands (see `setup.py`)

Schemes

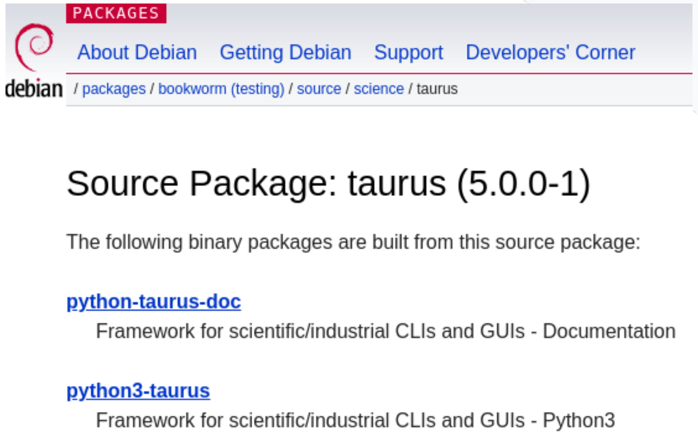
- Description: new schemes in taurus.core to support other sources of data.
- Group: `taurus.core.schemes`
- Expected API: python module implementing a Taurus scheme. It should at least provide a `taurus.core.taurusfactory.TaurusFactory` specialization. See more details in the Taurus Core Tutorial
- Entry point name convention: the name must match the module name. For example: `h5file = h5file`
- Loading pattern: Driver
- Enabling pattern: installation
- Examples: taurus_tangoarchiving registers itself as a scheme

Packaging

- PyPI
- Conda-forge
  thanks to Benjamin Bertrand (MAXIV)
  - taurus
  - taurus-qt
  - taurus-core
  - taurus_pyqtgraph
- Debian
  thanks to Frederic Picca and Carlos Pascual


conda-forge / packages / taurus 5.1.5

A framework for scientific/industrial CLIs and GUIs


PACKAGES
About Debian Getting Debian Support Developers' Corner
/ packages / bookworm (testing) / source / science / taurus

Source Package: taurus (5.0.0-1)

The following binary packages are built from this source package:

python-taurus-doc
Framework for scientific/industrial CLIs and GUIs - Documentation

python3-taurus
Framework for scientific/industrial CLIs and GUIs - Python3

Learn from successful communities

TANGO controls

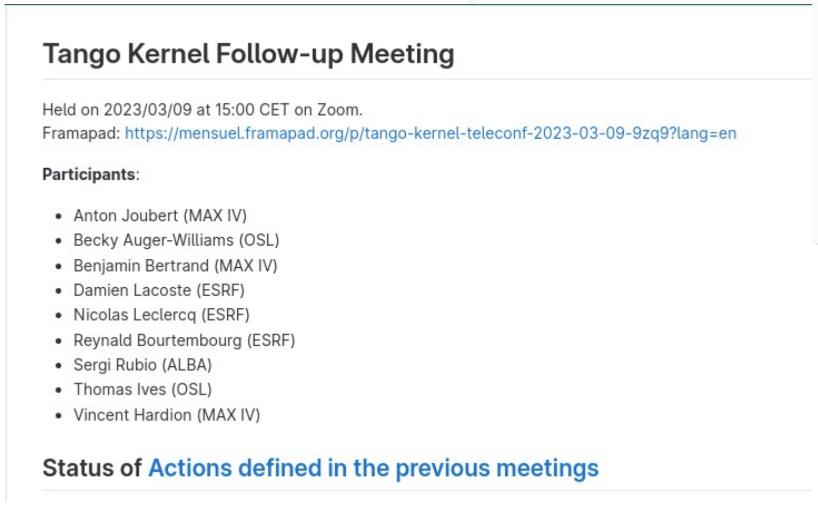
TARANTA
TANGO ON WEB

sardana

14-15/03/2023

Regular follow-up meetings

- Organize regular follow-up meetings e.g. 1 per month
  - Identify core institutes that want to participate
  - Ask for contact person from these institutes
  - ALBA is willing to mentor any contact persons if needed



