Panic v10 : new features and collaboration

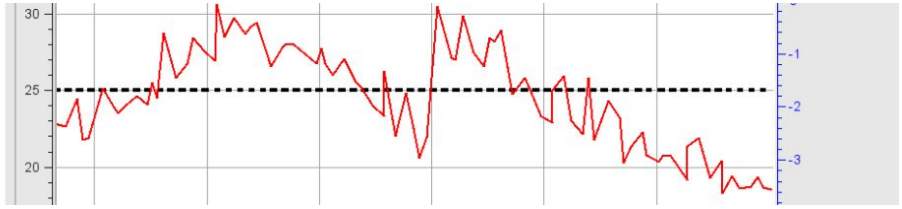
Sergi Rubio Manrique - ALBA Synchrotron

What is PANIC?

PANIC is the Alarm System developed at ALBA to detect and notify abnormal conditions in our controls systems.

The PyAlarm Tango Device Server, which is the core of the PANIC system, was developed in 2008 to provide remote monitoring of ALBA during installation; while engineers and technicians offices where several kilometers far from building site.

PyAlarm simply reads an Alarm Formula, Description and Receivers from the configuration. Then evaluates the formula, given a pre-defined conditions. If the formula activates the alarm, the alarm description is sent to the receivers.



if my/plc/01/Temperature **>** SETPOINT then :

PyAlarm has been later extended by PANIC API and PANIC GUI to provide extended Alarm capabilities for all ALBA accelerators and laboratories, providing an scalable system with multiple notification services, user authentication, logging, ...

Current Alarm Systems at ALBA provide **1214 Alarms from 28 different Tango Control Systems** (Accelerators, Beamlines, Laboratories, Infrastructures and Cooling facilities and IT Data Center), checking **2568 attributes** at periods between 500 ms and 4h. It is used by many other institutes within the TANGO Community like **ESRF, Max IV, Soleil, Solaris, SKAO,** ...

What is an Alarm System?

According to **IEC 62682:2014** "Management of Alarm Systems for the Process Industries" definition

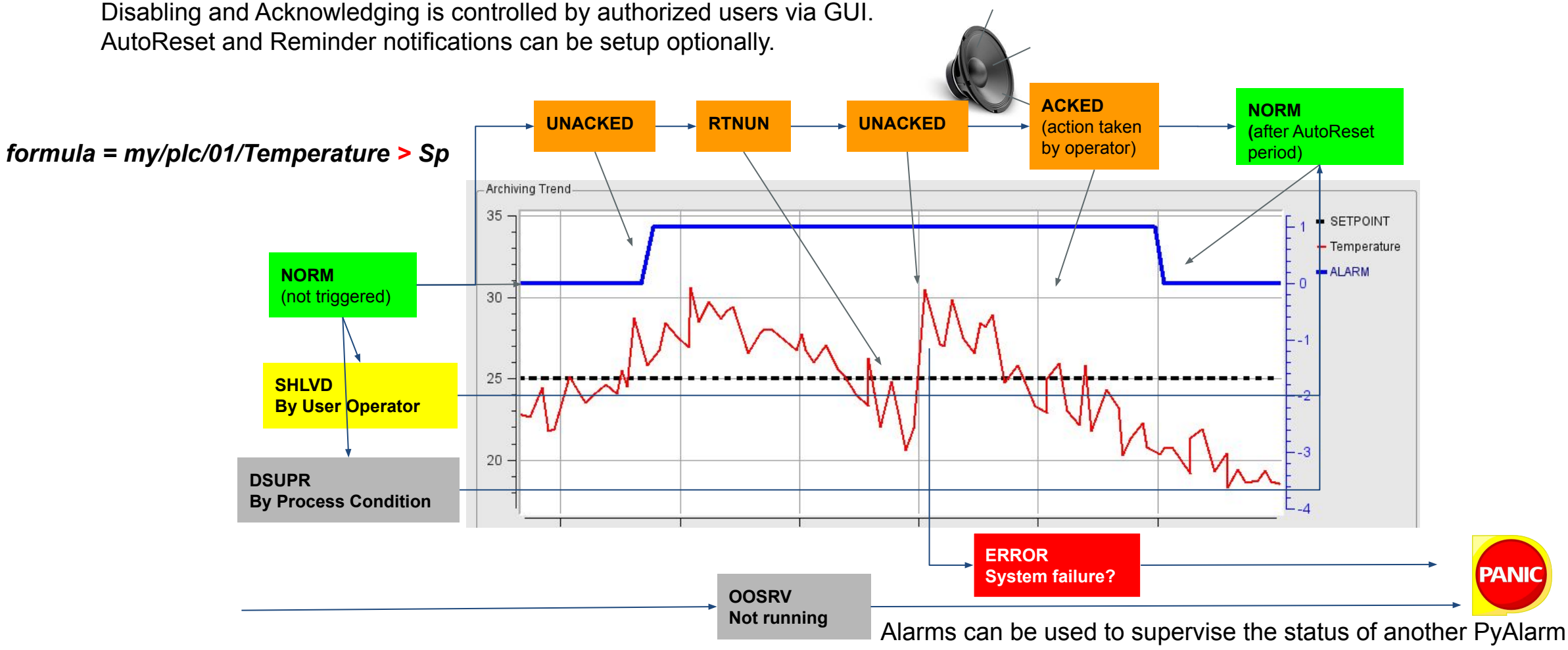
- The primary function of an Alarm System will be to notify abnormal process conditions or equipment malfunctions and support the operator response.

- The Alarm System is NOT part of the protection nor safety systems. Safety and Protection Systems are separate systems following its own regulation.

- The Alarm System is part of the Operator Response, thus it's part of the HMI (including the non-graphical part of it).

PANIC Alarm Cycle (IEC62682)

