Introductory workshop to the PANIC Alarm System

Sergi Rubio Manrique - Julen Rodriguez Millan - ALBA Synchrotron

ALBA
www.cells.es

Index

Prepare your setup for the workshop:

https://gitlab.com/tango-controls/panic/-/blob/training/doc/training/setup.md

A setup using conda (microforge), docker and MailHog have been prepared to make more reproducible the exercises from this training.

The setup should work in Linux, Mac and Windows, although the last parts of the exercises (speech) may not fully work

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).

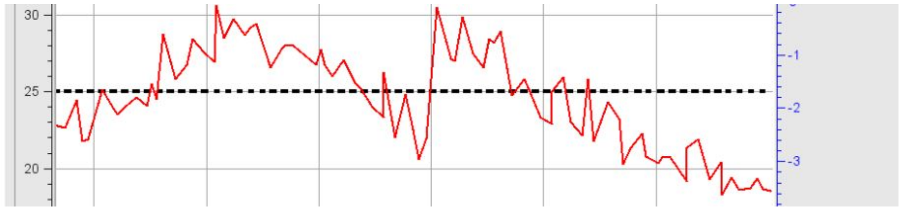
www.cells.es

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 :

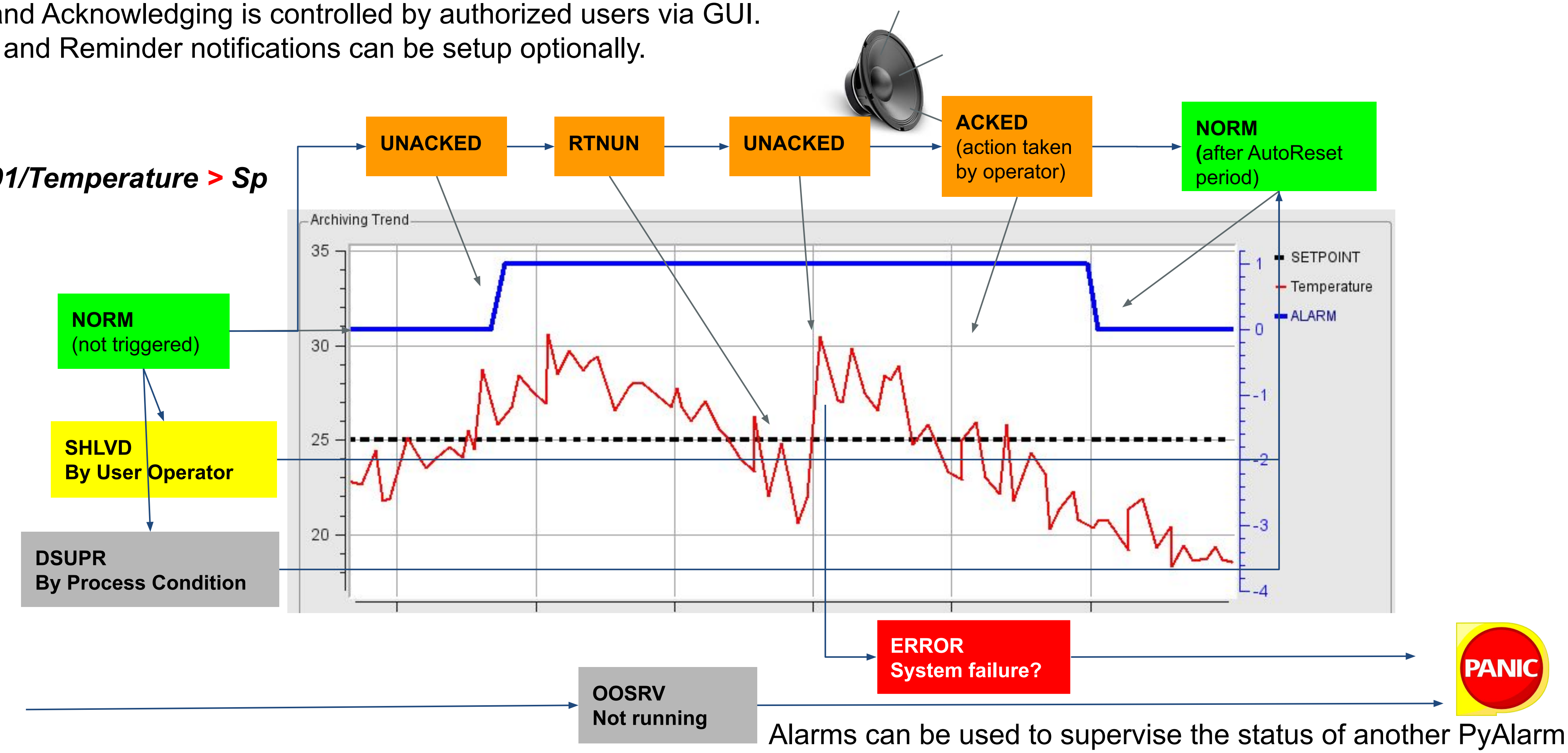
PyAlarm has been later extended by PANIC API and PANIC GUI to provide extended Alarm capabilities for all ALBA accelerators and laboratories, providing an scalable system with multiple notification services, user authentication, logging, ...

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,** ...

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.

formula = my/plc/01/Temperature > Sp



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

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.

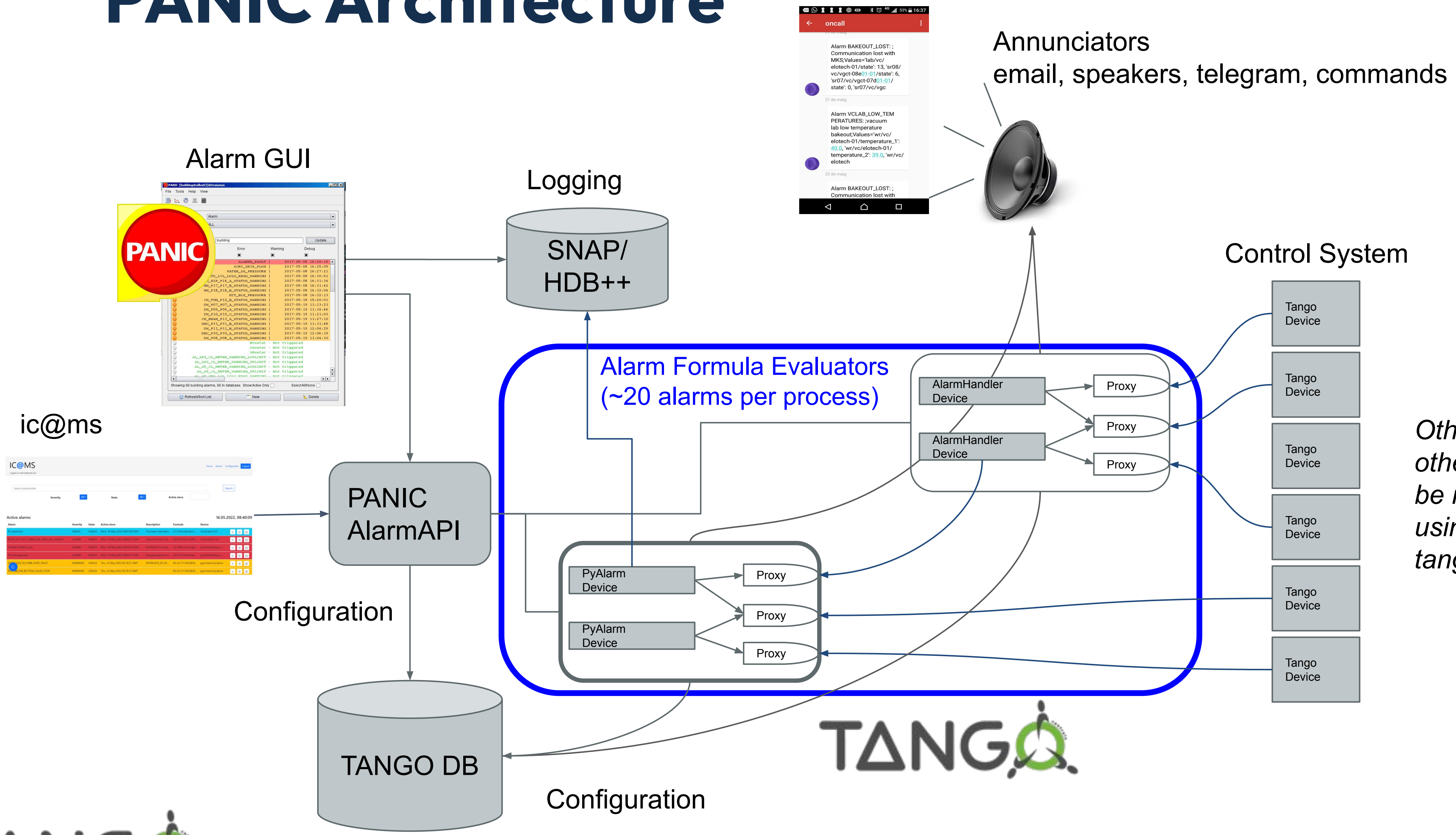
Logging Database :

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.

PANIC Architecture



oncall

Alarm BAKEOUT_LOST: ; Communication lost with MKS;Values='lab/vc/elotech-01/state': 13, 'sr08/vc/vgct-08e01-01/state': 6, 'sr07/vc/vgct-07d01-01/state': 0, 'sr07/vc/vgc

21 de maig

Alarm VCLAB_LOW_TEMPERATURES: ;vacuum lab low temperature bakeout;Values='wr/vc/elotech-01/temperature_1': 40.0, 'wr/vc/elotech-01/temperature_2': 39.0, 'wr/vc/elotech

22 de maig

Alarm BAKEOUT_LOST: ; Communication lost with

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

ALBA
www.cells.es

Guide to Exercises

https://gitlab.com/tango-controls/panic/-/blob/training/doc/training/exercises_guide.md

TANGO

ALBA
www.cells.es

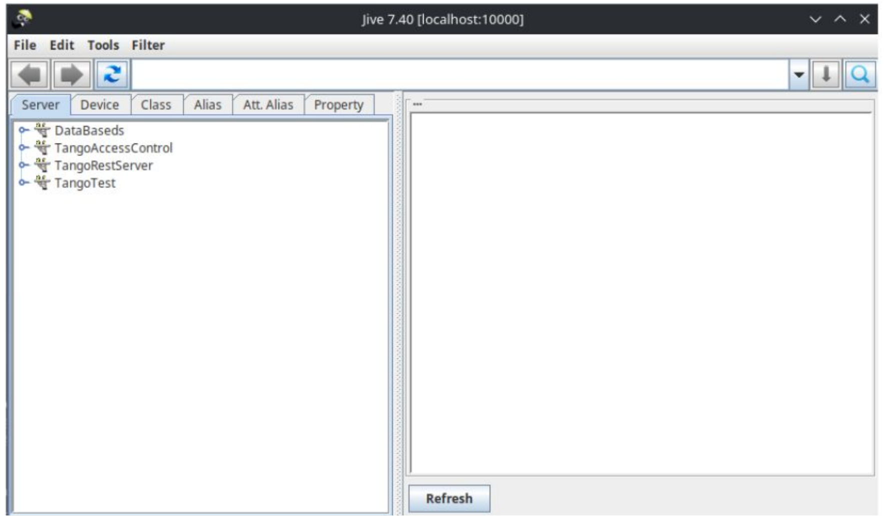
EX.1 : Create an alarm from Tango Test device.

1. Open Jive, the Tango device browser.

$ conda activate panic-workshop

$ export TANGO_HOST=localhost:10000

$ jive &

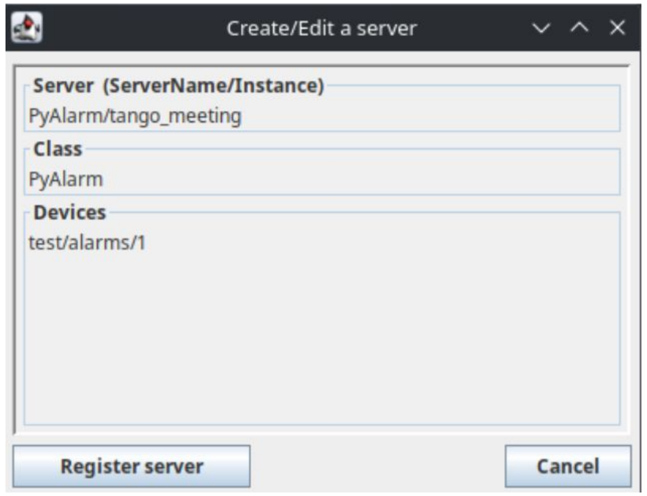


2. Create a PyAlarm-based server

Use Jive to create a new server with a PyAlarm device.