Tango Kernel Follow-up Meeting

Held on 2023/03/09 at 15:00 CET on Zoom.
Framapad: https://mensuel.framapad.org/p/tango-kernel-teleconf-2023-03-09-9zq9?lang=en

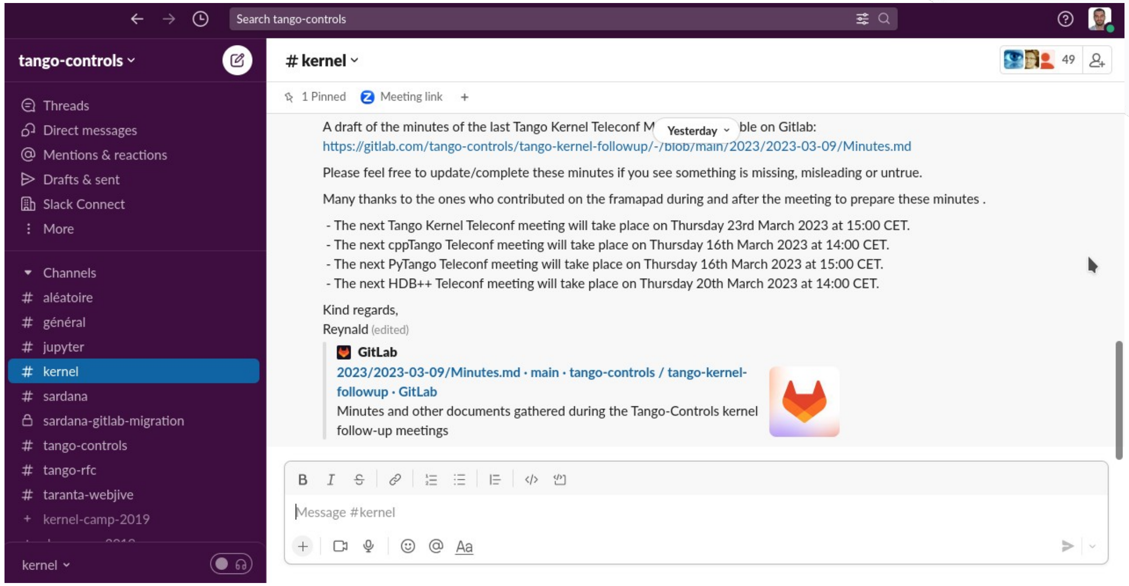
Participants:

- Anton Joubert (MAX IV)
- Becky Auger-Williams (OSL)
- Benjamin Bertrand (MAX IV)
- Damien Lacoste (ESRF)
- Nicolas Leclercq (ESRF)
- Reynald Bourtembourg (ESRF)
- Sergi Rubio (ALBA)
- Thomas Ives (OSL)
- Vincent Hardion (MAX IV)

Status of Actions defined in the previous meetings

Encourage IM/tele-conferences

- Promote IM/tele-conferences between contact persons for specific topics if this will speed up issue resolution
  - ...but continue documenting conclusions on GitLab issues – knowledge sharing



A draft of the minutes of the last Tango Kernel Teleconf M ble on Gitlab:
https://gitlab.com/tango-controls/tango-kernel-followup/-/blob/main/2023/2023-03-09/Minutes.md

Please feel free to update/complete these minutes if you see something is missing, misleading or untrue.

Many thanks to the ones who contributed on the framapad during and after the meeting to prepare these minutes .

