

# PANIC, The ALBA Alarm System



**Package for Alarms  
and Notification of Incidences  
from Controls**

*Sergi Rubio Manrique, ALBA Controls Section*

What an Alarm System should do:

- Verification of a set of conditions.
- Notification.
- Keep a log of what happened.
- Take automatic actions?
- Tools for configuration/visualization.

# Alarm Life Cycle

- Condition: CIRCE\_PRESSURE:BL24/VC/VGCT-01/P1>3e-5

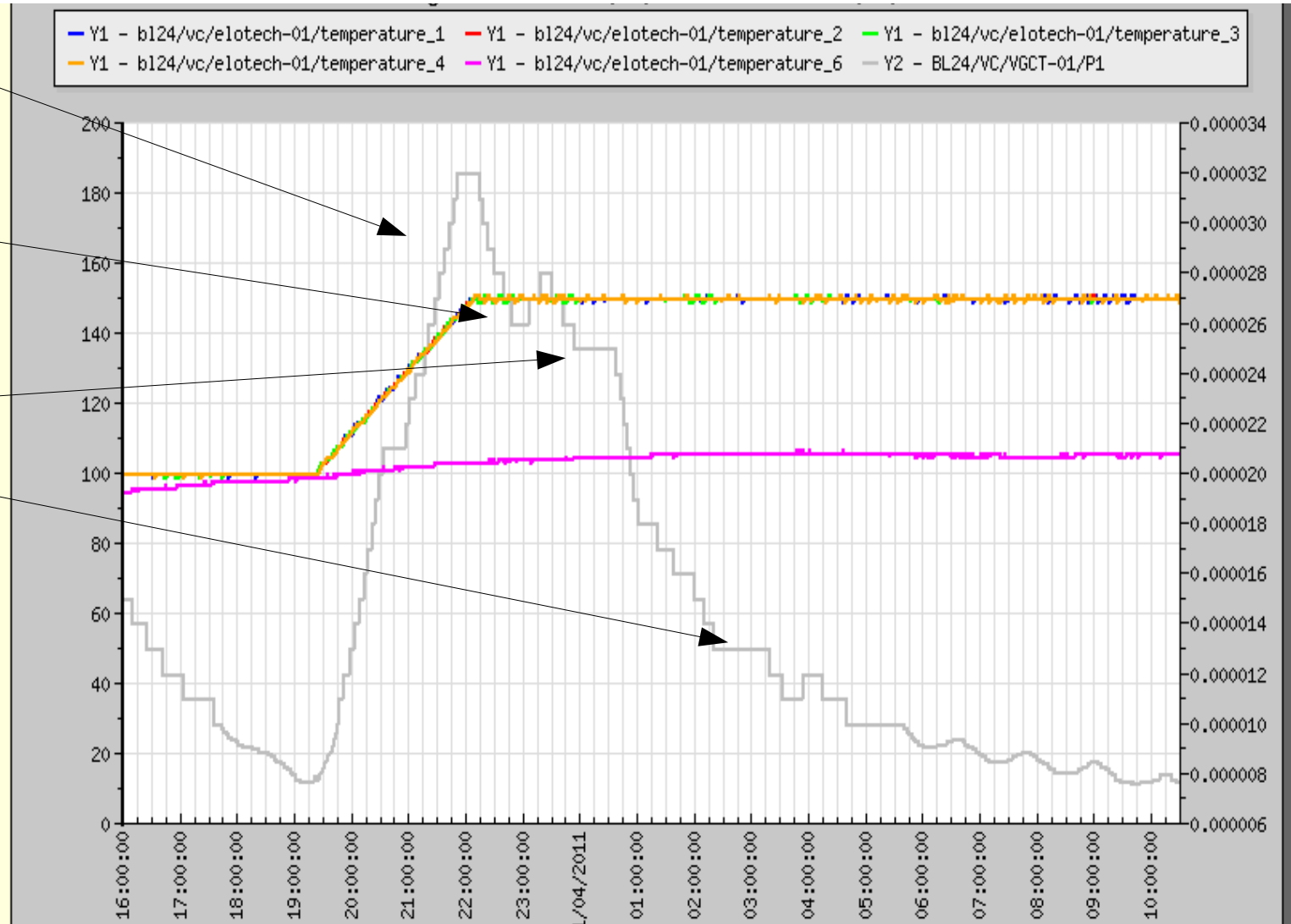
- Incidence  
- Notification  
- Logging