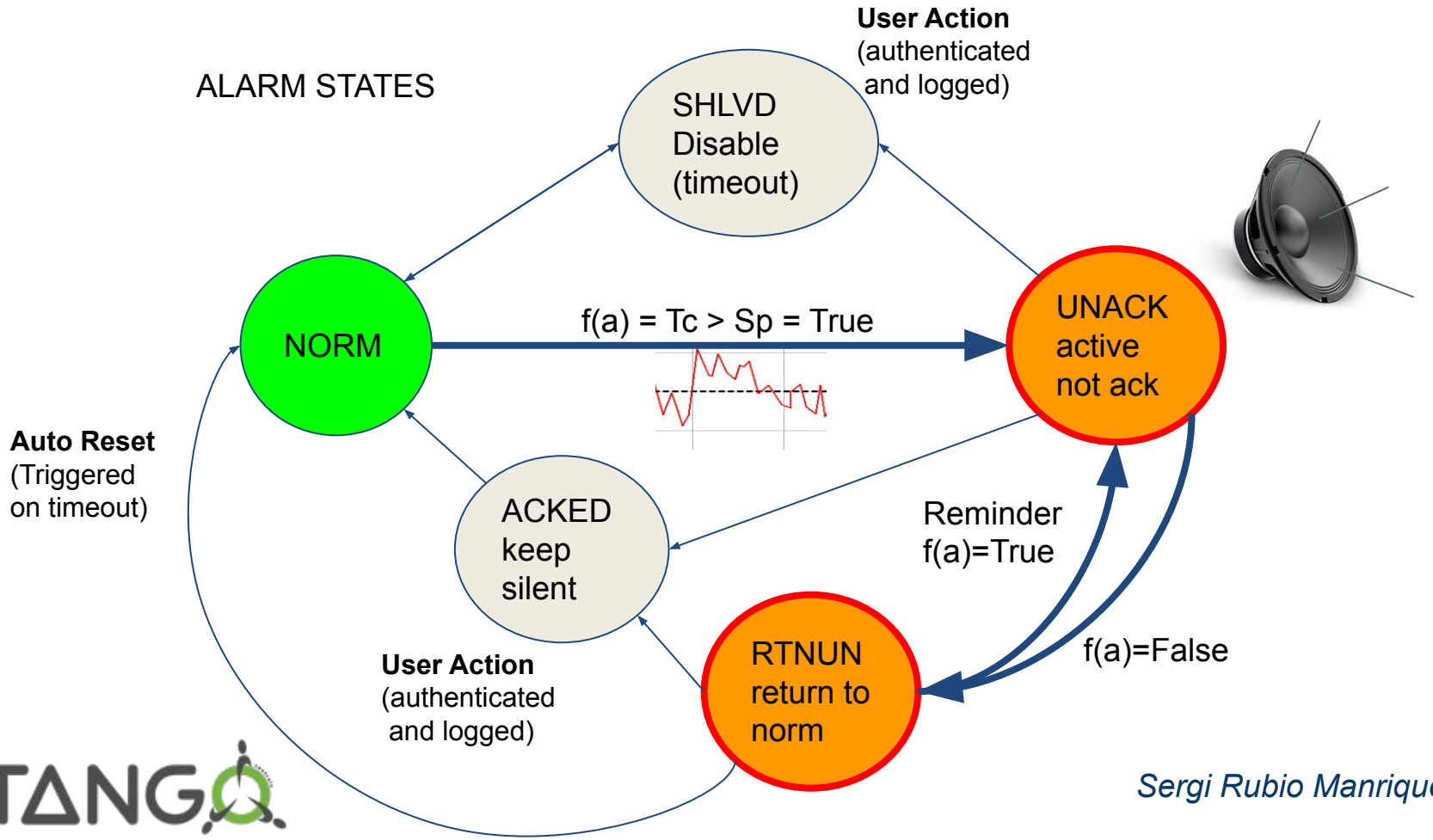
The process of alarm activation and notification is controlled by PyAlarm and Alarm formulas.
Disabling and Acknowledging is controlled by authorized users via GUI.
AutoReset and Reminder notifications can be setup optionally.



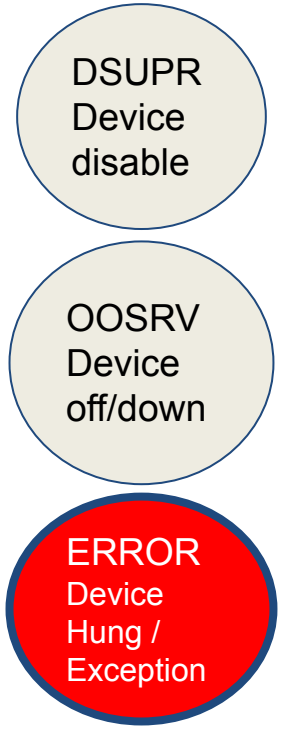
Alarms can be used to supervise the status of another PyAlarm.

PANIC Alarm Cycle (IEC62682)

The process of alarm activation and notification is controlled by PyAlarm and Alarm formulas.
Disabling and Acknowledging is controlled by authorized users via GUI.
AutoReset and Reminder notifications can be setup optionally.



SYSTEM STATES



Alarms can be used to supervise the status of another PyAlarm.

PANIC in the Tango Ecosystem

- PANIC is used by ALBA, ESRF, MaxIV, SKA, Solaris and Soleil amongst other institutes.

- PANIC is built using the **fandango** python library, a set of utilities developed at ALBA to build dynamic python device servers and configure them.

- PANIC **PyAlarm** and the Elettra Tango **AlarmHandler** are complementary tools:
  - **Both are supported** by the PANIC GUI.
  - AlarmHandler can use PyAlarm as a notification service.
  - **C++** AlarmHandler offers speed while **python** PyAlarm offers **flexibility**.

- **S2Innovation** offers its own web-based Alarm System Application, **IC@MS**, build on top of PANIC. Other institutes developed their own PANIC web clients (Achtung!) or other web-based applications.

PANIC Architecture

Basic elements that are part of our Alarm System. Alarms are distributed across multiple PyAlarm servers, each of them handling a sub-group of alarms. This becomes transparent to users thanks to the PANIC API and GUI.

Alarm Evaluators :

PANIC PyAlarm or Alarm Handler Tango device servers, that connect to the control system and evaluate alarm formulas.
As PyAlarm provides evaluation, notification and configuration it is the only element required to build a minimal Alarm System.

Notification Services (Annunciators) :

Additional Tango Device servers executing commands on Alarm Evaluator request (PyAlarm, FestivalDS, ...)

PANIC Alarm GUI:

To provide a graphical interface to users and system administrators. A native Qt client is provided, web-based commercial applications are also available (IC@MS).

Configuration Database :

To store alarm formulas and evaluator configuration, use Tango Database for flexible deployment, or .jive config file for stand-alone service.

Logging Database :

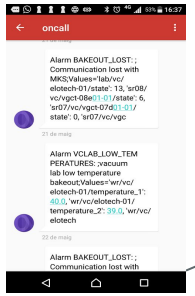
To store activation / deactivation of alarms, alarm conditions and GUI user comments.

PANIC Alarm API:

Python library for automated configuration, data inspection and development of alarm tools.

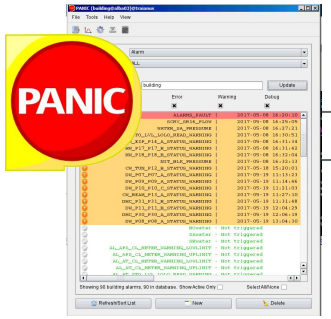
ALBA
www.cells.es

PANIC Architecture



Annunciators
email, speakers, telegram, commands

Alarm GUI

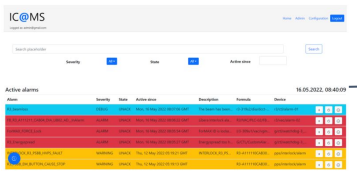


Logging

SNAP/
HDB++

Control System

ic@ms



PANIC
AlarmAPI

Configuration

Alarm Formula Evaluators
(~20 alarms per process)

AlarmHandler
Device

AlarmHandler
Device

Proxy

Proxy

Proxy

PyAlarm
Device

PyAlarm
Device

Proxy

Proxy

Proxy

Tango
Device

Tango
Device

Tango
Device

Tango
Device

Tango
Device

Tango
Device

TANGO DB

Configuration

TANGO

Other control systems other than TANGO can be integrated into PANIC using Taurus library or tango2epics as a bridge.