Edit -> Create Server

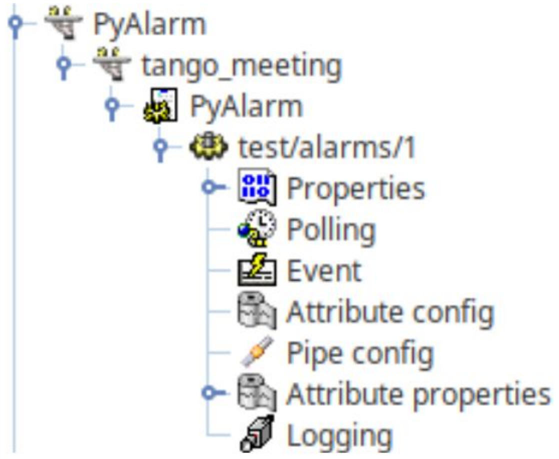
- Instance: PyAlarm/tango_meeting
- Class: PyAlarm
- Devices: test/alarms/1


Create/Edit a server

Server (ServerName/Instance)
PyAlarm/tango_meeting

Class
PyAlarm

Devices
test/alarms/1



TANGO

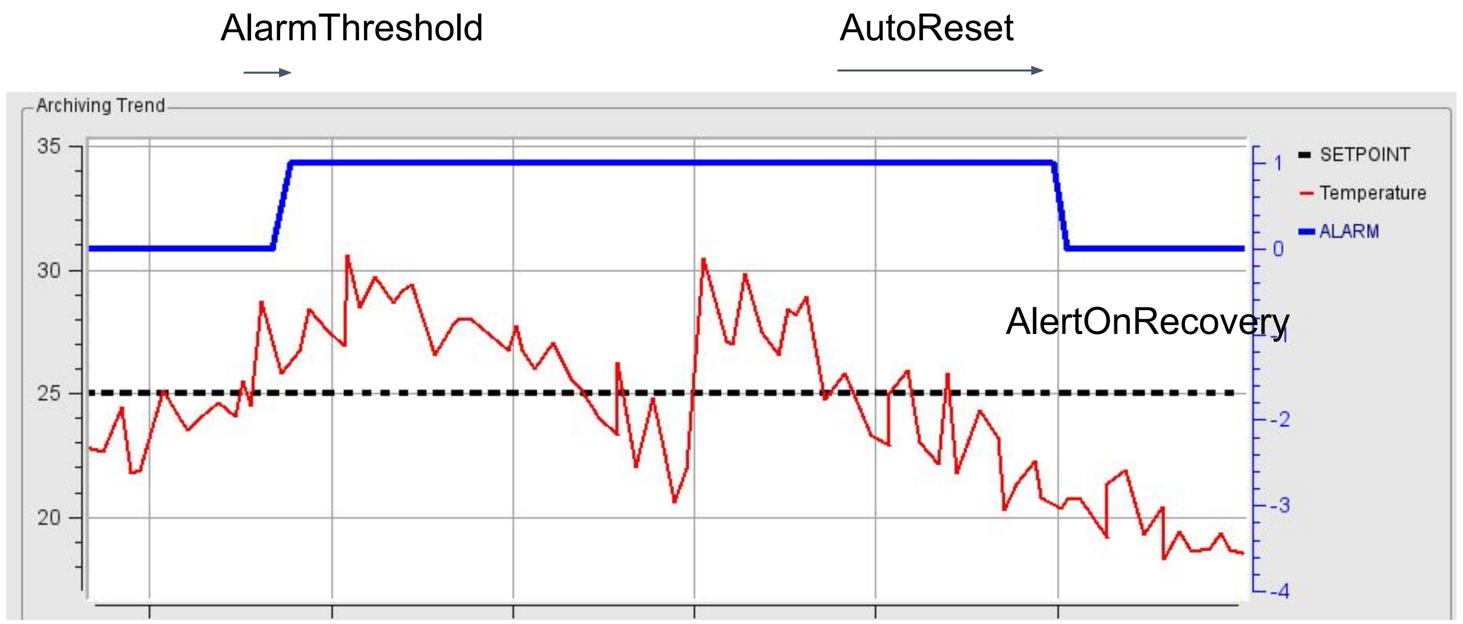
ALBA www.cells.es

EX.1 : Create an alarm from Tango Test device.

3. Set some properties to show alarm cycle

Edit the device properties to enhance speed and auto-reset behavior.

- **AlarmThreshold**: 1
- **AlertOnRecovery**: True
- **AutoReset**: 5
- **Enabled**: True
- **MailMethod**: smtp:localhost:1025
- **PollingPeriod**: 1



Full description of properties:

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

ALBA
www.cells.es

EX.1 : Create an alarm from Tango Test device.

4. Export the server

Run the following commands in your terminal (we use linux screen to hide traces and recover them later):

```
conda activate panic-workshop
export TANGO_HOST=localhost:10000
screen -dm -S pyalarm PyAlarm tango_meeting -v4
```

or, alternatively:

```
python3 -m panic.ds.PyAlarm tango_meeting -v2
```

This will run a screen with the server output.

For the training we only need to know the following commands:

```
screen -ls
screen -r <session_id>
```

Where `<session_id>` is the id of the screen session you want to attach.

From the screen session you can stop the server with `Ctrl+C` and detach it with `Ctrl+A` then `D`.

For more details of use screen, see the screen documentation.

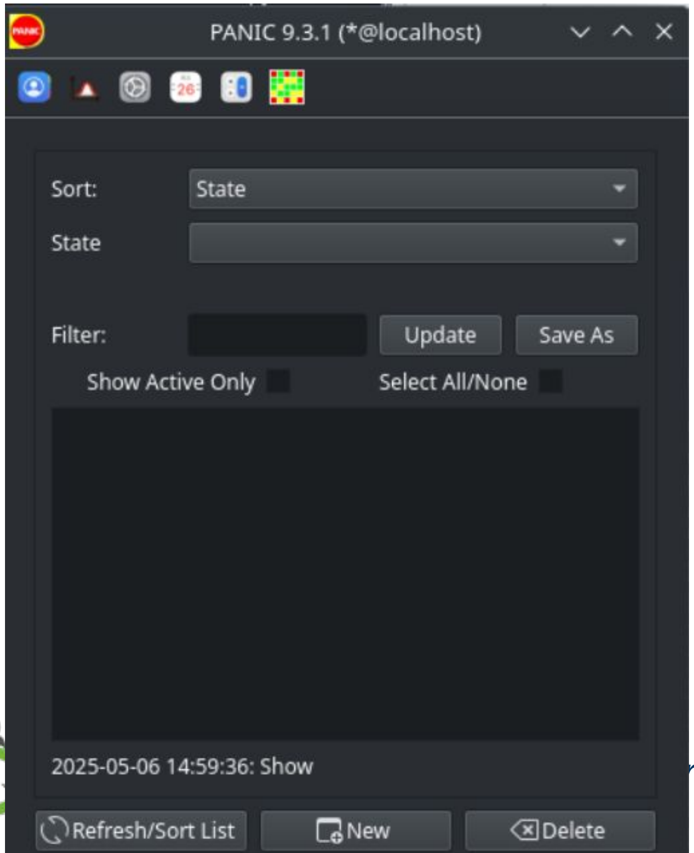
TANGO

EX.1 : Create an alarm from Tango Test device.

5. Open PANIC

Launch the PANIC GUI to manage alarms.

type "panic" or "python3 -m panic.gui.gui"



PANIC 9.3.1 (*@localhost)

Sort: State

State

Filter:

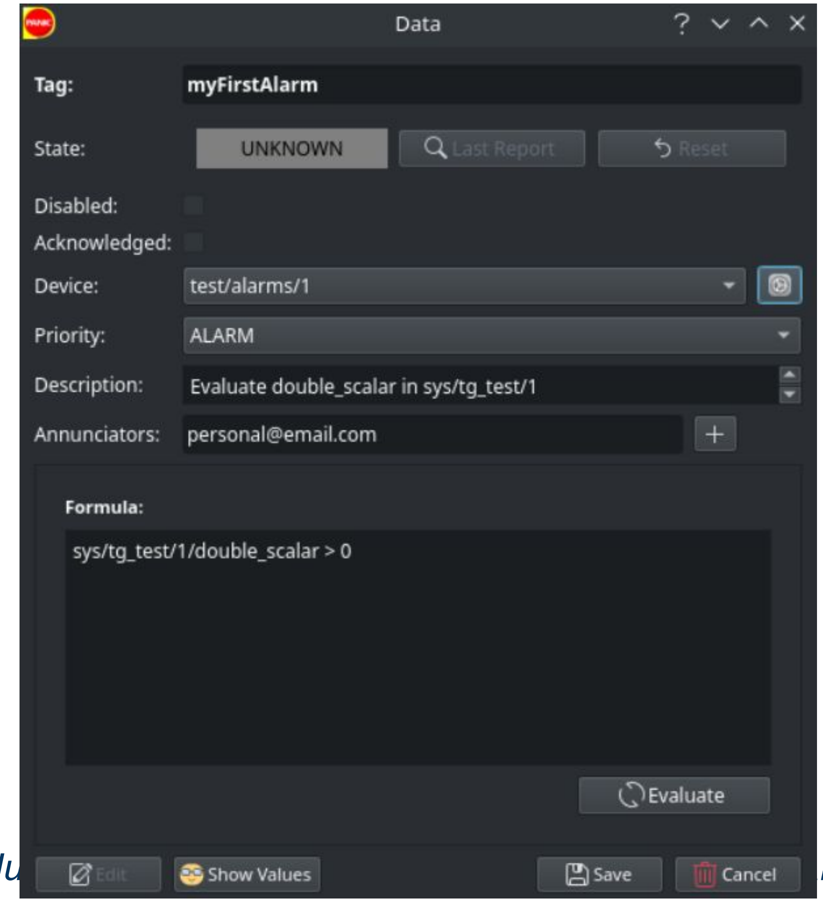
Show Active Only

Select All/None

2025-05-06 14:59:36: Show

6. Create an alarm

Create a new alarm using the GUI.



Data

Tag: myFirstAlarm

State: UNKNOWN

Disabled:

Acknowledged:

Device: test/alarms/1

Priority: ALARM

Description: Evaluate double_scalar in sys/tg_test/1

Annunciators: personal@email.com

Formula:

```
sys/tg_test/1/double_scalar > 0
```

EX.1 : Create an alarm from Tango Test device.

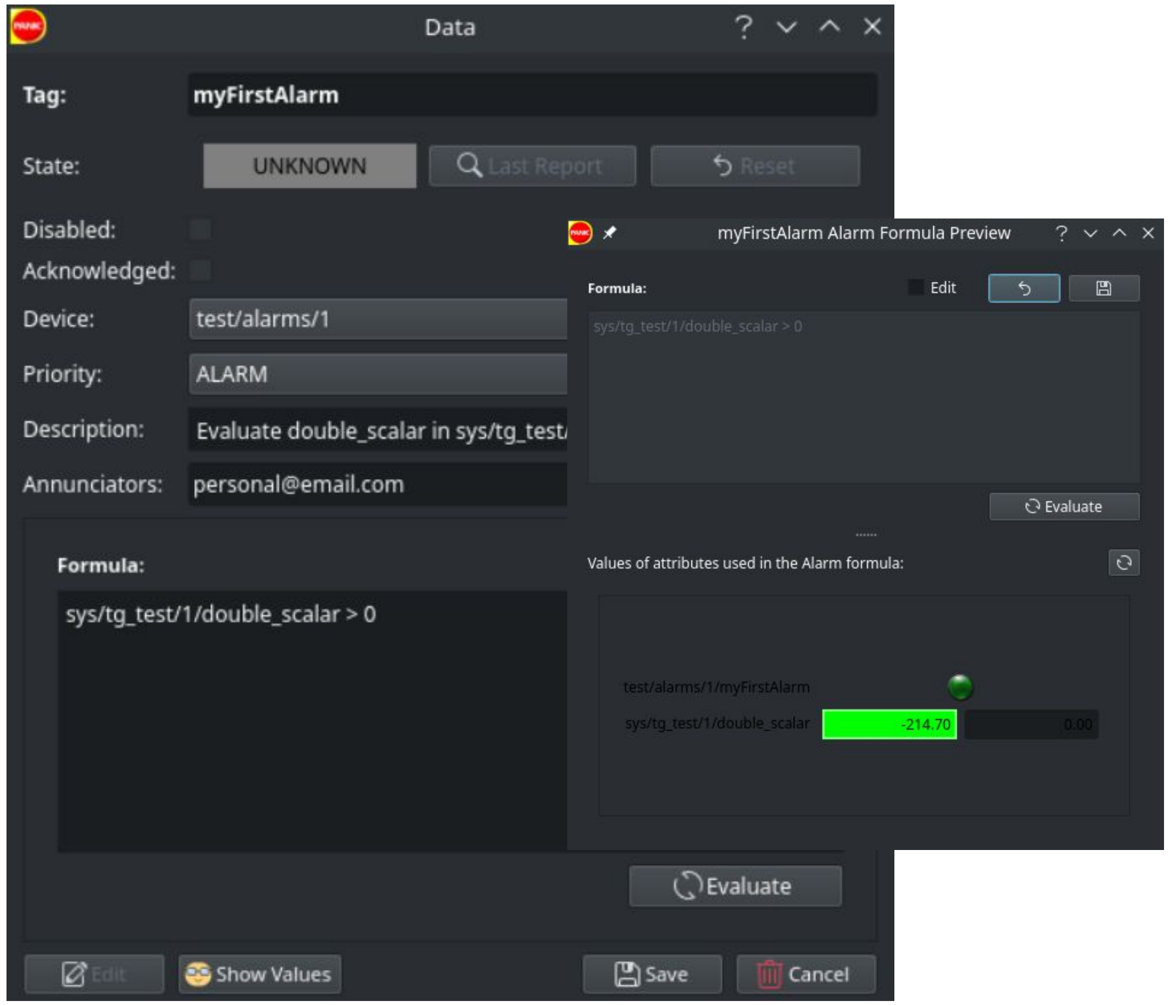
6. Create an alarm

Create a new alarm using the GUI.

myFirstAlarm Details:

- **Tag**: `myFirstAlarm`
- **Device**: `test/alarms/1`
- **Priority**: `ALARM`
- **Description**: `Evaluate double_scalar in sys/tg_test/1`
- **Annunciators**: `personalmail@mail.com`
- **Formula**:

  sys/tg_test/1/double_scalar > 0



Tag: myFirstAlarm
State: UNKNOWN
Disabled:
Acknowledged:
Device: test/alarms/1
Priority: ALARM
Description: Evaluate double_scalar in sys/tg_test/
Annunciators: personal@email.com
Formula:
sys/tg_test/1/double_scalar > 0

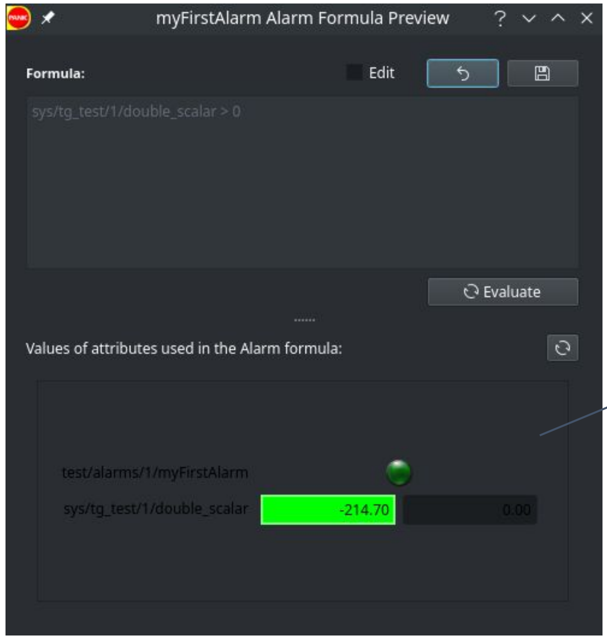
Formula:
sys/tg_test/1/double_scalar > 0
Values of attributes used in the Alarm formula:
test/alarms/1/myFirstAlarm
sys/tg_test/1/double_scalar -214.70 0.00

EX.1 : Create an alarm from Tango Test device.