- Recovery  
- Reminder

- Acknowledge

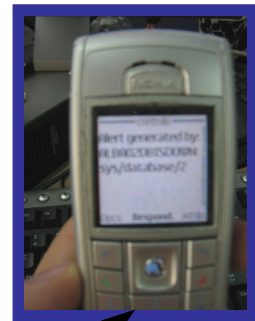
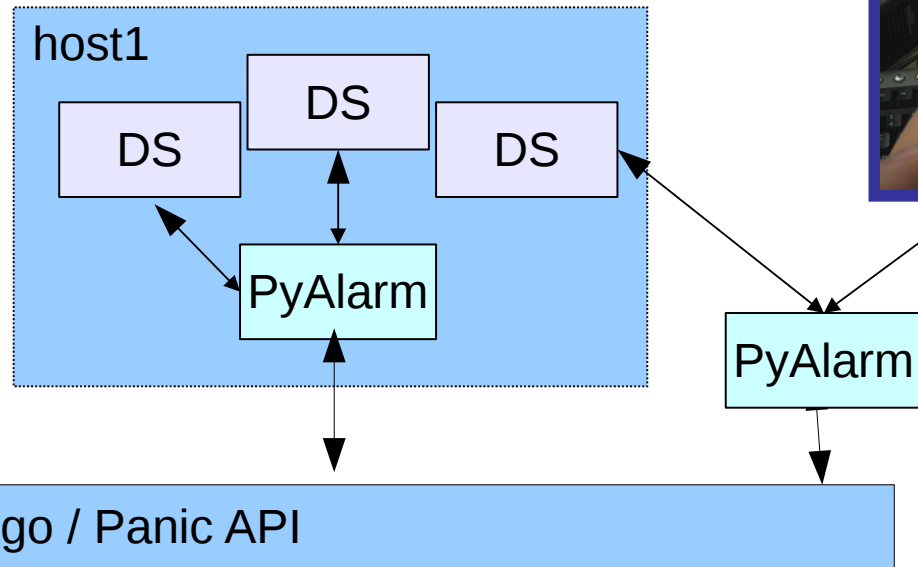
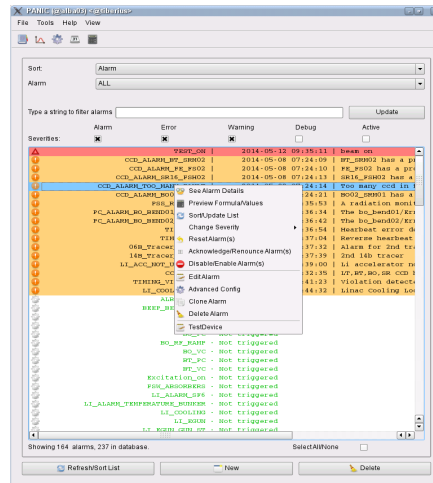
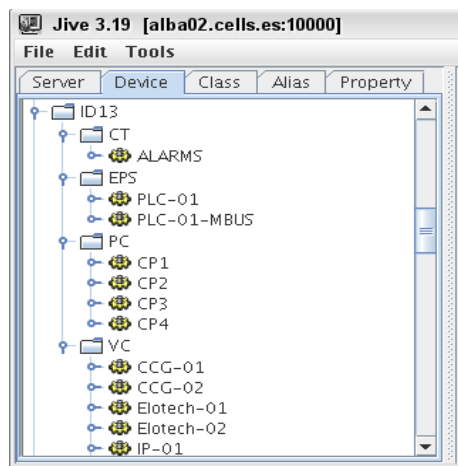
- Reset  
- Logging

- Further Actions



# PANIC, The Alba Alarm System

- Distributed in PyAlarm Device Servers, each managing a collection of **alarm formulas**.
- **Each Alarm is unique** in the system and can be managed by only 1 PyAlarm device.
- Each PyAlarm device performs locally **both Logging and Notification (email/SMS)**
- **Additional actions** are passed to **external devices** (SnapArchiver, PopupDevice)
- Configuration is stored in the **Tango Database**.
- **Alarm logging** in the Tango **Snapshotting** database.
- The Panic API provides an **homogeneous view** of the system for devices and GUI.



Database

BL24/VC/VGCT-01/P1>3e-5

# Yet another Alarm System??

The core of Panic is the **PyAlarm device**, developed in 2007 as an SMS Notification Device for **Tango Alarm System (Elettra's)**

Both Panic and Elettra Alarm Systems evaluate **formulas of Tango attributes** and return boolean results. Both are capable of **triggering commands on other devices**.

Unlike Tango Alarm System (as it was in 2007), **PANIC is distributed and doesn't require other database than Tango**. It was developed to use no-db device servers in isolated machines

Like Soleil's alarm system it can be configured to trigger **alarms based on Attribute qualities**, and do not require any client as the devices operate stand-alone.