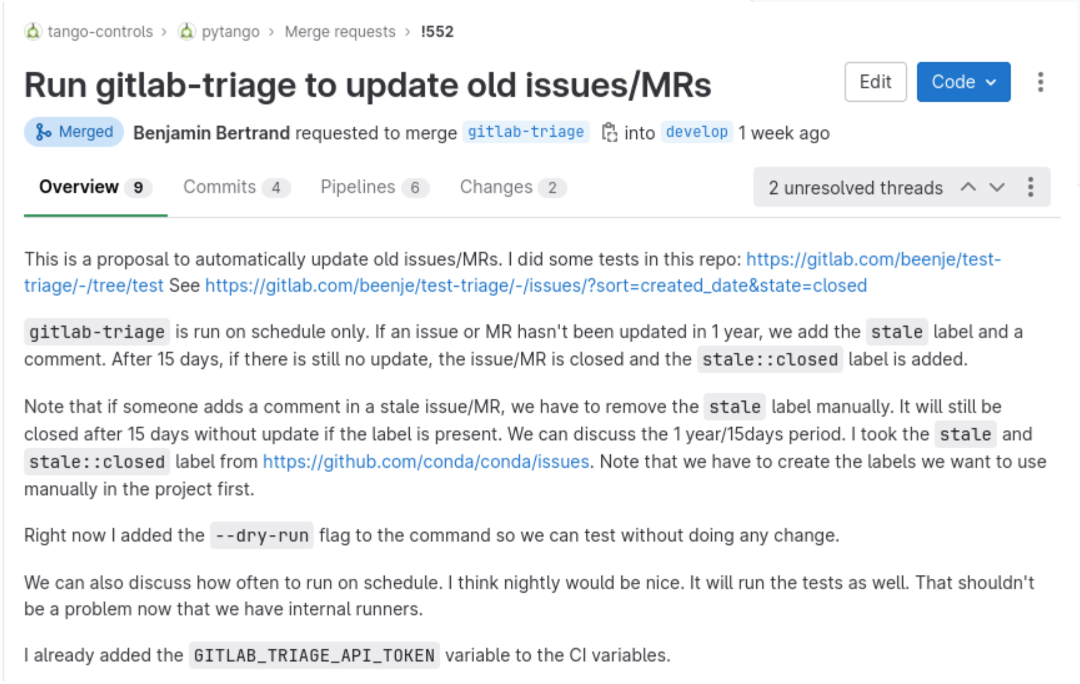
- The next Tango Kernel Teleconf meeting will take place on Thursday 23rd March 2023 at 15:00 CET.
- The next cppTango Teleconf meeting will take place on Thursday 16th March 2023 at 14:00 CET.
- The next PyTango Teleconf meeting will take place on Thursday 16th March 2023 at 15:00 CET.
- The next HDB++ Teleconf meeting will take place on Thursday 20th March 2023 at 14:00 CET.

Kind regards,
Reynald (edited)

GitLab
2023/2023-03-09/Minutes.md · main · tango-controls / tango-kernel-followup · GitLab
Minutes and other documents gathered during the Tango-Controls kernel follow-up meetings

Backlog of issues

- Review backlog of issues (> 200) and close stale issues:
  - PyTango vs. Sardana strategy



Run gitlab-triage to update old issues/MRs

Merged Benjamin Bertrand requested to merge `gitlab-triage` into `develop` 1 week ago

This is a proposal to automatically update old issues/MRs. I did some tests in this repo: https://gitlab.com/beenje/test-triage/-/tree/test See https://gitlab.com/beenje/test-triage/-/issues/?sort=created_date&state=closed

`gitlab-triage` is run on schedule only. If an issue or MR hasn't been updated in 1 year, we add the `stale` label and a comment. After 15 days, if there is still no update, the issue/MR is closed and the `stale::closed` label is added.

Note that if someone adds a comment in a stale issue/MR, we have to remove the `stale` label manually. It will still be closed after 15 days without update if the label is present. We can discuss the 1 year/15days period. I took the `stale` and `stale::closed` label from https://github.com/conda/conda/issues. Note that we have to create the labels we want to use manually in the project first.