⚠️ The Taurus Trend View is a powerful tool for visualizing the evolution of attributes over time.

Open a taurus trend by a button in the PANIC GUi menu. Then select your alarm **and open the "Show Values" preview** panel (also in the PANIC GUI menu) and drag the attributes from that panel to your trend.

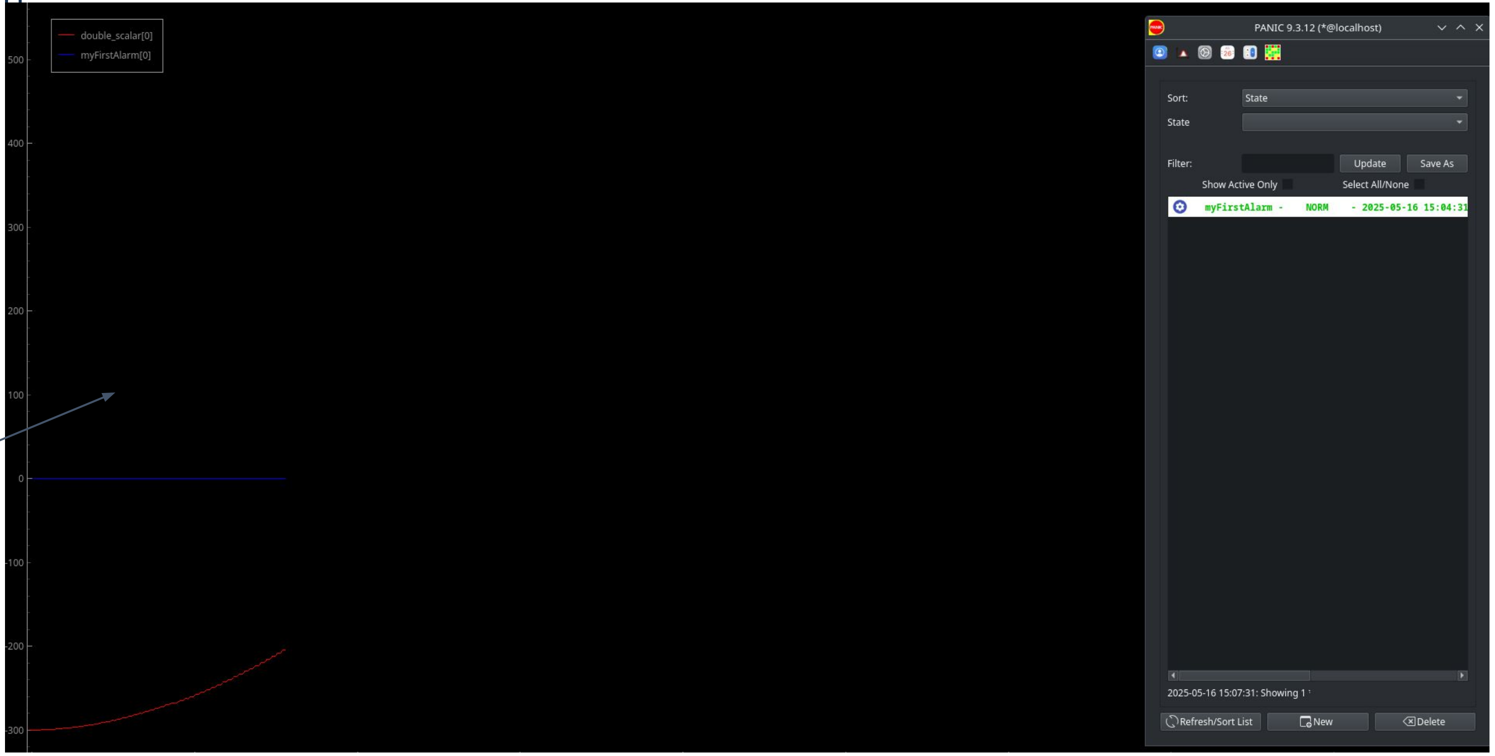
See the alarm activity. The alarm will be triggered when the attribute is positive. The alarm is active in the positive wave part of the sinusoidal function.


myFirstAlarm Alarm Formula Preview
Formula:
Edit
Evaluate
Values of attributes used in the Alarm formula:

```
sys/tg_test/1/double_scalar > 0
```

test/alarms/1/myFirstAlarm

sys/tg_test/1/double_scalar -214.70 0.00



```
myFirstAlarm -    NORM    - 2025-05-16 15:04:31
```

2025-05-16 15:07:31: Showing 1

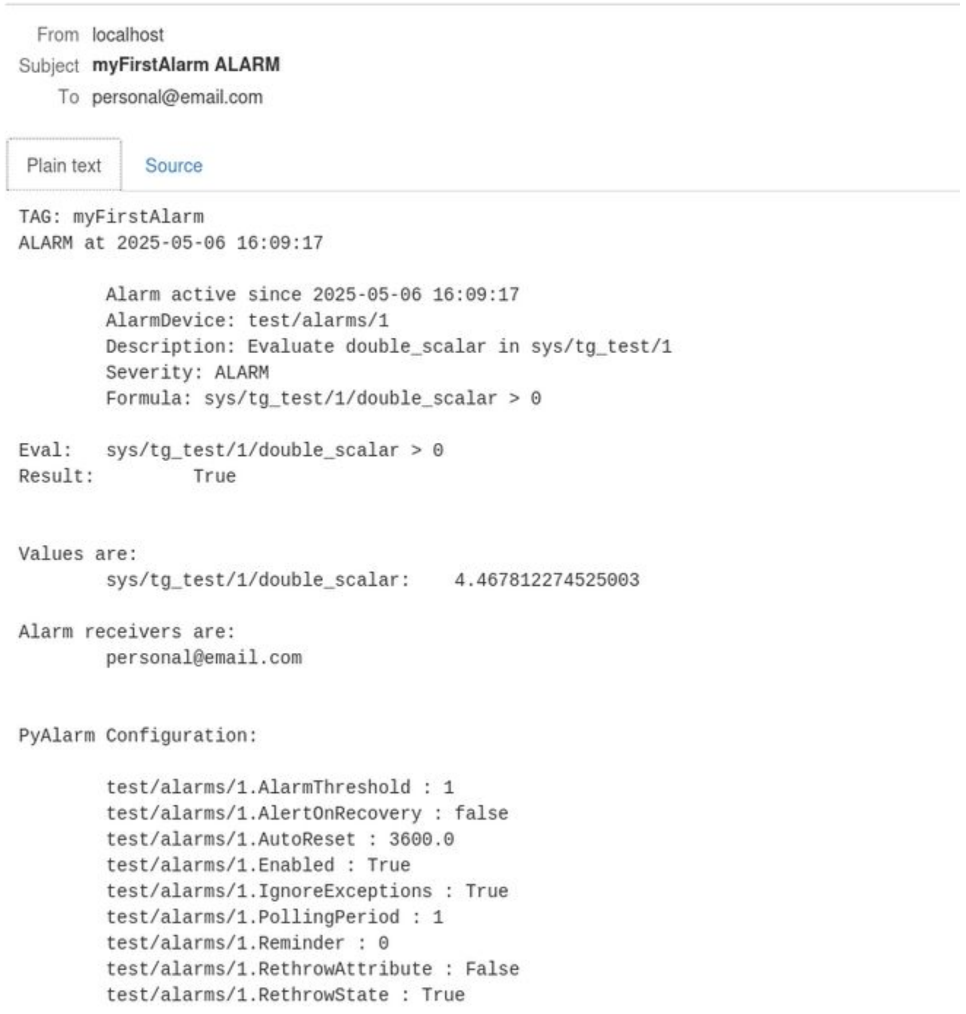
Sergi Rubio Manrique - Julen Rodriguez Millan - 39th Tango Meeting - INAF - 21h May 2025

EX.1 : Create an alarm from Tango Test device.

8. Check local email simulation

View the simulated email message triggered by the alarm.

https://localhost:8025



```
From     localhost
Subject  myFirstAlarm ALARM
To       personal@email.com
```

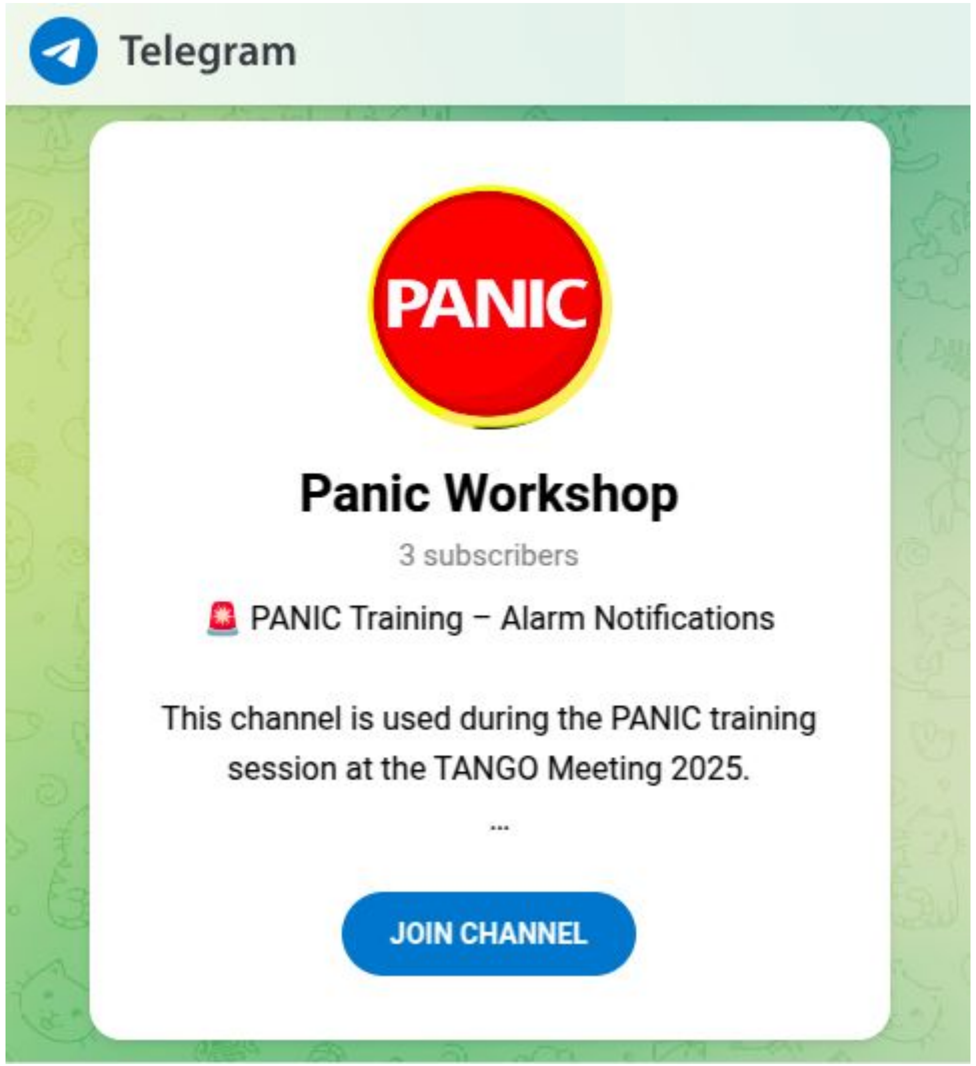
```
TAG: myFirstAlarm
ALARM at 2025-05-06 16:09:17

        Alarm active since 2025-05-06 16:09:17
        AlarmDevice: test/alarms/1
        Description: Evaluate double_scalar in sys/tg_test/1
        Severity: ALARM
        Formula: sys/tg_test/1/double_scalar > 0

Eval:   sys/tg_test/1/double_scalar > 0
Result:         True


Values are:
        sys/tg_test/1/double_scalar:    4.467812274525003

Alarm receivers are:
        personal@email.com


PyAlarm Configuration:

        test/alarms/1.AlarmThreshold : 1
        test/alarms/1.AlertOnRecovery : false
        test/alarms/1.AutoReset : 3600.0
        test/alarms/1.Enabled : True
        test/alarms/1.IgnoreExceptions : True
        test/alarms/1.PollingPeriod : 1
        test/alarms/1.Reminder : 0
        test/alarms/1.RethrowAttribute : False
        test/alarms/1.RethrowState : True
```

Telegram Group for the Workshop:

Alarms sent during the exercises will be sent to a telegram group, you may subscribe if you were not able to setup a local mail server.

https://t.me/+xHAXagP2vdZINzg8
TG:-1002594519846

execute: python3 scripts/setup_telegram.py
then, add %WORKSHOP as receiver of your alarms



Panic Workshop

3 subscribers

🚨 PANIC Training – Alarm Notifications

This channel is used during the PANIC training session at the TANGO Meeting 2025.