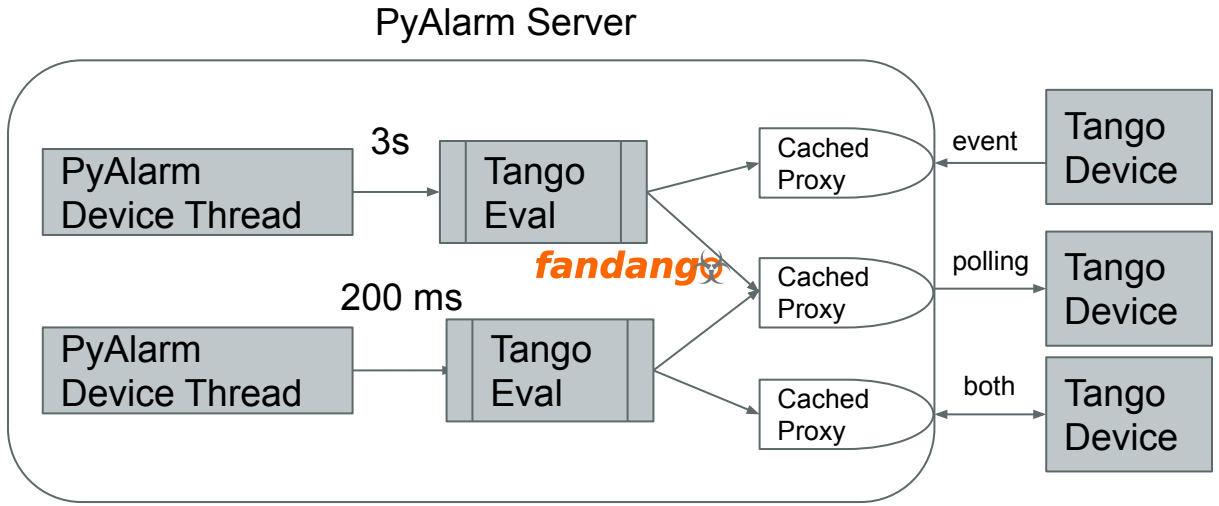
TANGO

PyAlarm Device Server

PyAlarm Device is still working mostly on polling to evaluate the formulas.

Attributes are read either using client-polling (default) or Tango Events and then cached for its reading from formulas. Using Tango Events increases CPU usage, this must be taken into account on balancing the number of attributes/formulas per device (I started tests of an **asyncio** implementation).

New ALARM events are already subscribed in 9.3.13, but still treated as change events (do not trigger immediate alarm if not configured in the formula). This feature requires latest fandango (on staging).



Alarm formulas are exported as *attributes* , and its changes are *annunciated*
So, each PyAlarm is an AlarmSystem on its own!

PyAlarm Event Usage

Middle-layer devices that depend on lower-level devices cannot guarantee an homogeneous read-time per attribute ... thus suffering from many Tango polling exceptions (**Timeout/Serialization/OutOfSync**/...)

PyAlarm avoids these exceptions by executing periodic reading of target attributes in a background thread. Less efficient approach than using events, but robust.

Since Panic 9.1 we tested the evaluation of all alarms triggered asynchronously at each event received, recalculating the alarm value on the fly.

But!, cpu peaked due to the large number of events subscribed. For a beam position alarm (**4*10Hz ADC + 3Hz current**) we got **50Hz** alarm evaluation!!

We replaced on-event evaluation by just event caching plus polling

alarm re-evaluation every 300 ms (fandango.callbacks).

PANIC Alarm Formulas

PANIC Alarms are single-line human readable codes, on which any reference to TANGO attributes, either by direct name or wildcard search, is replaced by its latest acquired value.

explicit alarm:

T1_ALARM = my/plc/01/Temperature **>** 45

alarm on attribute quality:

T2_ALARM = my/plc/02/Temperature.ALARM

alarm on attribute change:

T2_ALARM = my/plc/02/Temperature.delta > 5

composed alarm:

TG_ALARM = T1_ALARM **OR** T2_ALARM

alarm on wildcard search:

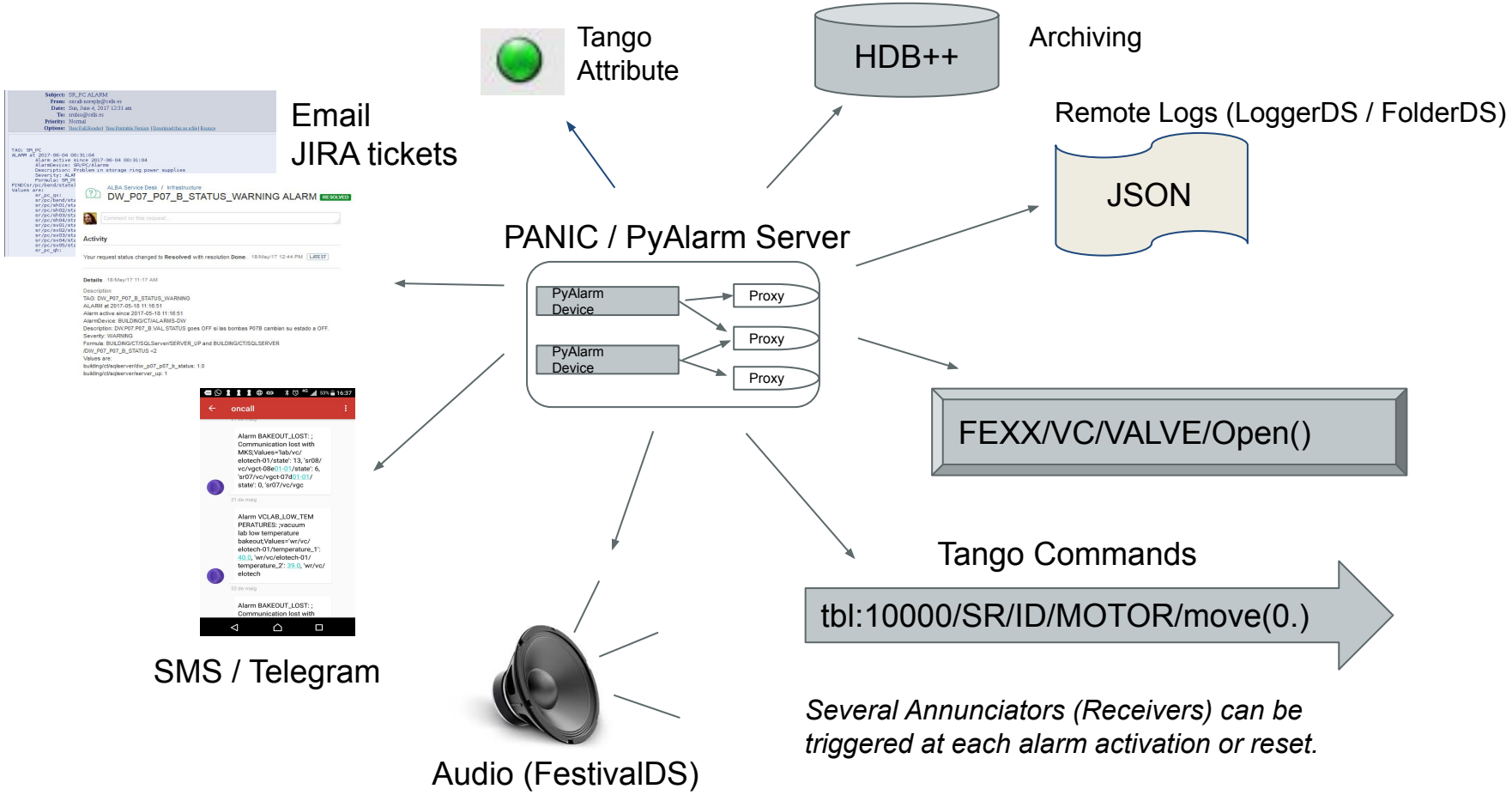
TS_ALARM = **any**(t>SETPOINT for t in **FIND**(*/plc/*/Temperature))