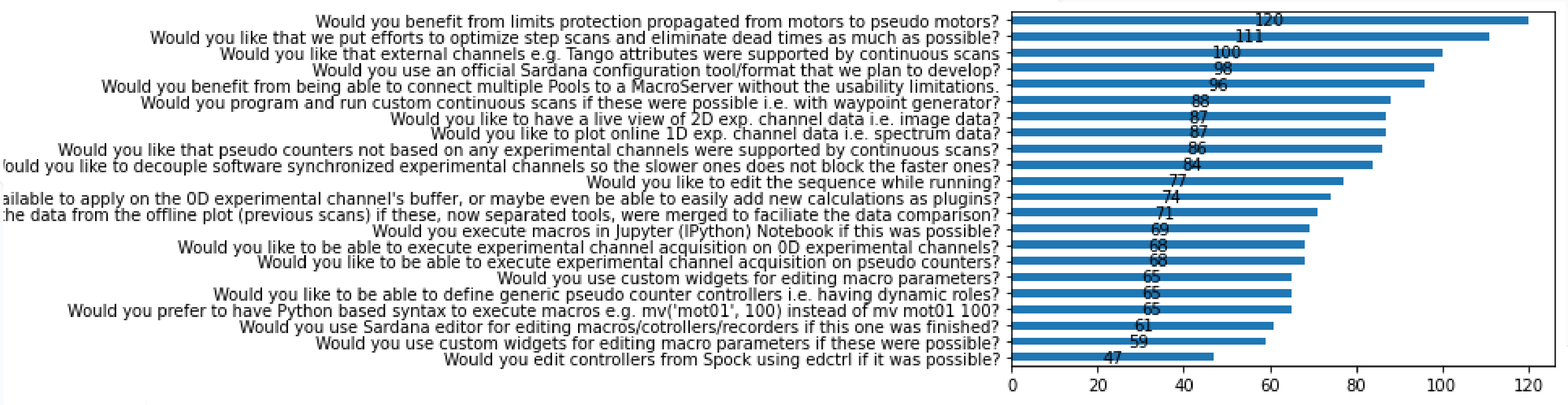
Right now I added the `--dry-run` flag to the command so we can test without doing any change.

We can also discuss how often to run on schedule. I think nightly would be nice. It will run the tests as well. That shouldn't be a problem now that we have internal runners.

I already added the `GITLAB_TRIAGE_API_TOKEN` variable to the CI variables.

Define Roadmap

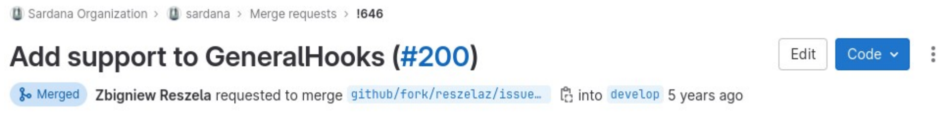
- Identify the most popular major bugs/enhancements
  - Users questionnaire



Would you benefit from limits protection propagated from motors to pseudo motors?	120
Would you like that we put efforts to optimize step scans and eliminate dead times as much as possible?	111
Would you like that external channels e.g. Tango attributes were supported by continuous scans	100
Would you use an official Sardana configuration tool/format that we plan to develop?	98
Would you benefit from being able to connect multiple Pools to a MacroServer without the usability limitations.	96
Would you program and run custom continuous scans if these were possible i.e. with waypoint generator?	88
Would you like to have a live view of 2D exp. channel data i.e. image data?	87
Would you like to plot online 1D exp. channel data i.e. spectrum data?	87
Would you like that pseudo counters not based on any experimental channels were supported by continuous scans?	86
'ould you like to decouple software synchronized experimental channels so the slower ones does not block the faster ones?	84
Would you like to edit the sequence while running?	77
ailable to apply on the 0D experimental channel's buffer, or maybe even be able to easily add new calculations as plugins?	74
:he data from the offline plot (previous scans) if these, now separated tools, were merged to faciliate the data comparison?	71
Would you execute macros in Jupyter (IPython) Notebook if this was possible?	69
Would you like to be able to execute experimental channel acquisition on 0D experimental channels?	68
Would you like to be able to execute experimental channel acquisition on pseudo counters?	68
Would you use custom widgets for editing macro parameters?	65
Would you like to be able to define generic pseudo counter controllers i.e. having dynamic roles?	65
Would you prefer to have Python based syntax to execute macros e.g. mv('mot01', 100) instead of mv mot01 100?	65
Would you use Sardana editor for editing macros/cotrollers/recorders if this one was finished?	61
Would you use custom widgets for editing macro parameters if these were possible?	59
Would you edit controllers from Spock using edctrl if it was possible?	47