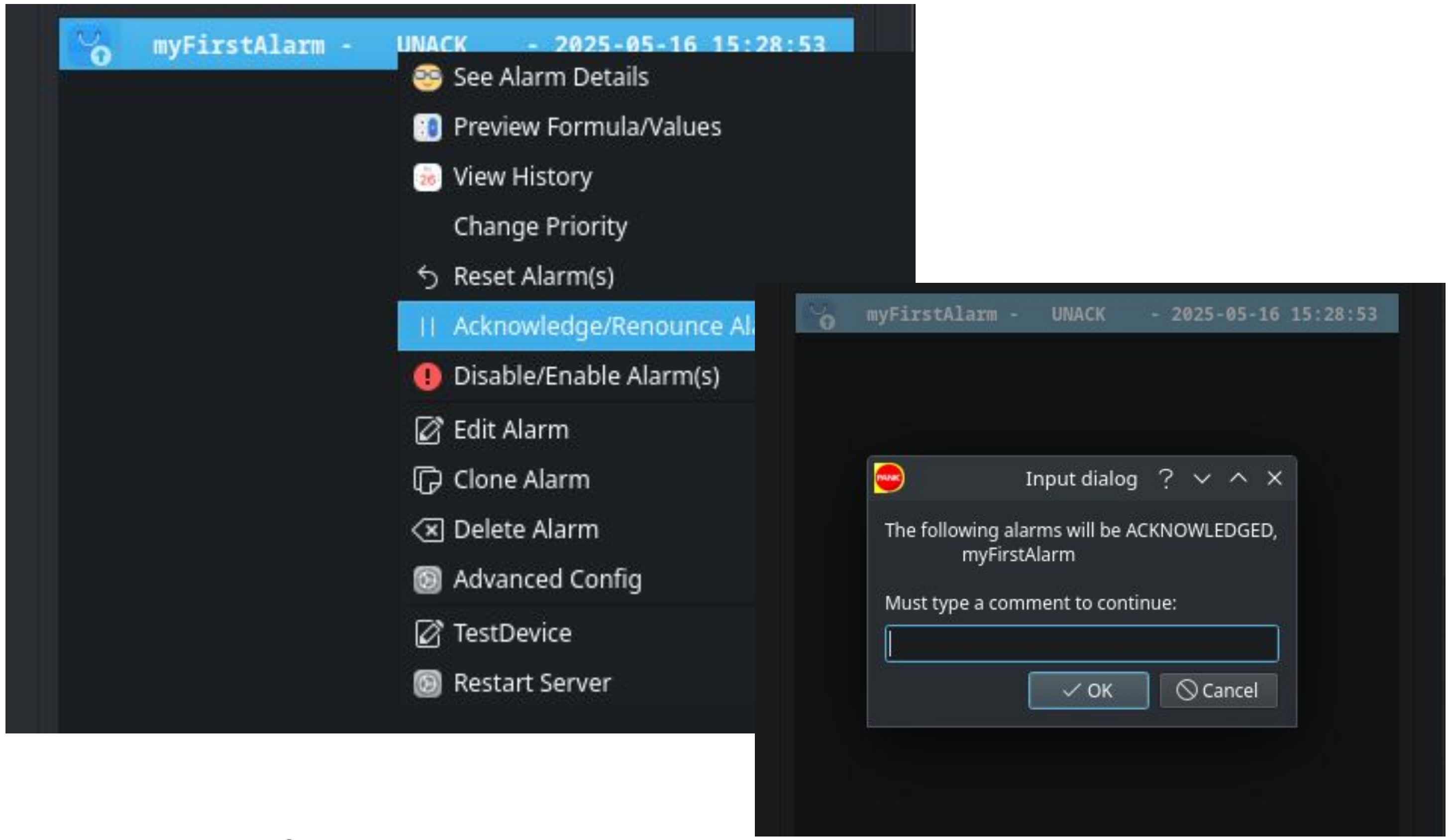
...

ALBA
www.cells.es

EX.1 : Create an alarm from Tango Test device.

9. Acknowledge the alarm

Right-click on the alarm in the Panic GUI and select **"Acknowledge"**.



```
myFirstAlarm -    UNACK    - 2025-05-16 15:28:53
```

```
myFirstAlarm -    UNACK    - 2025-05-16 15:28:53
```

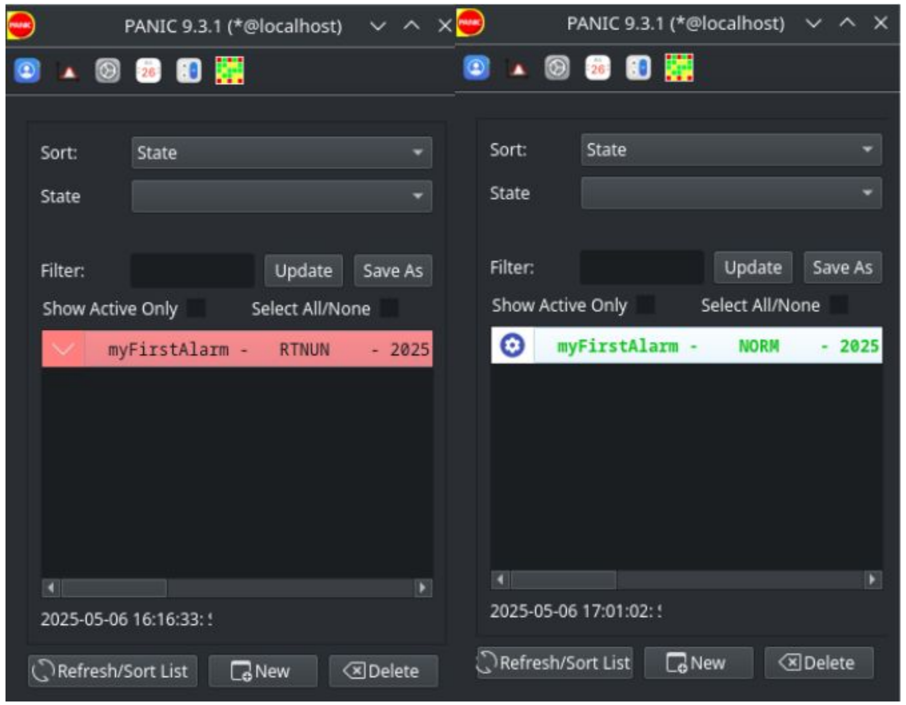
Input dialog

The following alarms will be ACKNOWLEDGED,
myFirstAlarm

Must type a comment to continue:

10. Alarm returns to normal state

Once the condition is false, the alarm resets to normal.



PANIC 9.3.1 (*@localhost)

Sort: State
State:
Filter:
Show Active Only
Select All/None

```
myFirstAlarm -    RTNUN    - 2025
```

2025-05-06 16:16:33:

PANIC 9.3.1 (*@localhost)

Sort: State
State:
Filter:
Show Active Only
Select All/None

```
myFirstAlarm -    NORM    - 2025
```

2025-05-06 17:01:02:

TANGO

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

1. Create DynamicDs by script

To create alarms in this exercise and the next ones, we will use the `DynamicDs` device. It allows to create dynamic attributes and to set their values. Is a good way to simulate the behavior of a Tango device to test alarms.

```
export TANGO_HOST=localhost:10000
```

```
python3 scripts/launch_workshop_simulator.py
```

As the previous exercise, the script will run the server in a screen session. You can use the same commands to attach and detach the screen session.

For a full description of DynamicDS and its Dynamic Attributes behaviour see:

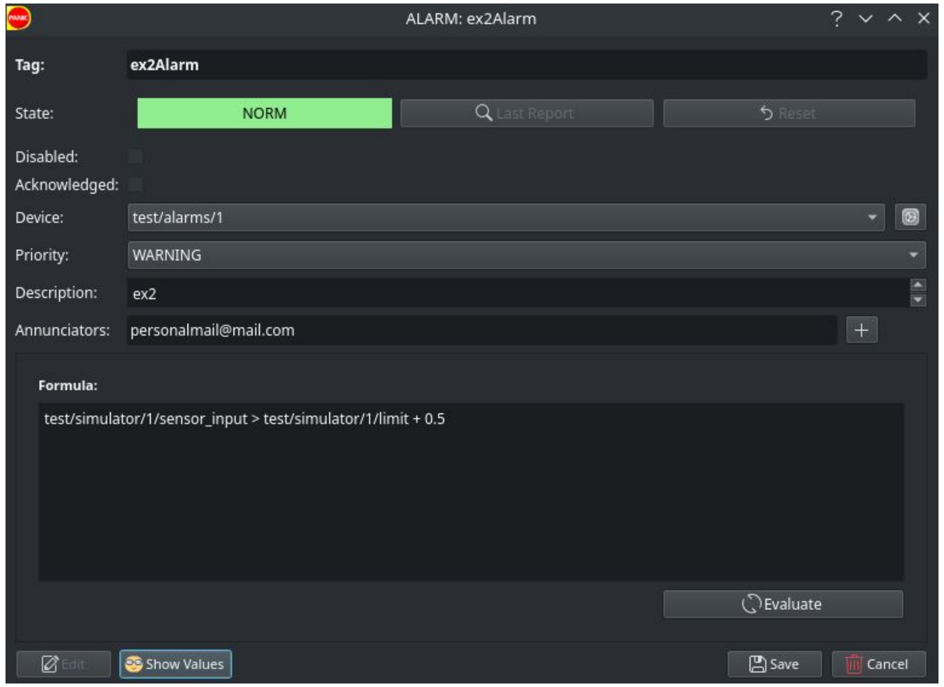
https://gitlab.com/tango-controls/fandango/-/blob/master/doc/recipes/DynamicDS_and_Simulators.rst

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

2. Create a new alarm

In the PANIC GUI, create a new alarm using the **"Create Alarm"** button

This alarm use an input attribute and a limit attribute. The alarm will be triggered when the input attribute value is greater than the limit attribute value but with a hysteresis of 0.5. This means that the alarm will only be triggered when the input attribute value is greater than the limit attribute value plus 0.5.


ALARM: ex2Alarm

Tag:	ex2Alarm
State:	NORM
Disabled:	
Acknowledged:	
Device:	test/alarms/1
Priority:	WARNING
Description:	ex2
Annunciators:	personalmail@mail.com

Formula:

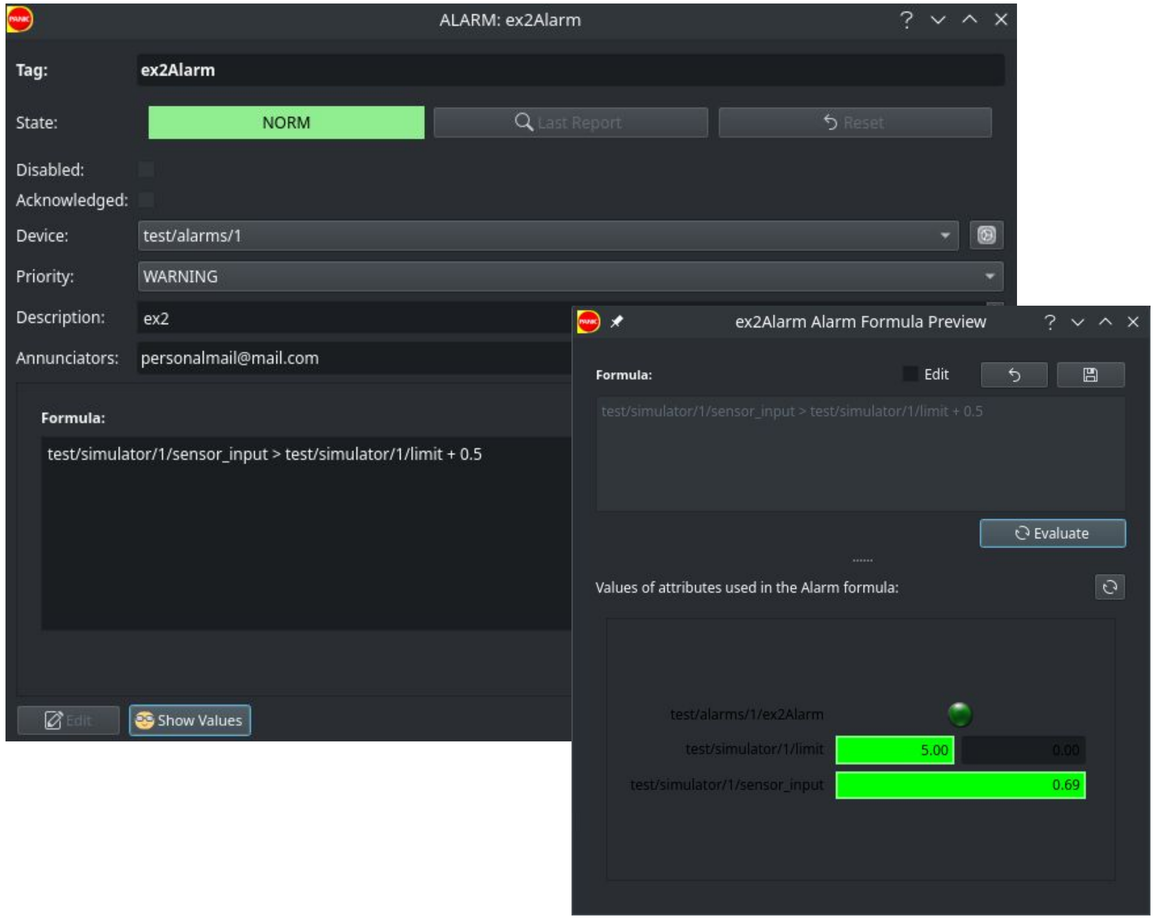
```
test/simulator/1/sensor_input > test/simulator/1/limit + 0.5
```

ALBA
www.cells.es

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

ex2Alarm Details:

- **Tag**: `ex2Alarm`
- **Device**: `test/alarms/1`
- **Priority**: `WARNING`
- **Description**: `Evaluate sensor input with hysteresis`
- **Annunciators**: `personalmail@mail.com`
- **Formula**:
  test/simulator/1/sensor_input > test/simulator/1/limit + 0.5



ALARM: ex2Alarm

Tag:	ex2Alarm
State:	NORM
Disabled:	☐
Acknowledged:	☐
Device:	test/alarms/1
Priority:	WARNING
Description:	ex2
Annunciators:	personalmail@mail.com

Formula:

```
test/simulator/1/sensor_input > test/simulator/1/limit + 0.5
```

ex2Alarm Alarm Formula Preview

Formula:

```
test/simulator/1/sensor_input > test/simulator/1/limit + 0.5
```

Values of attributes used in the Alarm formula:

test/alarms/1/ex2Alarm	●	
test/simulator/1/limit	5.00	0.00
test/simulator/1/sensor_input	0.69	