alarm on aggregated values:

TMax_ALARM = **max(FIND**(*/plc/*/Temperature))) > 45

In addition to FIND, alarm formulas allow multiple "macros" that allow to acces previous values history, timestamp, delta.

PANIC Notification Services



PANIC Notification Services

PANIC Provides Several notification services, and allow to easily integrate new annunciators via Tango Commands and Scripts.

- **Email :** Provided by PyAlarm, requires the previous configuration of an SMTP email server or the OS mail application.
- **Telegram:** Provided by PyAlarm, it is aimed at sending Telegram messages to groups via Telegram API.
- **SMS:** Provided by PyAlarm, requires a compatible python smslib library and an online provider of html-to-sms services.
- **Sound and Speech:** Provided by FestivalDS device server, publicly available, it allows to play sounds and text-to-speech conversion on Linux systems using the open-source festival library.
- **Tango Commands:** PyAlarm can execute Tango Commands (if previously registered by PANIC Admin). This allows to, in response to an alarm condition, move motors to home positions, open or close valves, switch off heaters, ...
- **OS Scripts:** PyAlarm can execute OS commands (if previously registered by PANIC Admin). This allows to, in response to an alarm condition, restart processes, cleanup logs, execute an script on the command line ...

Prevent Alarm Flooding! (IEC626862)

The standard requires the Alarm System to be carefully tuned to prevent Alarm Flooding or "operator overload".
This is achieved by carefully adjusting the PyAlarm device configuration.
This is the most complex process of configuring the Alarm System.

Table 2: KPI's[4]EEMUA

KPI	Acceptable value	Flood
Alarms/Day	**12*24*operator**	
Alarms/Hour	12/operator [UM]	>=60
Alarms/10mins	*2/operator*	>=10
Alarm Peaks (N hours with N>30)	**(<1%)**	
(floods) 10mins with >10	*(<1%)*	
TOP 10 recurring alarms	**(<5% of total)**	
Chattering alarms	0	
Stale alarms/day	< 5	
Priority proportion	80,15,5,1 80% of alarms should not require immediate action	
Unauthorized alarm change	0	

PyAlarm Configuration Parameters

Each PyAlarm device is an "evaluation machine" for a list of Alarms, collecting attribute values and recalcullating the formulas and alarm states according a pre-defined configuration; independent for each device.

Amongst others, typical configuration parameters would be:

PollingPeriod: time in seconds at which alarms will be re-evaluated
AlarmThreshold: number of occurrences of each alarm (formula = True) to consider it active (or RTNUN)
AlertOnRecovery: whether to send a new notification when the alarm state returns to normal
Reminder: whether to send a new notification every X seconds or when alarm oscillates
AutoReset: number of seconds after which an alarm in RTNUN goes back to NORM
StartupDelay: number of seconds to wait after a reboot before starting evaluating alarms
Enabled: a pre-condition to enable/disable the whole PyAlarm device
EvalTimeout: maximum time to spent evaluating an alarm
UseEvents: whether to use ZMQ events to read attributes
IgnoreExceptions: whether to consider an error in a formula as an activation or not
MaxMessagesPerAlarm: maximum number of messages to send per day

PyAlarm Configuration Parameters

In addition to PyAlarm Device properties, we also have Class Properties, that affect the apply to the whole Alarm System.

Amongst others, typical class configuration parameters would be:

SMSConfig: configuration of SMS plugin

MailConfig: configuration of Mail plugin

TelegramConfig: configuration of Telegram plugin

UserValidator: python path to the user validator class

AllowedActions: scripts that can be executed on alarm

PanicAdminUsers: users with admin access to the configuration

FromAddress: receiver address to be append to all sent messages

Phonebook: aliases to common alarm notification services or groups of receivers

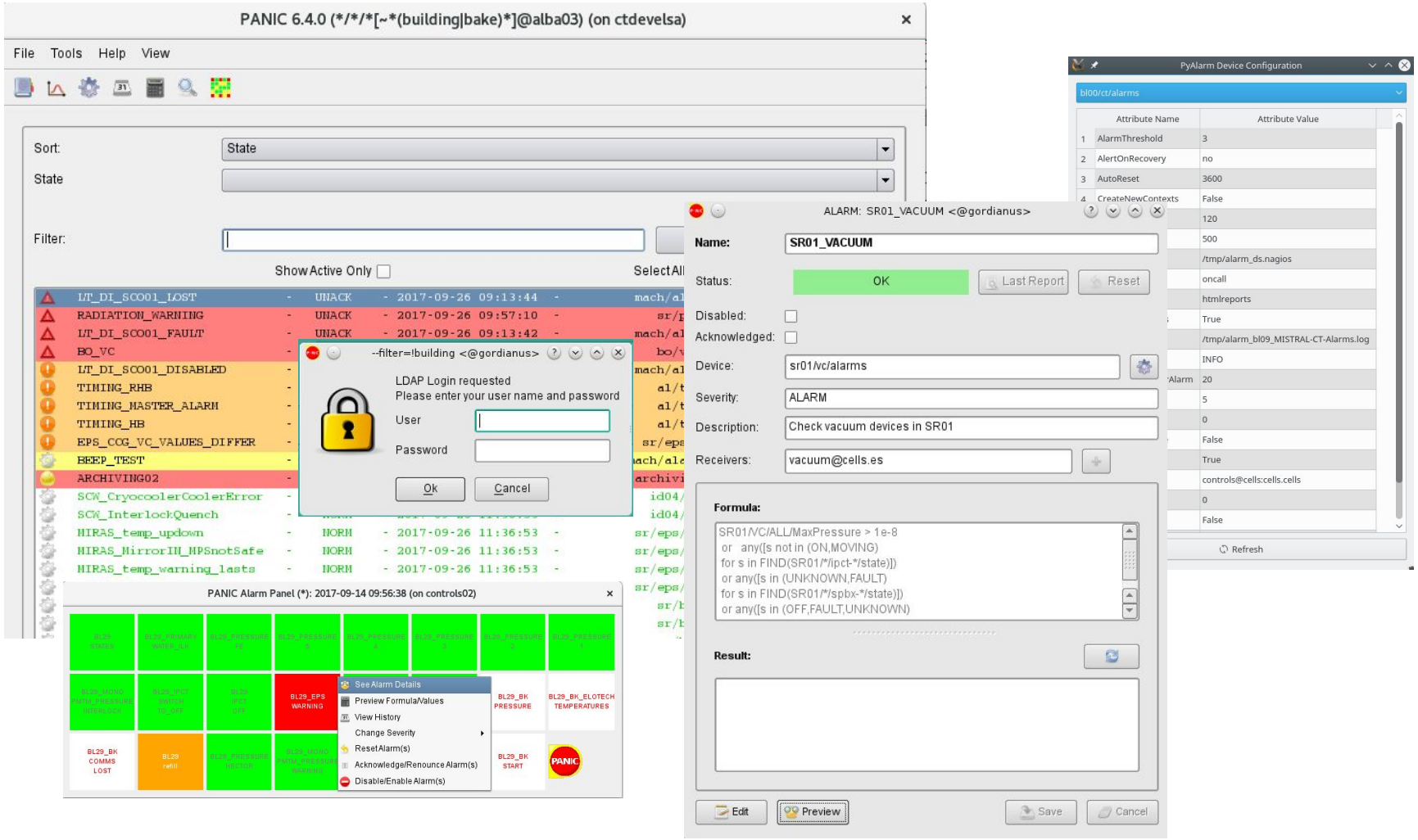
PANIC User Interface

PANIC is a distributed Alarm System. But this is transparent to the user thanks to the unified PANIC API and GUI.

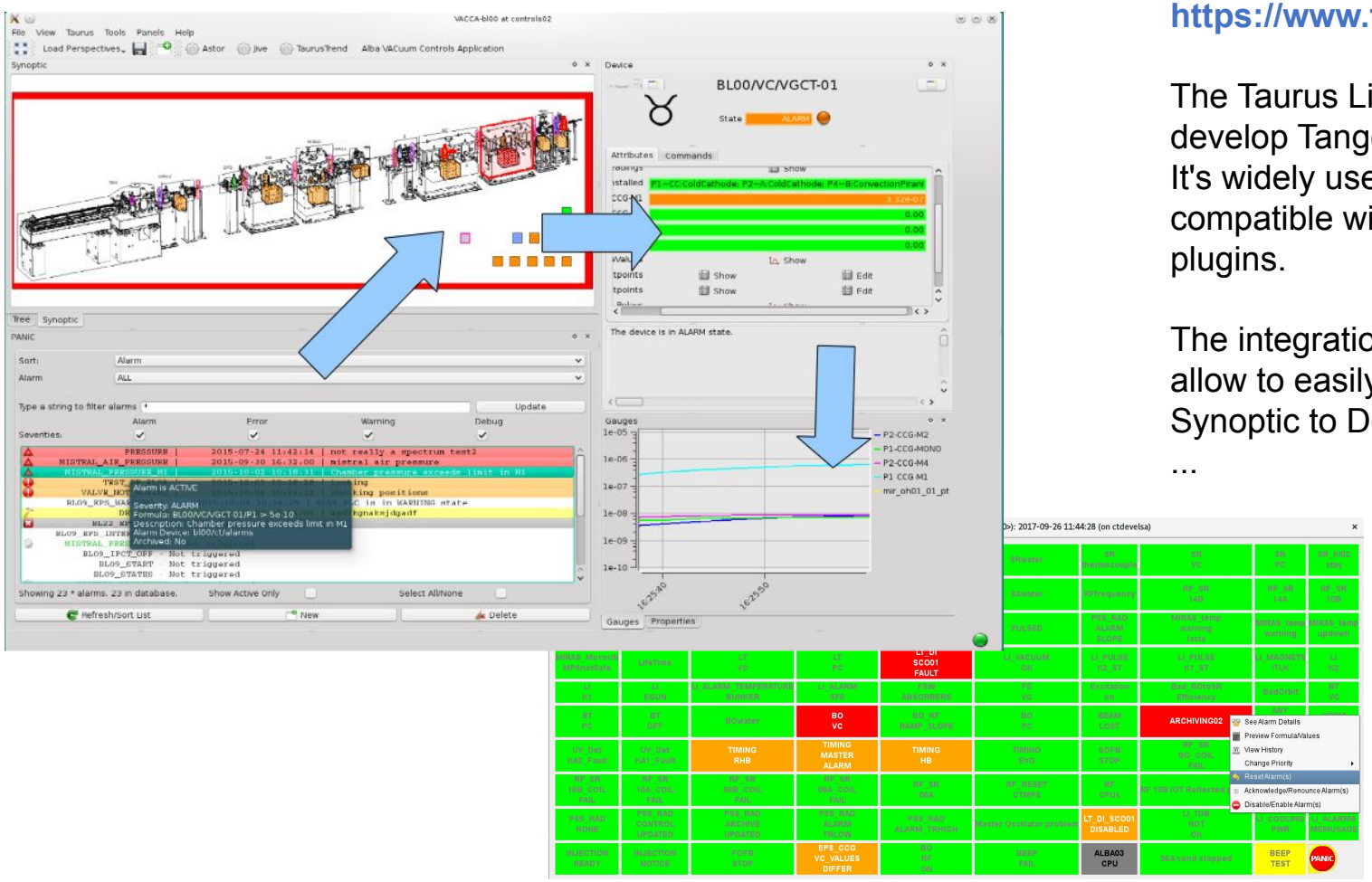
PANIC GUI allows to visualize, create, modify, disable, acknowledge, filter and configure Alarms in the whole system.

All Editing and Disabling/Acknowledging actions are restricted to admins and receivers and must be validated by authentication plugins (e.g. LDAP).

Admin users can access to each PyAlarm configuration and system-side setup.



PANIC Integration with Taurus 4 Applications



https://www.taurus-scada.org/

The Taurus Library is a python Qt toolkit to develop Tango Controls Graphical Interfaces. It's widely used on the Tango Community and compatible with other Control Systems via plugins.