Inter-institute developments

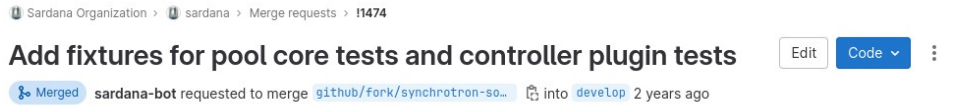
- Organize Inter-institute developments/hackathons/meetings



Sardana Organization › sardana › Merge requests › !646

Add support to GeneralHooks (#200)

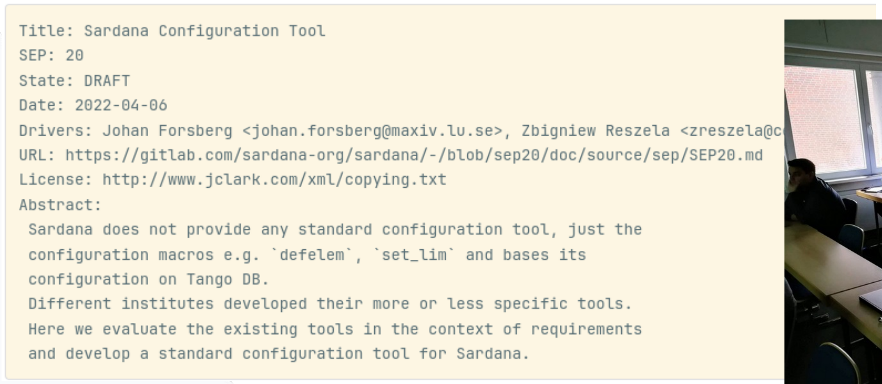
Merged **Zbigniew Reszela** requested to merge `github/fork/reszelaz/issue…` into `develop` 5 years ago



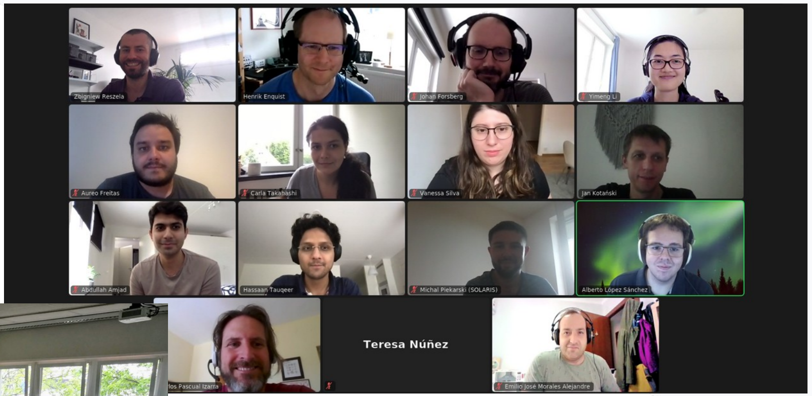
Sardana Organization › sardana › Merge requests › !1474

Add fixtures for pool core tests and controller plugin tests

Merged **sardana-bot** requested to merge `github/fork/synchrotron-so…` into `develop` 2 years ago