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

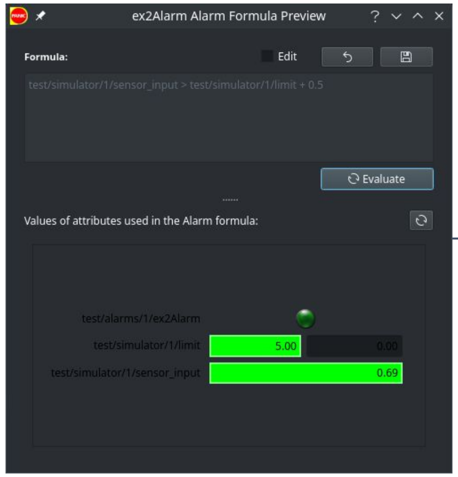
3. Plot the alarm

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

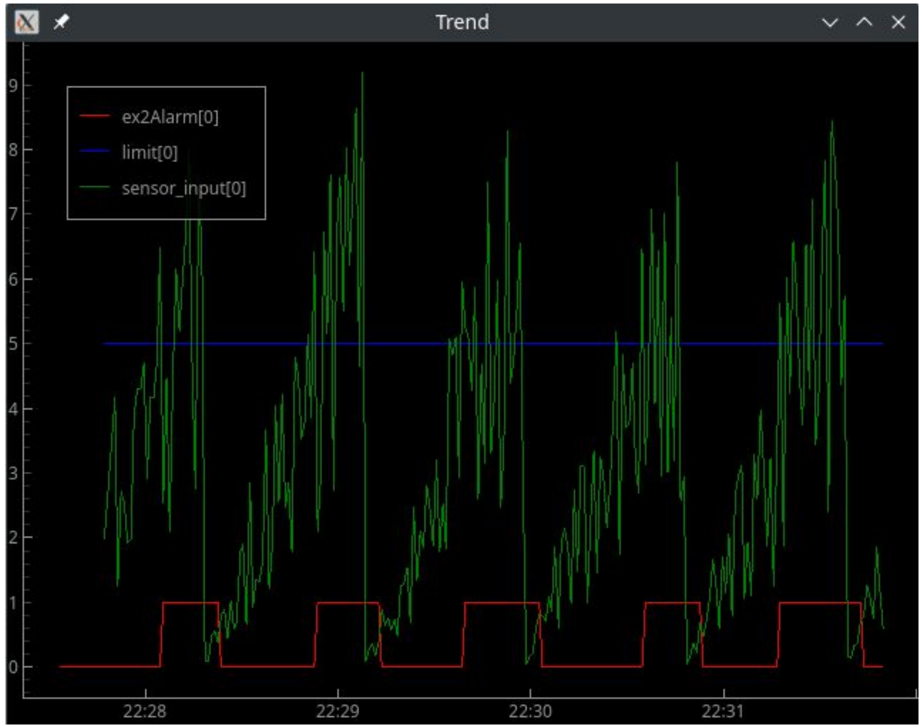


Formula:

```
test/simulator/1/sensor_input > test/simulator/1/limit + 0.5
```

Values of attributes used in the Alarm formula:

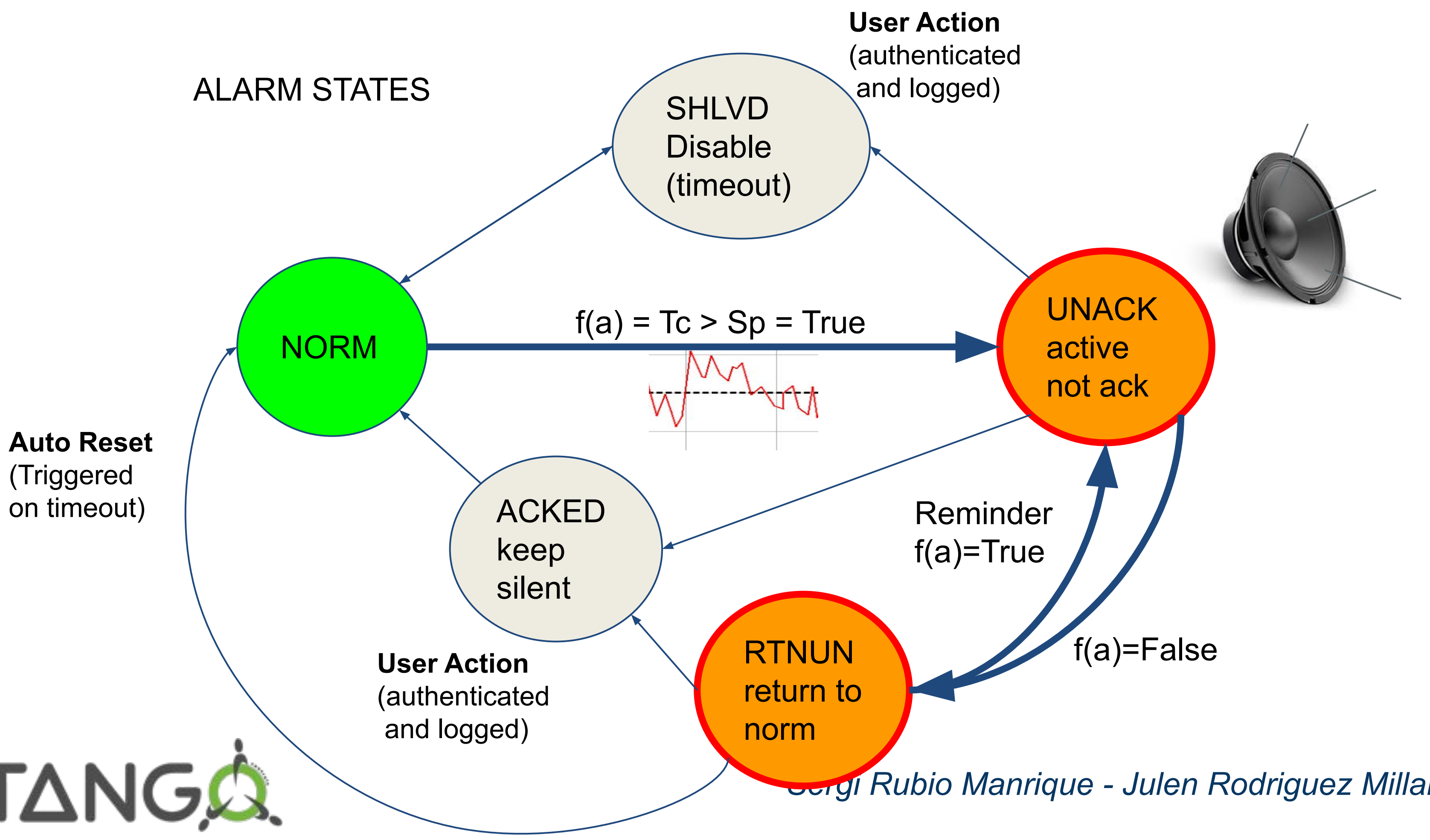
test/alarms/1/ex2Alarm	●	
test/simulator/1/limit	5.00	0.00
test/simulator/1/sensor_input	0.69	



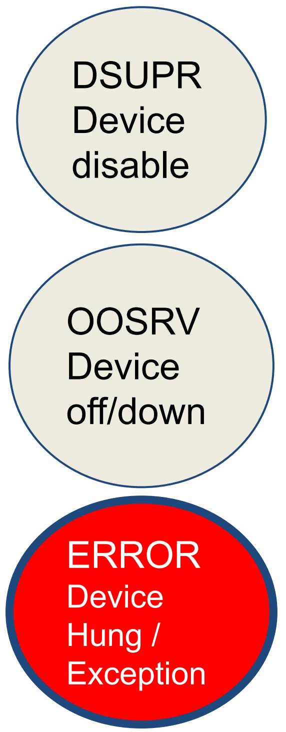
PANIC Alarm Cycle (IEC62682)

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.

ALARM STATES



SYSTEM STATES



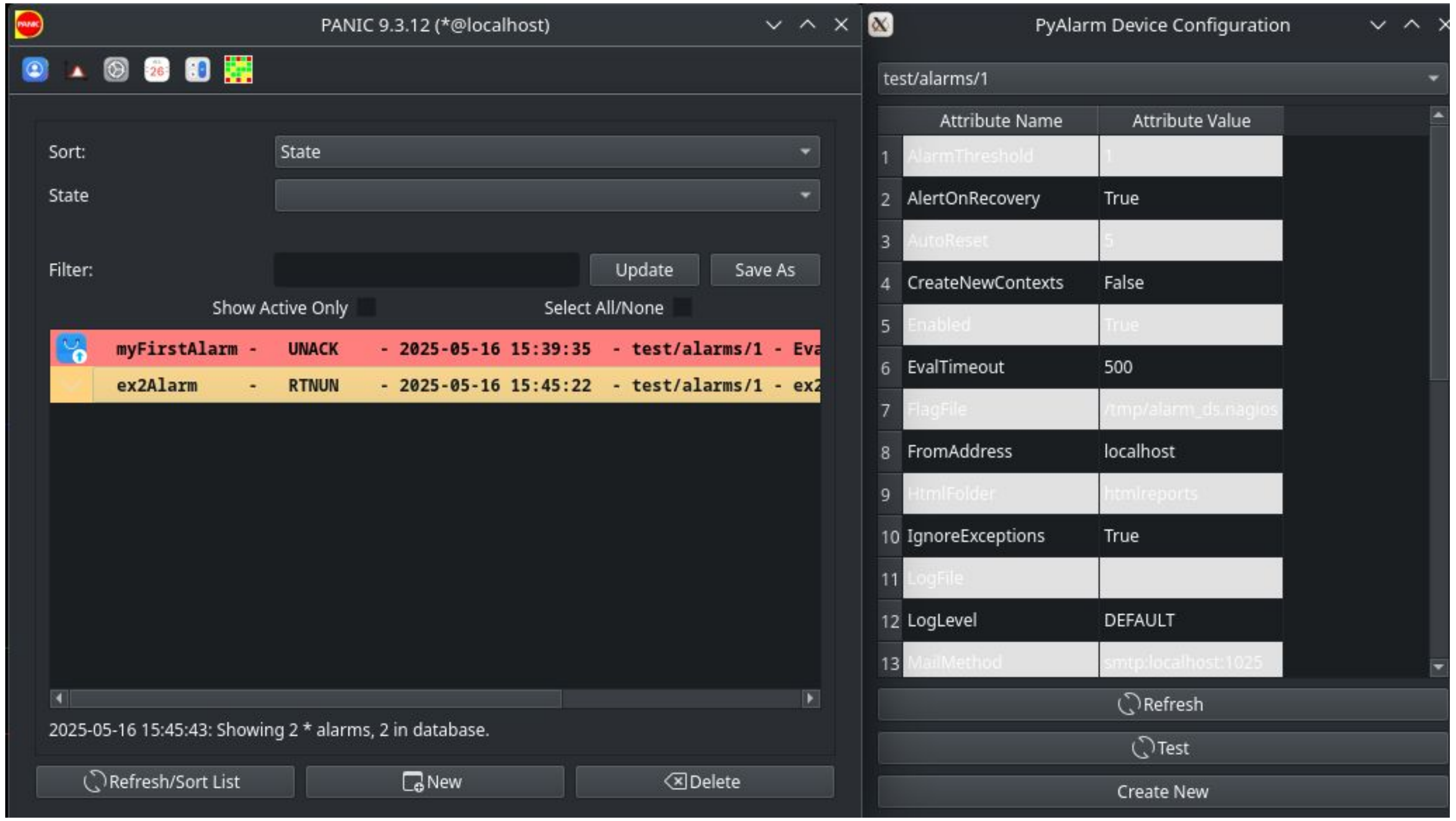
Alarms can be used to supervise the status of another PyAlarm.

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

5. Advanced Config

In the PANIC GUI, try to change the alarm properties by Advanced Config button in the alarm editor.

⚠️ The advanced config allows to change the alarm properties by hand. But the workshop is already configured to use the trainig defaults. So you can change the properties but then revert to the previous values so next exercises will be not affected.



```
myFirstAlarm -   UNACK   - 2025-05-16 15:39:35  - test/alarms/1 - Eva
ex2Alarm     -   RTNUN   - 2025-05-16 15:45:22  - test/alarms/1 - ex2
```

2025-05-16 15:45:43: Showing 2 * alarms, 2 in database.

test/alarms/1

	Attribute Name	Attribute Value
1	AlarmThreshold	1
2	AlertOnRecovery	True
3	AutoReset	5
4	CreateNewContexts	False
5	Enabled	True
6	EvalTimeout	500
7	FlagFile	/tmp/alarm_ds.nagios
8	FromAddress	localhost
9	HtmlFolder	htmlreports
10	IgnoreExceptions	True
11	LogFile	
12	LogLevel	DEFAULT
13	MailMethod	smtp:localhost:1025

EX.3 : Create an alarm depending on another

For this exercise, we want to create an alarm that depends on another alarm.

In this exercise, we simulate a situation where a detector is powered on while the beam intensity is too high and no attenuation filter is in place.

This combination could easily occur due to a user mistake or automation error, and can lead to:

- Permanent detector damage
- Sensor saturation
- Unusable data or triggered interlocks

The alarm aims to detect not just a high-intensity condition, but a risky action performed in that context — activating a sensitive device under unsafe beam conditions.