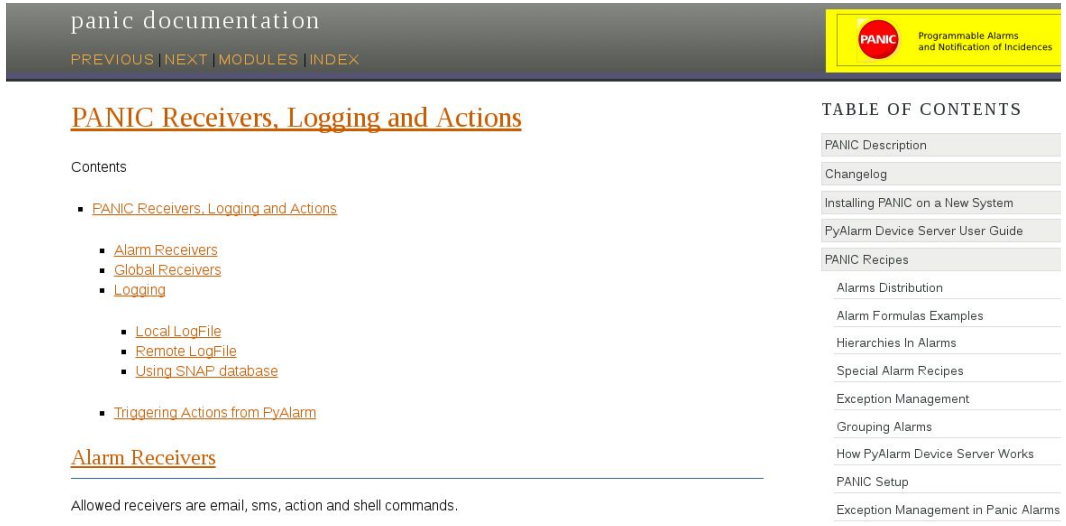
The integration of PANIC widgets with Taurus allow to easily link Alarms with Synoptics, Synoptic to Device panels, Panels with plots, ...

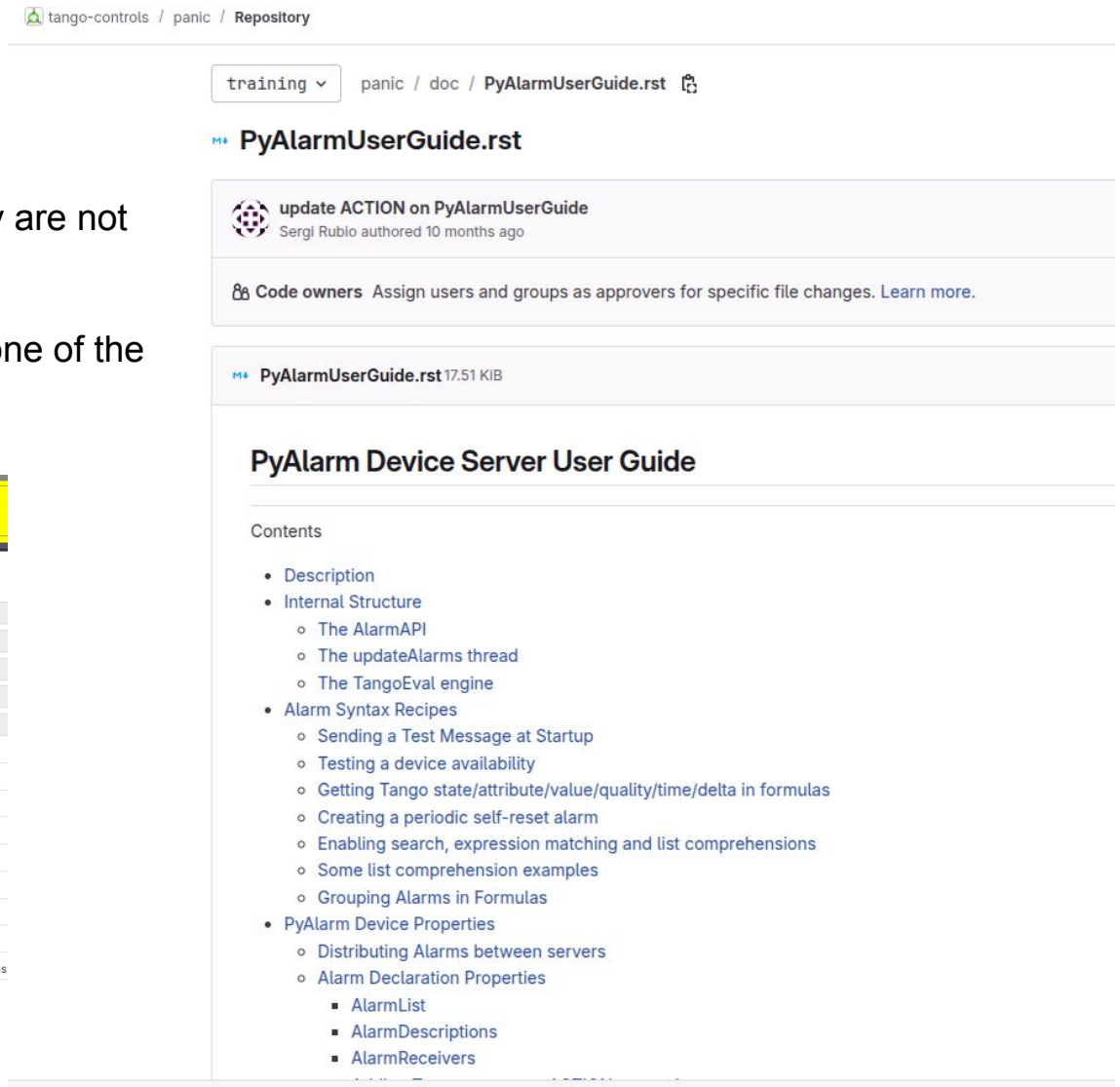
PANIC Documentation

shinx documentation (outdated) vs gitlab documentation

PANIC provides a repository of common-usage recipes, but they are not tagged and may not be easy to find.

Migration to read-the-docs as a project within tango-controls is one of the most urgent tasks.





ALBA
www.cells.es

Panic Workshop at 39th Tango Meeting

Workshop prepared with online resources made by Julen Rodriguez.

Participants were able to configure PyAlarm devices and create new alarms.

Alarms sent during the exercises were sent to a shared telegram group.