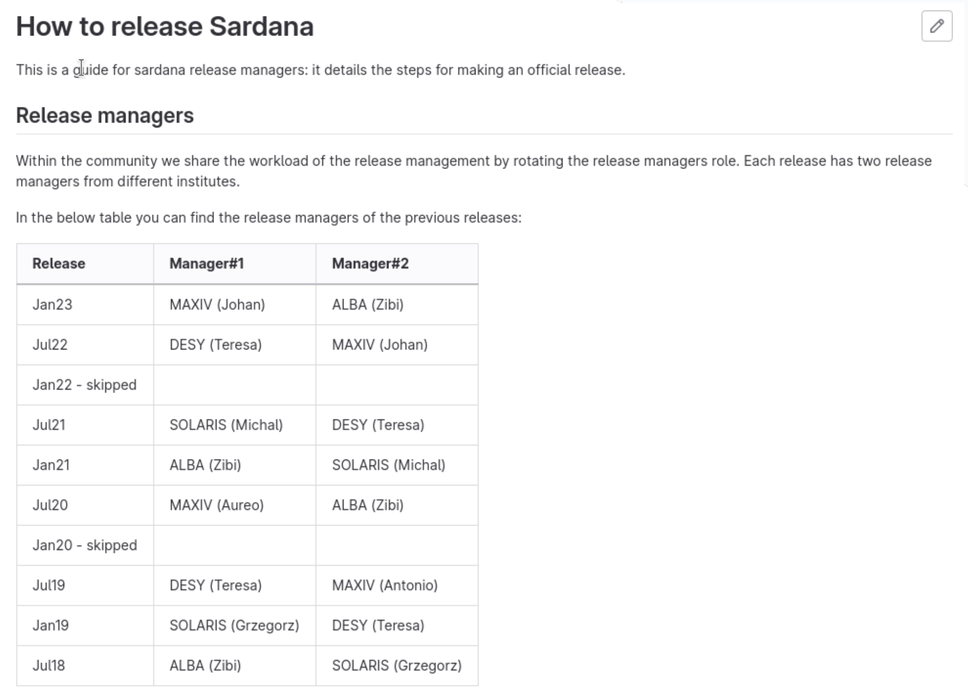


```
Title: Sardana Configuration Tool
SEP: 20
State: DRAFT
Date: 2022-04-06
Drivers: Johan Forsberg <johan.forsberg@maxiv.lu.se>, Zbigniew Reszela <zreszela@c
URL: https://gitlab.com/sardana-org/sardana/-/blob/sep20/doc/source/sep/SEP20.md
License: http://www.jclark.com/xml/copying.txt
Abstract:
 Sardana does not provide any standard configuration tool, just the
 configuration macros e.g. `defelem`, `set_lim` and bases its
 configuration on Tango DB.
 Different institutes developed their more or less specific tools.
 Here we evaluate the existing tools in the context of requirements
 and develop a standard configuration tool for Sardana.
```



Release management

- Distribute release management work



How to release Sardana

This is a guide for sardana release managers: it details the steps for making an official release.

Release managers

Within the community we share the workload of the release management by rotating the release managers role. Each release has two release managers from different institutes.

In the below table you can find the release managers of the previous releases:

Release	Manager#1	Manager#2
Jan23	MAXIV (Johan)	ALBA (Zibi)
Jul22	DESY (Teresa)	MAXIV (Johan)
Jan22 - skipped		
Jul21	SOLARIS (Michal)	DESY (Teresa)
Jan21	ALBA (Zibi)	SOLARIS (Michal)
Jul20	MAXIV (Aureo)	ALBA (Zibi)
Jan20 - skipped		
Jul19	DESY (Teresa)	MAXIV (Antonio)
Jan19	SOLARIS (Grzegorz)	DESY (Teresa)
Jul18	ALBA (Zibi)	SOLARIS (Grzegorz)

Conclusions

- Taurus project needs to be revived
  - ALBA is now training newcomers to take over the development and maintenance of Taurus and we hope to revive also the Community
- Taurus has a lot of good things that will facilitate collaboration
- Many practices from other successful communities proved to be useful
- If you use Taurus or you are interested in it, you want to collaborate exchange and participate in the project then let’s do it together!

Thank You

Good things to facilitate collaboration:

- Good docs
- Well maintained code
- Good tests
- Plugin system
- Packaging

How to revive the community:

- Regular follow-up meetings
- IM/ specific tele-conferences
- Refine backlog
- Define roadmap
- Inter-institute developments, hackathons, meetings