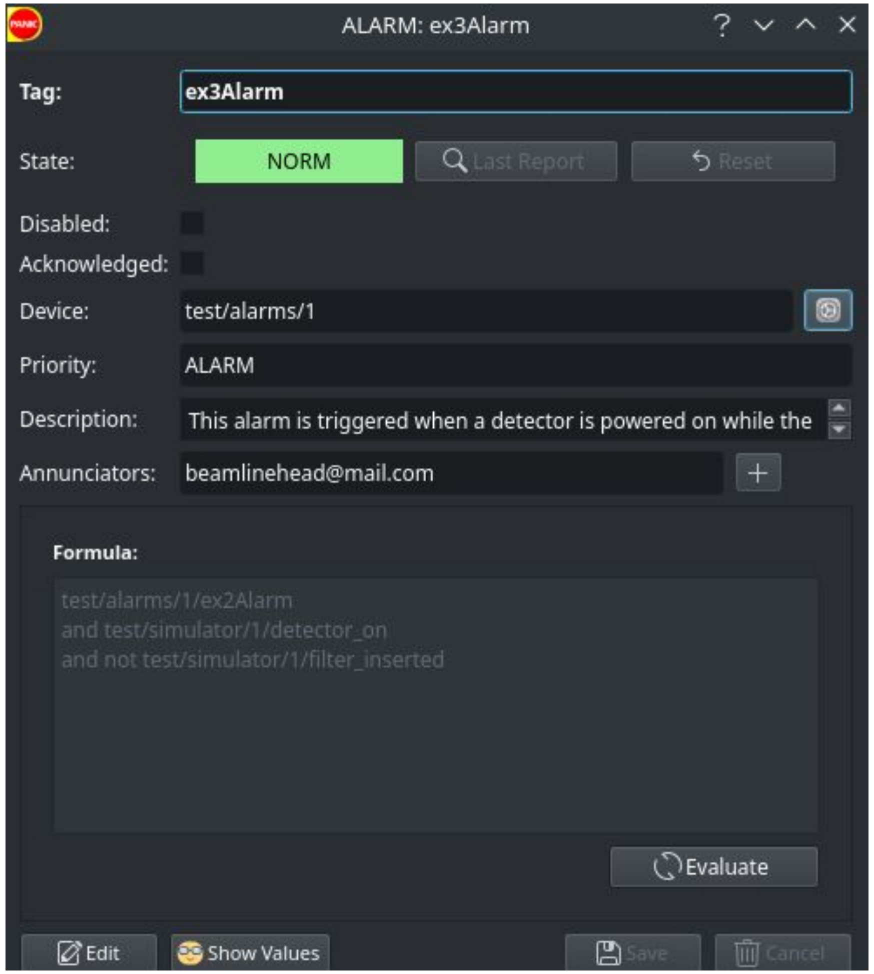
It demonstrates how alarms can capture procedural violations, not just limit exceedances.

It is necessary to use the previous exercise alarm to create this alarm as the beam intensity alarm.

EX.3 : Create an alarm depending on another

1. Create a new alarm

An alarm previously created represents its value in a Tango attribute in the PyAlarm device. This attribute is a boolean value that indicates if the alarm is active or not.


ALARM: ex3Alarm

Tag: ex3Alarm

State: NORM

Disabled:

Acknowledged:

Device: test/alarms/1

Priority: ALARM

Description: This alarm is triggered when a detector is powered on while the

Annunciators: beamlinehead@mail.com

Formula:

```
test/alarms/1/ex2Alarm
and test/simulator/1/detector_on
and not test/simulator/1/filter_inserted
```

EX.3 : Create an alarm depending on another

ex3Alarm Details:

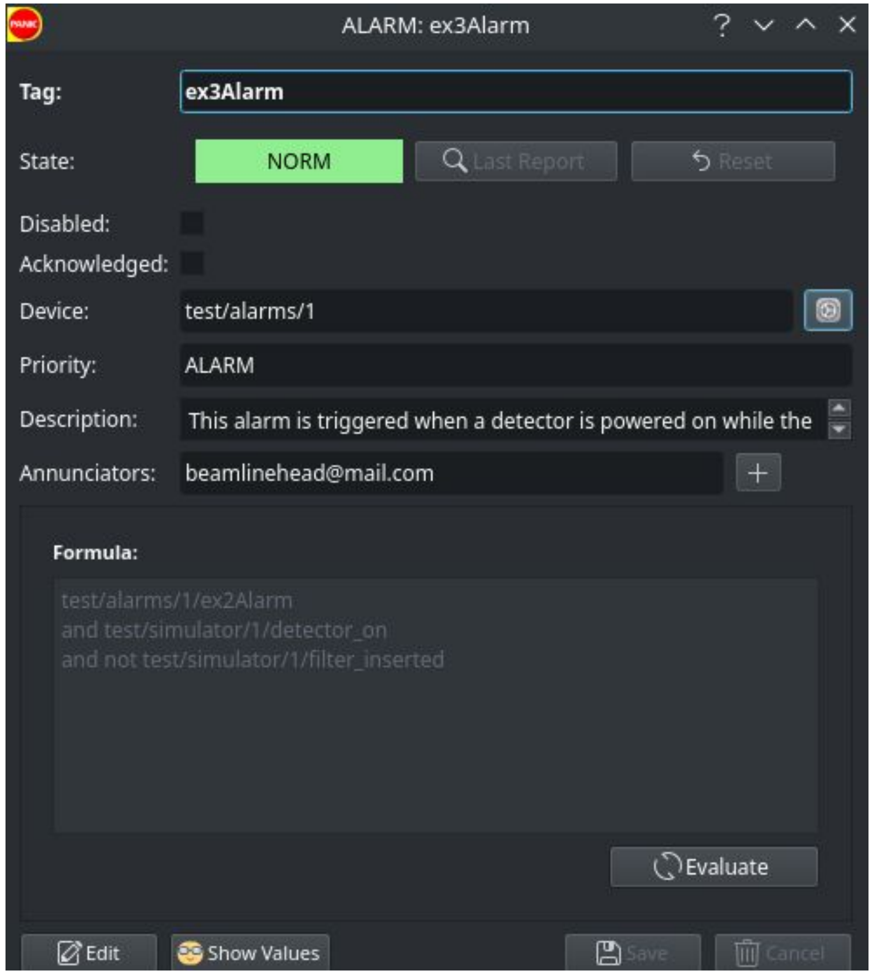
- **Tag**: `ex3Alarm`
- **Device**: `test/alarms/1`
- **Priority**: `ALARM`
- **Description**: `Troiggered when the beam intensity alarm is active, the detector is powered on and the filter is not in place`
- **Annunciators**: `beamhead@mail.com`

Formula:

test/alarms/1/ex2Alarm

and test/simulator/1/detector_on

and not test/simulator/1/filter_inserted


ALARM: ex3Alarm

Tag: ex3Alarm

State: NORM

Disabled:

Acknowledged:

Device: test/alarms/1

Priority: ALARM

Description: This alarm is triggered when a detector is powered on while the

Annunciators: beamlinehead@mail.com

Formula:

```
test/alarms/1/ex2Alarm
and test/simulator/1/detector_on
and not test/simulator/1/filter_inserted
```

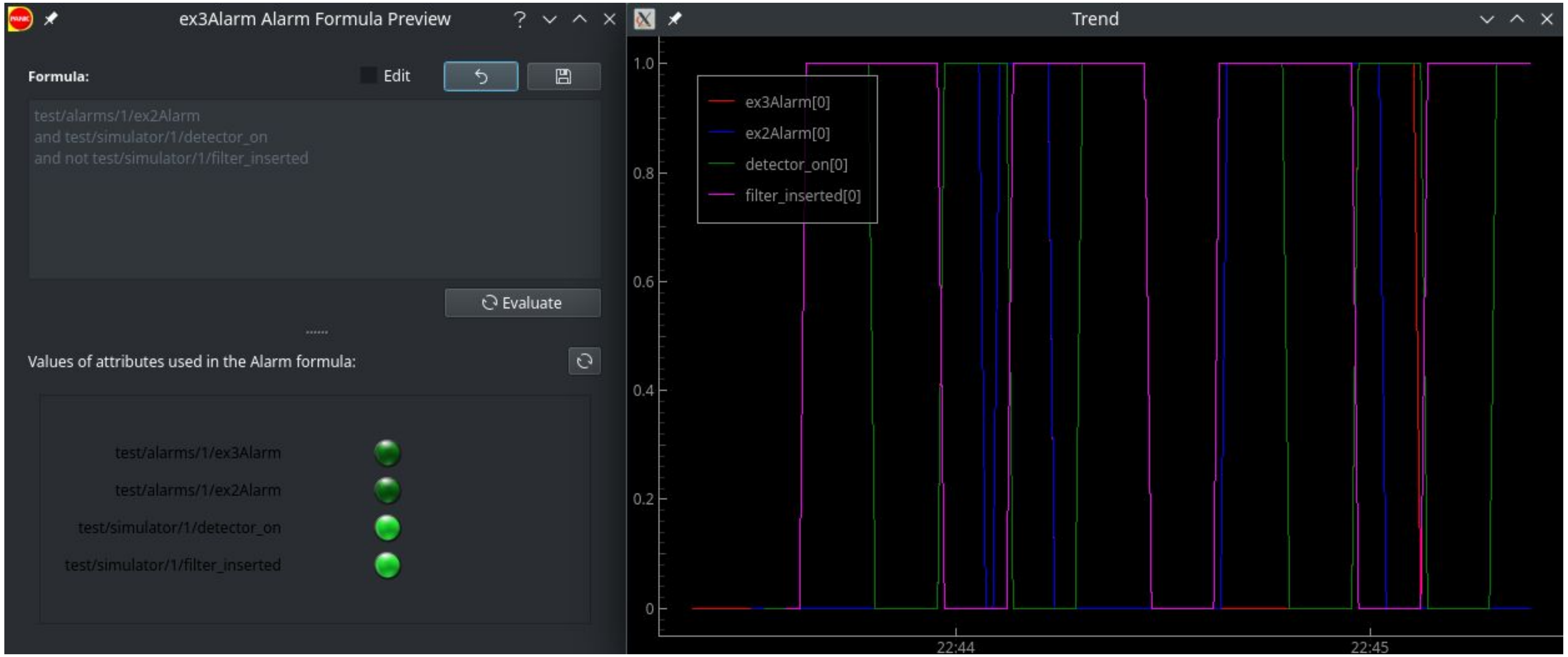
ALBA
www.cells.es

EX.3 : Create an alarm depending on another

2. Plot the alarm

The alarm is active when the beam intensity alarm is active, the detector is powered on, and the filter is not in place.

As in the previous exercise, this could serve as a warning to advise control room operators to check the beam intensity. This new alarm, however, could be directed to the beamline responsible to disable the detector and put the filter in place



Formula:

```
test/alarms/1/ex2Alarm
and test/simulator/1/detector_on
and not test/simulator/1/filter_inserted
```

Values of attributes used in the Alarm formula:

test/alarms/1/ex3Alarm	●
test/alarms/1/ex2Alarm	●
test/simulator/1/detector_on	●
test/simulator/1/filter_inserted	●

Legend:
- ex3Alarm[0]
- ex2Alarm[0]
- detector_on[0]
- filter_inserted[0]

TANGO

EX.4 : Create an alarm using regexp and Panic API

Create an alarm with regular expressions and panic api Panic API allows to create alarms with code. This is useful to programmatically create alarms automatically, in cases where the alarm configuration is too complex to be done by hand or imporve the mantainability and deployability of the alarms.

For this exercise, an alarm is created using regular expressions to match the attribute names. This is useful to create alarms that match a group of attributes with similar names.

The alarm is triggered when any of the previous exercise alarms are triggered. This is useful to create alarms that match a group of alarms with similar names.

1. Create a new alarm

```
python3 scripts/create_ex4AnyAlarmActive.py
```

EX.4 : Create an alarm using regexp and Panic API

scripts/create_ex4AnyAlarmActive.py

```python
import panic

# Connect to PANIC API
alarms = panic.api()
device = 'test/alarms/1'

# Alarm details
tag = 'ex4AnyAlarmActive'
formula = 'test/alarms/1/myFirstAlarm or any(FIND(test/alarms/1/ex*Alarm))'
description = 'Active if myFirstAlarm or any ex*Alarm is active.'
receivers = 'beamline@cells.es'
severity = 'ALARM'

# Create or update ex4AnyAlarm
if tag not in alarms:
   alarms.add(
       tag = tag,
       device = device,
       formula = formula,
       receivers = receivers,
       severity = severity,
       description = description
   )
   print(f'✅ Created alarm: {tag}')
else:
   alarms[tag].formula = formula
   alarms[tag].description = description
   alarms[tag].severity = severity
   alarms[tag].receivers = receivers
   print(f'♻️ Updated alarm: {tag}')
alarms[tag].write()
```

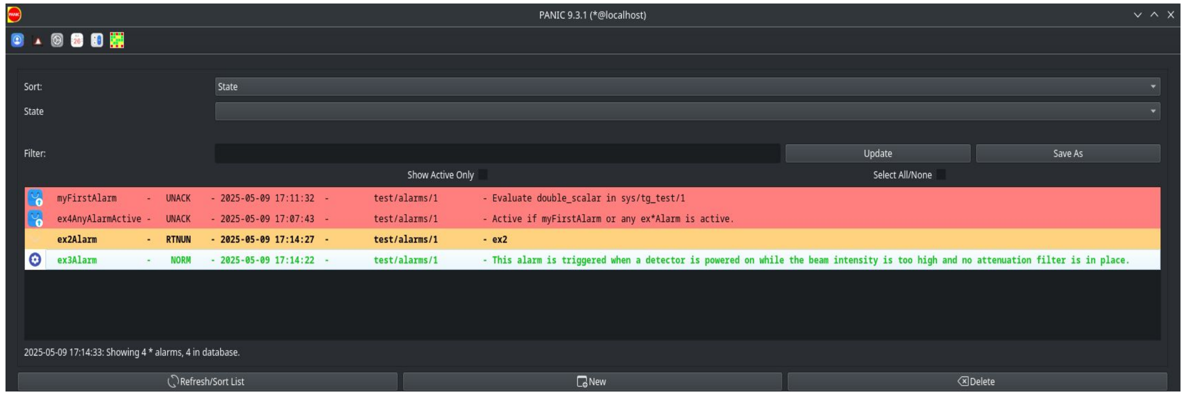
ALBA
www.cells.es

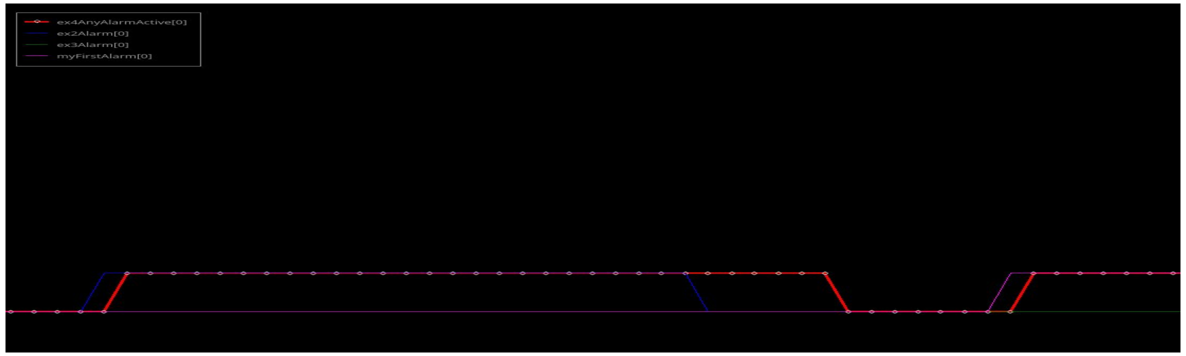
EX.4 : Create an alarm using regexp and Panic API