https://gitlab.com/tango-controls/panic/-/tree/training/doc/training

TANGO

EX.2 : Create an alarm with hysteresis and plot it

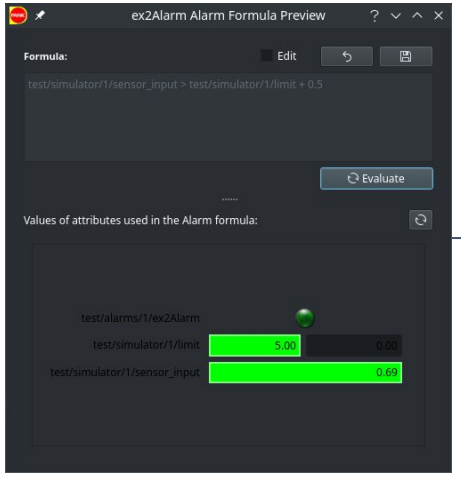
3. Plot the alarm

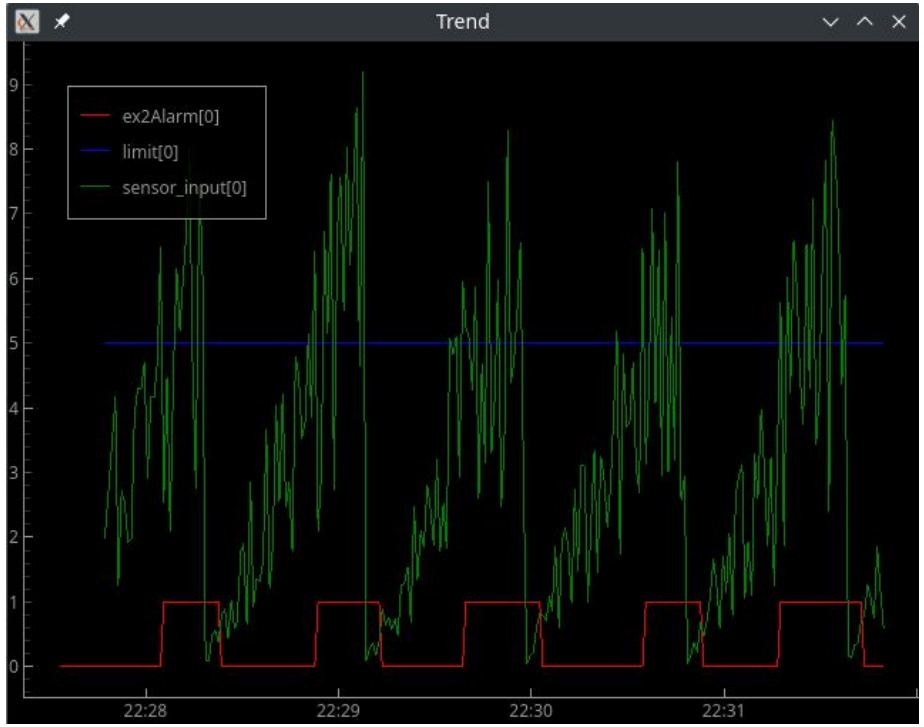
As in the previous exercise, open a taurus trend by a button in the PANIC GUi menu. Then select your alarm and open an alarm evaator (also in the PANIC GUI menu) and drag the attributes to your trend.

This alarm allows to see the alarm state depending on the input attribute value. Principal states are:

- **NORM**: The alarm condition is false. And the alarm is in normal state.
- **UNACK**: The alarm condition is true. And the alarm is in unacknowledged state.
- **ACK**: The alarm condition is true. And the alarm is in acknowledged state.
- **RTNUN**: The alarm condition is false. And the alarm is returned to normal state.

To know more about the alarm states, see the Alarm states documentation in this repository





How Panic Alarm System is used at ALBA?

PANIC Alarm System is used by all technical and scientific Divisions at ALBA.

As in the same facility (e.g. Linac) we may have different usage depending on the group (Operators, Vacuum, Safety, ...); we use the distributed nature of PANIC to separate alarms of different groups in different processes.

- An specific group (e.g. Vacuum) have permissions to modify their alarms, **but not the others.**
- Some Alarms can be disabled for an specific state during shutdown, without disabling all the system (e.g. keeping cryogenia always on).
- These permissions apply to both Alarm setup and Acknowledge, for each system a short list of "**admins**" have total permissions.

Alarms and formulas are created in different ways depending on user permissions:

- Accelerator Operators are **trained on Tango Controls** System usage, and **have permissions** to create/modify/disable alarms.
- Vacuum Engineers and Technicians are also trained, having different permission levels depending on the laboratory.
- Scientists, Infrastructure Engineers and Safety technicians specify alarm formulas, but then create **Jira tickets** to configure them.

Reaction on alarms:

- **Accelerator Operators** react on **speech and sound** notifications, plus email with alarm summary. **Control Room takes charge of it.**
- **Vacuum and Infrastructure** receive the Alarms on **SMS**, and they react on **OnCall** in case of Vacuum/Cooling/Cryogenia incident.
- **Scientists** may share a **Telegram Group** to receive alarms from their Laboratory/Beamline. Each group decides the reaction to it.

A bit of PANIC History

- Developed in 2008 as PyAlarm
- Until 2013 it was mostly used at ALBA, with PANIC version 4 it started to be widely used in the Tango Community.
- PANIC 5 was the first to be integrated into other Taurus User Interfaces.
- PANIC 6 was the first to be moved to gitlab and introduce new developments from the Tango Community. It introduced the **IEC62682** norm
- PANIC 7 had the first wave of Merge Requests. It was the first version to have compatibility with **Elettra Alarm Handler**.
- **PANIC 9** took a huge leap, MaxIV, Soleil and S2Innovation pushed for the **python3** migration of PANIC API, Device Server and GUI to python3 in this order (9.0, 9.1 and 9.2).
- **Unfortunately,** many features and MR were not merged in this transition from py2 to py3, and some features less used by all instititutes were broken due to missing dependencies (Snapshoting, AlarmViews, ...).
- In parallel, as developments in PANIC are increasingly coming from the community, our time to test and approve merge requests at ALBA is diminishing over time.

Latest features in PANIC: bugfixes

New features in **Panic 9.2**

- Device Servers (ready in 9.1) and GUI **migrated to Py3**
- New reports with much more debugging information
  - **Telegram** to replace SMS (mail-like, free, allows embedding images)
- **Ldap** integration for user authentication, user-specific logging
- Higher integration with **ProcessProfiler** (DevOps / performance monitor Tango Device server)

- **Ongoing:** new features from MaxIV/Soleil to be integrated in the Py3 code (sorry for the delay!)
- **Ongoing:** higher integration with pyhdbpp to include graphs/history on reports
  - *should hdb++ replace snap or should we go for a new snap system?*
- **Ongoing:** Event Based Alarms + High CPU Usage = filtering to prevent flooding
- **Testing:** PyAlarm disabling/enabling based on system conditions
  - Enabled=not mach/ct/alarms-servers/building_sql_error

- Next: **Multi-tango host** (implementation needs refurbish)
- Next: **AlarmViews/Groups** (to become a single thing)
- Next: **too many PyAlarm options**!! (some will be fused/renamed)

VS new features in **Panic 9.3: GUI bugfixes**

9.3.13
4078eac9 · 9.3.13 replace checkboxes by context menu · 5 days ago

```
9.3.13 replace checkboxes by context menu
```

9.3.12
c849337c · 9.3.12: fixed creation bugs, added alarm to preview · 5 days ago

```
9.3.12: fixed creation bugs, added alarm to preview
```

9.3.11
646f54e1 · Merge branch 'develop' of https://git.cells.es/ctpkg/panic into develop · 6 months ago

```
remove sys.exit() on failed send_message
```

9.3.10
dd273fc4 · fix edit of alarm when acknowledge/disable is not readable · 8 months ago

```
9.3.10 fix edit of alarm when acknowledge/disable is not readable
```

9.3.9
568f6218 · remove taurus dependency from pyalarm · 10 months ago

```
remove taurus dependency from pyalarm
```

9.3.8
19ed8877 · 9.3.8: fix bugs on checking exceptions (e.message) · 10 months ago

```
9.3.8: fix bugs on checking exceptions (e.message)
```

9.3.7
6dc4111a · 9.3.7 fix gui entry point · 11 months ago

```
9.3.7 fix gui entry point
```

PANIC v 10: Pending Work

PANIC v10 had clear objectives defined:

- Integrate ALARM Events into the subscription.
- Use events by default and reduce client-polling.
- Simplify the development of notification services.
- Provide regular conda packages via conda-forge.