2. Formula validation

The formula is validated by the alarm evaluator. The formula is a regular expression that matches the attribute names of the ex*Alarm pattern and also myFirstAlarm.

```
test/alarms/1/myFirstAlarm
or any(FIND(test/alarms/1/ex*Alarm))
```


```
PANIC 9.3.1 (*@localhost)
Sort: State
State
Filter:                                   Update    Save As
                 Show Active Only                  Select All/None
myFirstAlarm       -  UNACK  - 2025-05-09 17:11:32 -  test/alarms/1  - Evaluate double_scalar in sys/tg_test/1
ex4AnyAlarmActive  -  UNACK  - 2025-05-09 17:07:43 -  test/alarms/1  - Active if myFirstAlarm or any ex*Alarm is active.
ex2Alarm           -  RTNUN  - 2025-05-09 17:14:27 -  test/alarms/1  - ex2
ex3Alarm           -  NORM   - 2025-05-09 17:14:22 -  test/alarms/1  - This alarm is triggered when a detector is powered on while the beam intensity is too high and no attenuation filter is in place.
2025-05-09 17:14:33: Showing 4 * alarms, 4 in database.
Refresh/Sort List    New    Delete
```



Play with the API to create
as many generic alarms
as you want (on max values from
tg_test, on min values, ...)

Many examples of alarms recipes
are available at:

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

TAN

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.

ALBA
www.cells.es

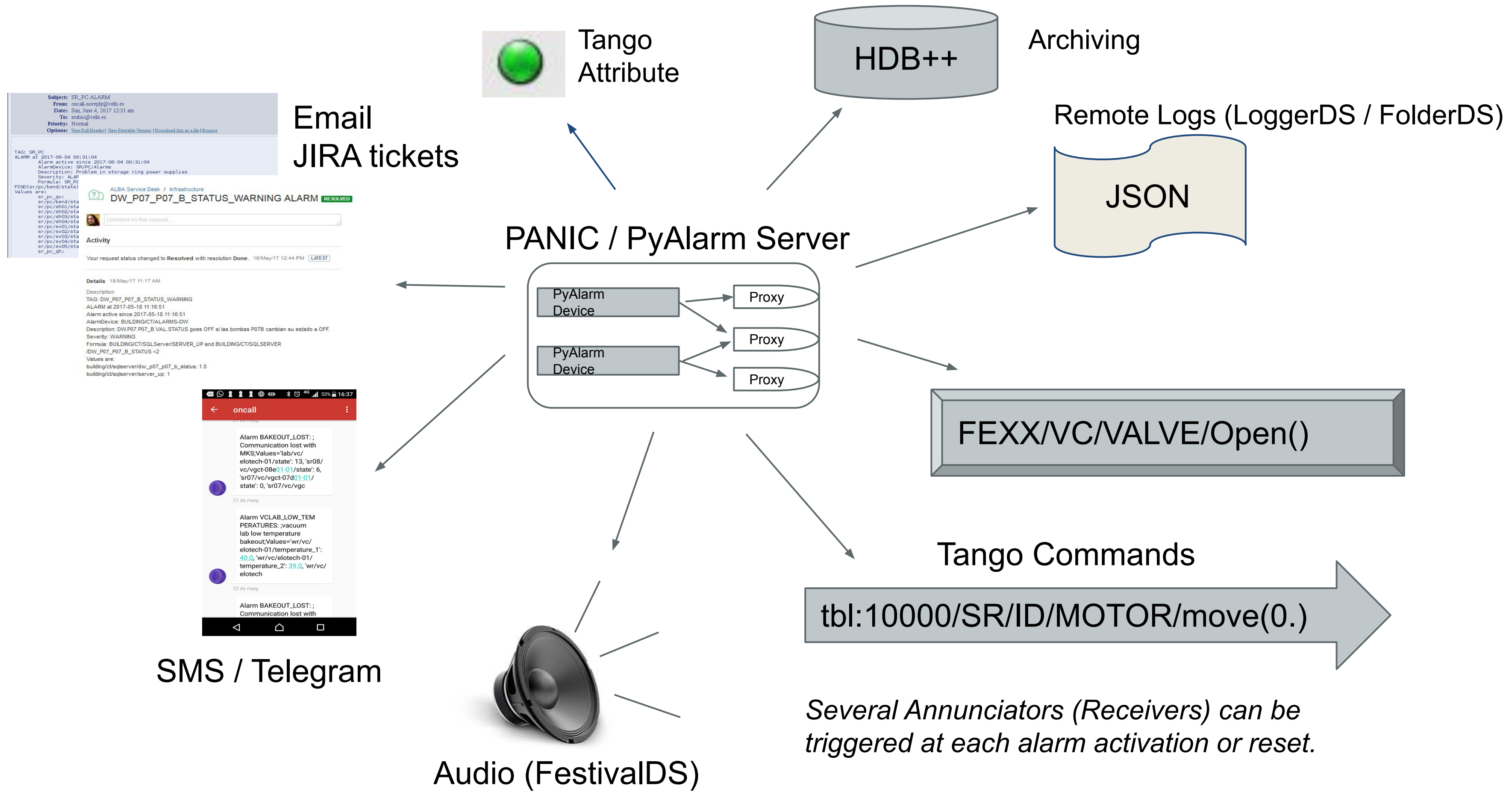
PANIC Notification Services

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 ...

TANGO

PANIC Notification Services



Subject: SR_PC ALARM
From: oncall-noreply@cells.es
Date: Sun, June 4, 2017 12:31 am
To: srubio@cells.es
Priority: Normal
Options: View Full Header | View Printable Version | Download this as a file | Bounce

```
TAG: SR_PC
ALARM at 2017-06-04 00:31:04
        Alarm active since 2017-06-04 00:31:04
        AlarmDevice: SR/PC/Alarms
        Description: Problem in storage ring power supplies
        Severity: ALAR
        Formula: SR_PC
FIND(sr/pc/bend/state)
Values are:
        sr_pc_qv:
        sr/pc/bend/sta
        sr/pc/sh01/sta
        sr/pc/sh02/sta
        sr/pc/sh03/sta
        sr/pc/sh04/sta
        sr/pc/sv01/sta
        sr/pc/sv02/sta
        sr/pc/sv03/sta
        sr/pc/sv04/sta
        sr/pc/sv05/sta
        sr_pc_qh:
```

ALBA Service Desk / Infrastructure
DW_P07_P07_B_STATUS_WARNING ALARM RESOLVED

Comment on this request...

Activity

Your request status changed to **Resolved** with resolution **Done**. 18/May/17 12:44 PM LATEST

Details 18/May/17 11:17 AM

Description
TAG: DW_P07_P07_B_STATUS_WARNING
ALARM at 2017-05-18 11:16:51
Alarm active since 2017-05-18 11:16:51
AlarmDevice: BUILDING/CT/ALARMS-DW
Description: DW.P07.P07_B.VAL.STATUS goes OFF si las bombas P07B cambian su estado a OFF.
Severity: WARNING
Formula: BUILDING/CT/SQLServer/SERVER_UP and BUILDING/CT/SQLSERVER
/DW_P07_P07_B_STATUS <2
Values are:
building/ct/sqlserver/dw_p07_p07_b_status: 1.0
building/ct/sqlserver/server_up: 1

oncall

Alarm BAKEOUT_LOST: ; Communication lost with MKS;Values='lab/vc/elotech-01/state': 13, 'sr08/vc/vgct-08e01-01/state': 6, 'sr07/vc/vgct-07d01-01/state': 0, 'sr07/vc/vgc

21 de maig

Alarm VCLAB_LOW_TEMPERATURES: ;vacuum lab low temperature bakeout;Values='wr/vc/elotech-01/temperature_1': 40.0, 'wr/vc/elotech-01/temperature_2': 39.0, 'wr/vc/elotech

22 de maig

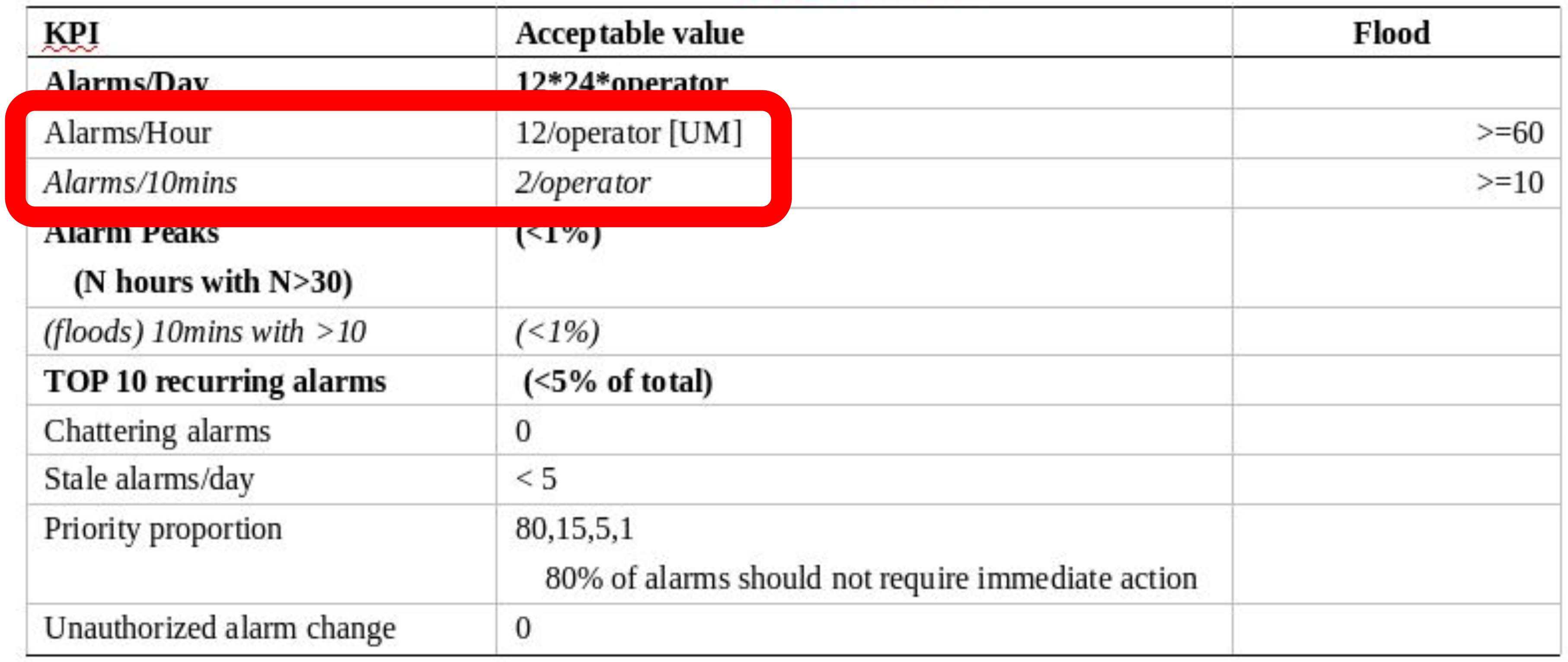
Alarm BAKEOUT_LOST: ; Communication lost with

Several Annunciators (Receivers) can be triggered at each alarm activation or reset.

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	

Extra Content: Telegram

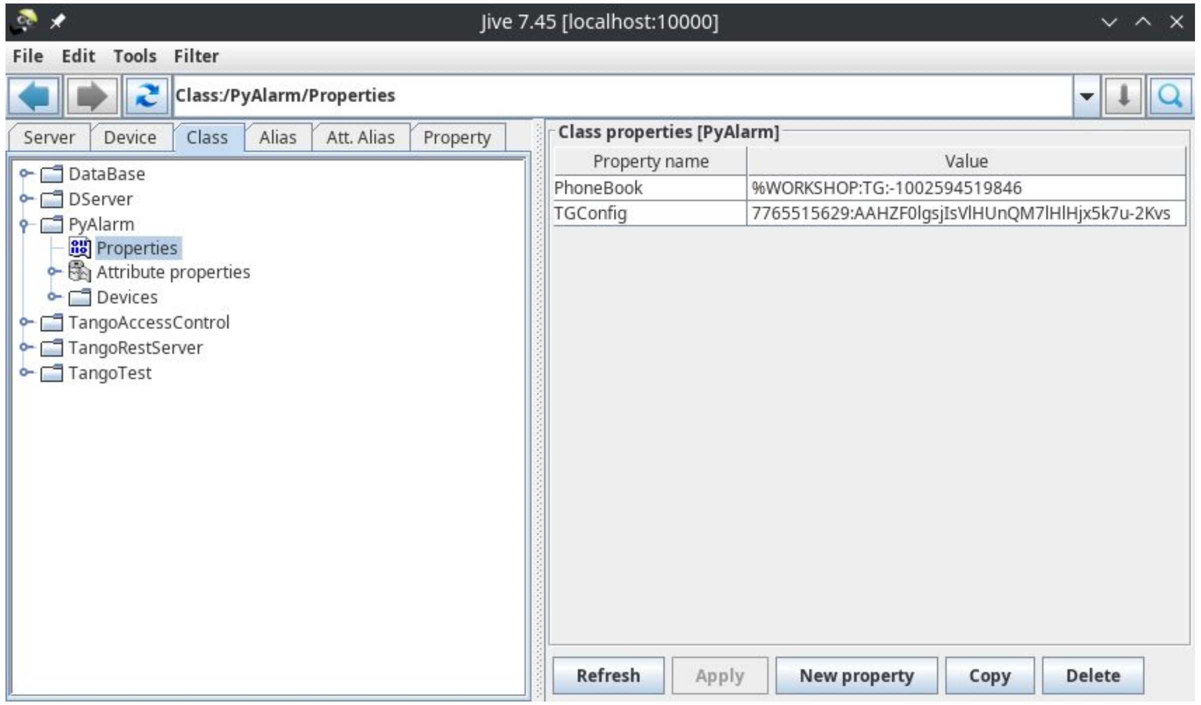
Create an alarm with Telegram notifications Panic allows to create alarms with Telegram notifications. This is useful to send notifications to a group of people or to a specific person.