But there are many pending thins in PANIC 9!

- Better integration with Taurus
- **Pending issues & Merge Requests on gitlab**
- Documentation rebuild
  - Current documentation fragmented
  - List of How To's to be organized
  - Pending readthedocs
- Need for RFCs!! (both for testing and future development)
- Conda packaging
- AlarmViews and Groups (rewrite or deprecate)
- Logging via snapshots

ALBA
www.cells.es

PANIC Open Issues & Merge Requests

tango-controls / panic / **Issues**

Open 42 | Closed 17 | All 59

Search or filter results...

- **PanicGUI: TypeError in alarm sorting**
  #70 · created 2 months ago by Michal Piekarski
- **script arguments ignored in actions**
  #69 · created 5 months ago by Raphaël GIRARDOT
- **New alarm: double clic clears both "Tag", "Description" and "Annunciators"**
  #68 · created 5 months ago by Raphaël GIRARDOT
- **No email for RECOVERED event**
  #67 · created 5 months ago by Raphaël GIRARDOT
- **Space after comma became mandatory in panic 9.3.9**
  #66 · created 5 months ago by Raphaël GIRARDOT
- **Renaming proposal**
  #65 · created 5 months ago by Raphaël GIRARDOT
- **Tango V10 new Features**
  #64 · created 8 months ago by Vincent Hardion
- **PANIC with restricted access to properties**
  #63 · created 11 months ago by Raphaël GIRARDOT
- **panic gui python3: unable to edit alarm of a pyalarm that is not running**
  #61 · created 1 year ago by S Rubio ALBA
- **panic gui 9.3.3: alarm editor does not refresh state after active/reset/active cycle**
  #60 · created 1 year ago by S Rubio ALBA
- **PyAlarm evolution: possibility not to enable disabled alarms at restart**
  #58 · created 1 year ago by Raphaël GIRARDOT
- **PANIC does not update alarm list at first alarm creation**
  #57 · created 1 year ago by Raphaël GIRARDOT

tango-controls / panic / **Merge requests**

Open 9 | Merged 6 | Closed 5 | All 20

Search or filter results...

- **Install pip and screen in conda enviroment for trainng**
  !51 · created 1 day ago by Julen Rodríguez Millán `training`
- **accept script arguments (solves issue #69)**
  !50 · created 5 months ago by Raphaël GIRARDOT
- **it is now possible to disable active alarms (solves issue #44)**
  !49 · created 5 months ago by Raphaël GIRARDOT
- **Replace script with entry-point**
  !48 · created 11 months ago by Benjamin Bertrand
- **Introduced hard_filtering**
  !43 · created 3 years ago by S Rubio ALBA
- **Added PyQt5 support**
  !40 · created 5 years ago by S Rubio ALBA `develop`
- **Add Start/Stop commands.**
  !35 · created 5 years ago by S Rubio ALBA `develop`
- **Add smsmodem support.**
  !34 · created 5 years ago by S Rubio ALBA `develop`
- **Add logstash reports.**
  !33 · created 5 years ago by S Rubio ALBA `develop`

TANGO

PANIC at the crossroad: v10 or community MR's?

PANIC and TANGO are technologies in continuous evolution, as well as our scientific facilities and needs.

As PANIC is widely used within the community ... almost every decision I'll take on further development will affect anyone; some features will be implemented, others will not, and others will be deprecated. This kind of decisions must be taken along with the users and developers using PANIC in other institutes.

My proposal:

- To create a **mattermost channel** for panic, so everyone using panic can share and discuss their problems and interests.
- Organize **a PANIC Community Meeting** (on remote) with users and developers interested on contributing.
- Define **a Roadmap from PANIC 9.3 to v10**, listening to the priorities of the institutes affected and also distributing those tasks that can be shared.
- Decide how we are going to use **ALARM** events!
- **Write RFCs and new tests!!**

Need for RFCs! : PANIC underwent big migrations of code, from python2 to python3 and then from Qt4 to Qt5, while adding new features and integrating them into the code but not with CI/CD tests; mostly because of the restrictions of testing time-dependent tasks and check notifications on a CI/CD routine. We need RFC specification for expected behaviors in PANIC in order to validate MR and prepare tests for each new functionality.

Future of PANIC : developments on sight

PANIC and TANGO are technologies in continuous evolution, as well as our scientific facilities and needs.

In the last years, and after collecting the experience of our users at ALBA and all the other institutes currently using PANIC, we have mad several proposals of improvement for the PANIC System:

- Improving system responsiveness using ALARM Event from TANGO, executing alarms on hardware/system request, reduce polling.
- Integrate usability / filtering / logging capabilities implemented by Max IV, Soleil, Solaris and S2Innovation (web tools).
- New logging system and better integration with TANGO Archiving (HDB++).
- Alarm Views, provide dashboards per-user to improve filtering on big control systems and prevent operator overload.
- Process and host profiling, extend PANIC capabilities to provide alarms not only on Control System variables but on IT infrastructure.
- Better configuration, allow to have shared setups between control systems, making easier the maintenance.
- Alarm templates, to allow users interacting with multiple systems (e.g. Vacuum) to apply similar alarm formulas on each of the laboratories.
- PyAlarm options rationalization, simplify the parameters to make the configuration more transparent to operators.
- Operation "Contexts" : enable / disable alarms depending on the status **(operation / test / shutdown).**
- **Recover some functionalities lost on the py3 transition, deprecation of functionalities not used.**

PANIC : Online Resources

Git Repository, tickets and merge requests :

https://gitlab.com/tango-controls/panic

https://gitlab.com/tango-controls/panic/-/blob/training/doc/PyAlarmUserGuide.rst

https://www.tango-controls.org/

Online Documentation and Recipes :

https://tango-controls.readthedocs.io/projects/panic

https://gitlab.com/tango-controls/panic/-/tree/documentation/doc/recipes

Panic is is already available in Conda environments via **pip install**; and in the process of being added to conda-forge:

https://pypi.org/project/panic

https://pypi.org/project/pytango/

https://pypi.org/project/fandango

https://pypi.org/project/taurus

IC@MS: commercial PANIC web-client by S2Innovation: https://s2innovation.com/

PANIC is a core project within the TANGO Community. An online training course is being prepared using docker for easy deployment. Will be published soon in the tango-controls youtube channel: *https://www.youtube.com/@tango-controls*

Time for Questions, Don't Panic and Join the effort!

www.cells.es

srubio@cells.es

Thanks to INAF for giving us the chance to prepare a PANIC Workshop!

And thanks to all PANIC Contributors from ALBA, S2Innovation, MaxIV, Soleil, SKA, ESRF, Solaris, Elettra