This functionality needs a Telegram account and a bot configuration. To see an overview of this functionality, see the Telegram documentation:

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

Configuration of notification services may be quite complex, the Phonebook property and widget are used to hide this complexity from the user by defining aliases to common receivers.

To enable Telegram alarms on your local setup (only valid for this workshop):
- execute: python3 scripts/setup_telegram.py
- then, add %WORKSHOP as receiver of your alarms



Class properties [PyAlarm]

Property name	Value
PhoneBook	%WORKSHOP:TG:-1002594519846
TGConfig	7765515629:AAHZF0lgsjIsVlHUnQM7lHlHjx5k7u-2Kvs

ExtraContent: Festival and speech alarms

Create an alarm with physical notifications as a sound alarm

To create an alarm with physical notifications, we will use the FestivalDS device. This device allows to create alarms with sound or visual notifications.

1. Install festival

For Debian or Ubuntu:

```
sudo apt-get install festival
```

For MacOS:

```
brew install festvox/cli/festival
```

In order to test the festival DS device, try these commands

```
echo "hello world" | festival --tts
```

If this step fails, then try

```
sudo apt install pulseaudio-utils
echo "hello world" | padsp festival --tts
```

Remember which of the two commands worked, as we will have to set it up on the next step.

You may also use FestivalDS to play .wav files, this can be tested using

```
aplay /usr/share/sounds/speech-dispatcher/test.wav
```

ExtraContent: Festival and speech alarms

Create an alarm with physical notifications as a sound alarm

To create an alarm with physical notifications, we will use the FestivalDS device. This device allows to create alarms with sound or visual notifications.

2. Execute the script

The script set_up_festivalds.py creates a new server with the FestivalDS device and configures myFirstAlarm to do actions with the FestivalDS device.

```
python3 scripts/set_up_festivalds.py
```

After running the script, you should restart the pyAlarm server to see the changes.

> ⚠️ Remember to stop your pyAlarm server screen session usig screen -ls and screen -r <session_id>, using Ctrl+C to stop the server and then start it again with the command:

```
PyAlarm tango_meeting -v4
```

When myFirstAlarm is triggered, the FestivalDS device will play a sound saying "Caution! You are gaining more control over the PANIC system. Thank you for participating in this workshop."

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

How Alarm System is used?

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.

www.cells.es

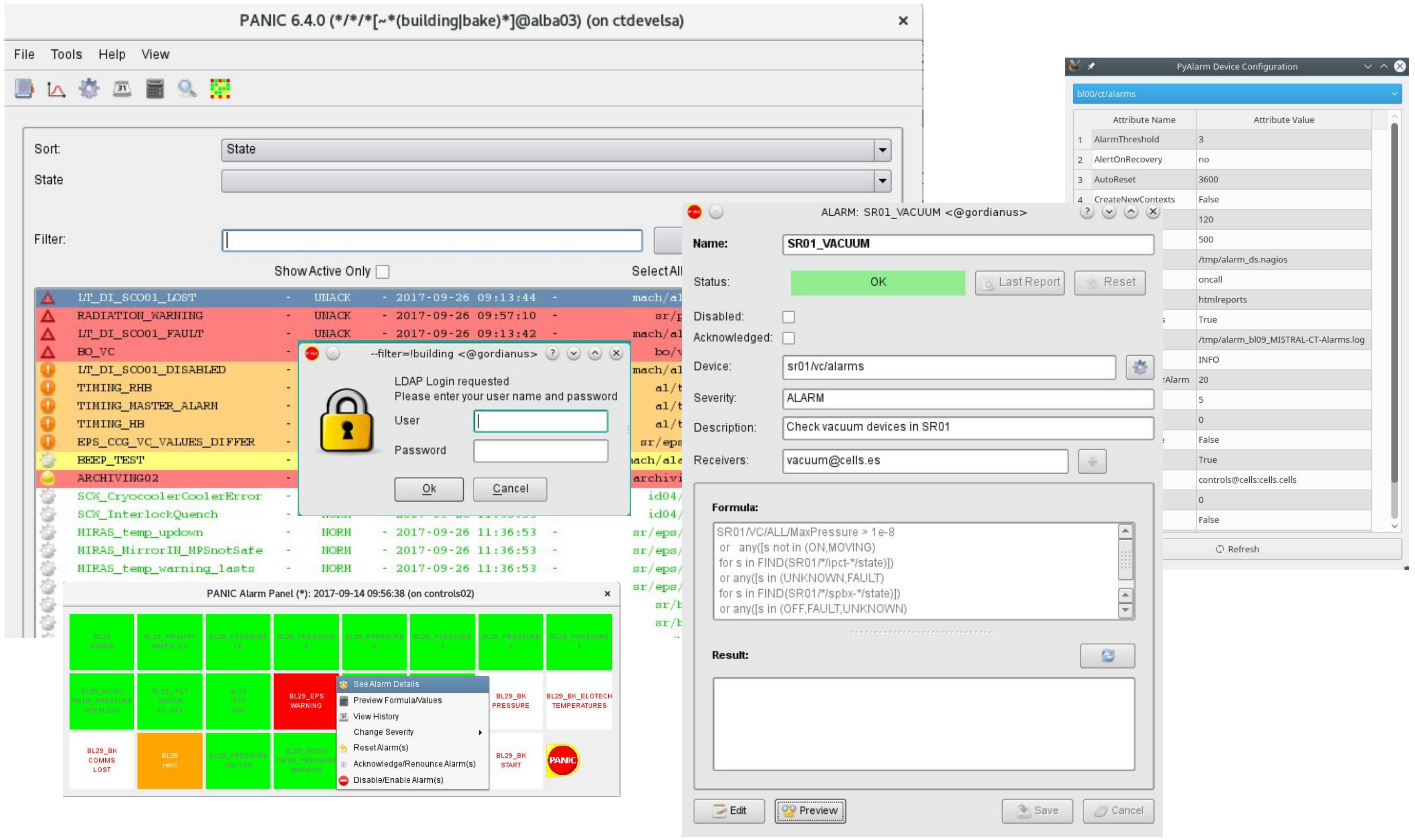
PANIC User Interface

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 6.4.0 (*/*/*[~*(building|bake)*]@alba03) (on ctdevelsa)

File Tools Help View

Sort: State
State
Filter:
Show Active Only ☐ Select All

```
LT_DI_SCO01_LOST             -  UNACK  - 2017-09-26 09:13:44 -   mach/al
RADIATION_WARNING            -  UNACK  - 2017-09-26 09:57:10 -   sr/p
LT_DI_SCO01_FAULT            -  UNACK  - 2017-09-26 09:13:42 -   mach/al
BO_VC                        -                                   bo/v
LT_DI_SCO01_DISABLED         -                                   mach/al
TIMING_RHB                   -                                   al/t
TIMING_MASTER_ALARM          -                                   al/t
TIMING_HB                    -                                   al/t
EPS_COG_VC_VALUES_DIFFER     -                                   sr/eps
BEEP_TEST                    -                                   mach/ala
ARCHIVING02                  -                                   archivi
SCW_CryocoolerCoolerError    -                                   id04/
SCW_InterlockQuench          -                                   id04/
MIRAS_temp_updown            -  NORM   - 2017-09-26 11:36:53 -   sr/eps/
MIRAS_MirrorIN_MPSnotSafe    -  NORM   - 2017-09-26 11:36:53 -   sr/eps/
MIRAS_temp_warning_lasts     -  NORM   - 2017-09-26 11:36:53 -   sr/eps/
```

--filter=!building <@gordianus>

LDAP Login requested
Please enter your user name and password
User
Password
Ok Cancel

PANIC Alarm Panel (*): 2017-09-14 09:56:38 (on controls02)

ALARM: SR01_VACUUM <@gordianus>

Name: SR01_VACUUM
Status: OK Last Report Reset
Disabled: ☐
Acknowledged: ☐
Device: sr01/vc/alarms
Severity: ALARM
Description: Check vacuum devices in SR01
Receivers: vacuum@cells.es

Formula:

```
SR01/VC/ALL/MaxPressure > 1e-8
 or any([s not in (ON,MOVING)
 for s in FIND(SR01/*/ipct-*/state)])
 or any([s in (UNKNOWN,FAULT)
 for s in FIND(SR01/*/spbx-*/state)])
 or any([s in (OFF,FAULT,UNKNOWN)
```

Result:

Edit Preview Save Cancel

PyAlarm Device Configuration

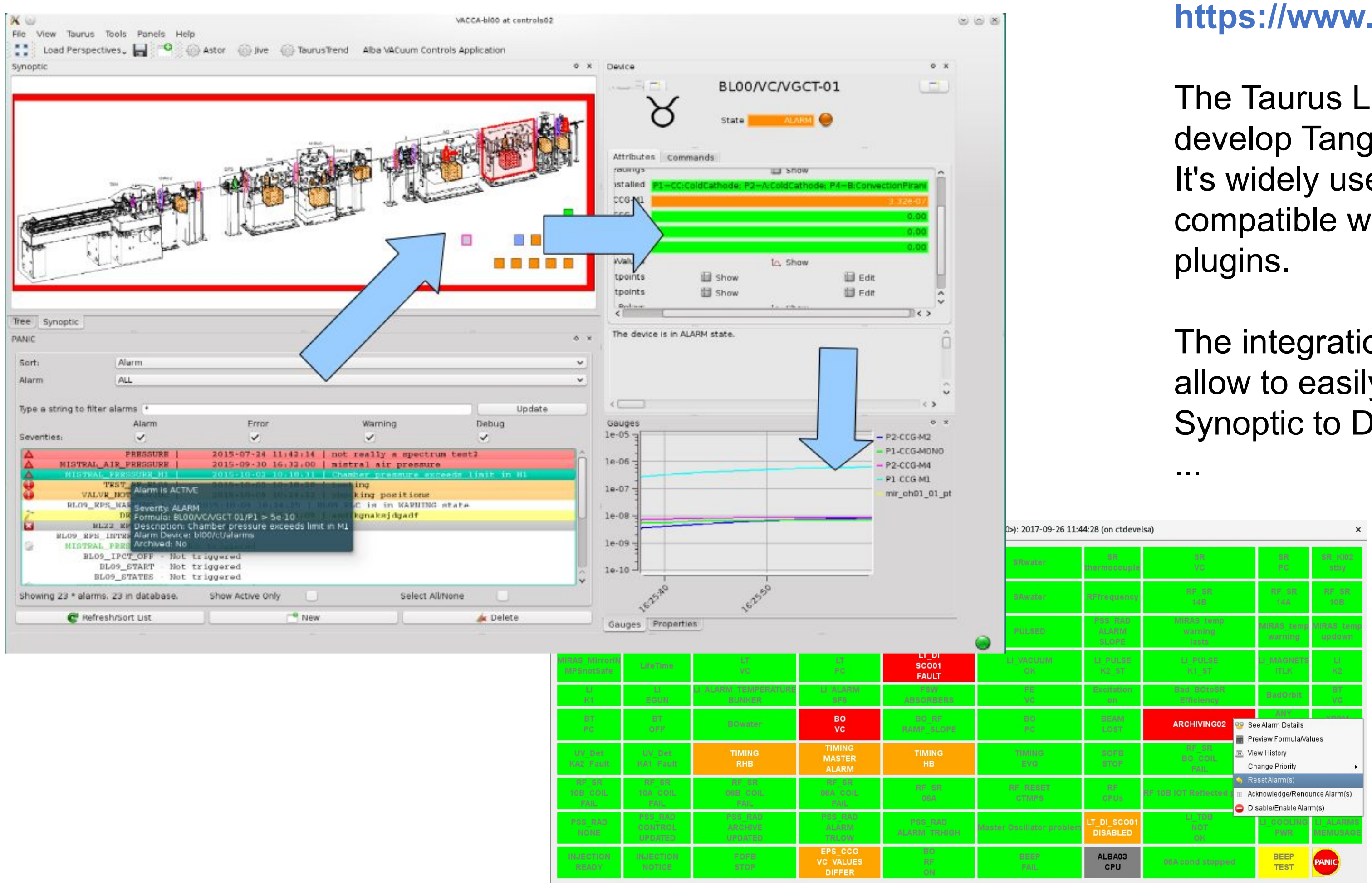
bl00/ct/alarms

	Attribute Name	Attribute Value
1	AlarmThreshold	3
2	AlertOnRecovery	no
3	AutoReset	3600
4	CreateNewContexts	False
		120
		500
		/tmp/alarm_ds.nagios
		oncall
		htmlreports
		True
		/tmp/alarm_bl09_MISTRAL-CT-Alarms.log
		INFO
	rAlarm	20
		5
		0
		False
		True
		controls@cells:cells.cells
		0
		False

Refresh

ALBA
www.cells.es

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, ...

ALBA

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.**

ALBA
www.cells.es

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!

ALBA

www.cells.es

srubio@cells.es
jrodriguez@cells.es

Cerdanyola del Vallès (Barcelona) Spain

Tel: (+34) 93 592 43 00