The image shows two screenshots from a Tango Alarm System interface. The top screenshot is the 'Tango Log Viewer' window, displaying a list of log entries with columns for Time, Level, Source, and Message. The bottom screenshot is the 'Alarms' window, showing a table of active alarms with columns for Time, Alarm, Status, Ack, Message, and Acknowledge. The table contains several entries for 'sr/psciads1.2/fault' and 'sr/psciads1.3/fault' with status 'ALARM' and 'NOT ACK'.

Time	Alarm	Status	Ack	Message	Acknowledge
Mon Sep 19 10:25:50 2005	sr/psciads1.2/fault	NORMAL	NOT ACK		Selected
Mon Sep 19 10:25:50 2005	sr/psciads1.2/fault	ALARM	ACK	messaggio di prova	Selected
Mon Sep 19 10:25:50 2005	sr/psciads1.3/fault	NORMAL	NOT ACK		All
Mon Sep 19 10:25:50 2005	sr/psciads1.3/fault	ALARM	ACK	messaggio di prova	
Mon Sep 19 10:25:50 2005	sr/psciads1.4/fault	NORMAL	NOT ACK		
Mon Sep 19 10:25:50 2005	sr/psciads1.4/fault	ALARM	NOT ACK	messaggio di prova	
Mon Sep 19 10:25:50 2005	sr/psciads1.6/fault	NORMAL	NOT ACK		
Mon Sep 19 10:25:50 2005	sr/psciads1.6/fault	ALARM	NOT ACK	messaggio di prova	
Mon Sep 19 10:25:50 2005	sr/psciads1.7/fault	NORMAL	NOT ACK		
Mon Sep 19 10:25:50 2005	sr/psciads1.7/fault	ALARM	NOT ACK	messaggio di prova	
Mon Sep 19 10:25:50 2005	sr/psciads1.8/fault	NORMAL	NOT ACK		
Mon Sep 19 10:25:51 2005	sr/psciads1.8/fault	ALARM	NOT ACK	messaggio di prova	
Mon Sep 19 10:25:51 2005	sr/psciads1.5/fault	ALARM	NOT ACK	messaggio di prova	



## Declaring Alarms

Each Alarm is defined by:

TAG,  
Formula,  
Description,  
Receivers,  
Severity,  
{Configuration})

Those are stored using **device properties** of PyAlarm devices.

The screenshot shows the configuration window for an alarm. The 'Name' field is 'CIRCE\_PRESSURE' with a green 'OK' button and an 'Acknowledge' button. The 'Device' is 'bl24-circe/ct/alarms'. The 'Description' is 'Chamber pressure exceeds limit'. The 'Receivers' are '%VACMV,%CTRLMV,%VIRGINIA,%CTRL2'. The main configuration area shows a logical expression: 'tbl2401:10000/BL24/VC/VGCT-01/P1>3e-5 OR tbl2401:10000/BL24/VC/VGCT-01/P2>3e-5'. Below this, there are two rows of configuration for each part of the OR expression. The first row is for 'tbl2401:10000/BL24/VC/VGCT-01/P1' with a '>' operator and a '3e-5' value. The second row is for 'tbl2401:10000/BL24/VC/VGCT-01/P2' with a '>' operator and a '3e-5' value. There are 'Add Expression', 'Add Relation', 'Raw Edit', and 'Clear' buttons. At the bottom, there are 'Edit', 'Save', 'Cancel', and 'Close' buttons.

## Alarm formulas:

CIRCE\_PRESSURE: **tbl2401:10000/BL24/VC/VGCT-01/P1>3e-5**

CIRCE\_LOST: BL24/VGCT-01/State == **UNKNOWN**

CIRCE\_TEMP: BL24/EPS/PLC-01/T1.quality == **ATTR\_ALARM**

CIRCE\_VALVE: FE24/VC/PNV-01/State.**delta!=0**

# Declaring Alarms with Regular Expressions

```
any([t>85 for t in FIND(ID13/EPS/PLC-01/TTAP*_VAL)])
```

```
any([min(t)<50 and 70<max(t)<1000 for t in  
[FIND(ID13/VC/Elotech*/Temperature_[0-9])]])
```

```
any([t==ATTR_ALARM for t in  
FIND(ID13/VC/Elotech*/Temperature_[0-9].quality)])
```

```
GROUP(BL24/CT/ALARMS/CIRCE_*)
```

Full Tango Attribute URL:

```
[tango:host/][device/]Attribute[.value/quality/delta/time/exception/all]g
```

## Yet another fandango.eval format ...

### fandango.dynamic.DynamicDS

(PyPLC, PySignalSimulator, PyAttributeProcessor, PyStateComposer, ...)

MaxP=DevBoolean(XATTR('tbl24:10000/BL24/VC/VGCT-01/P1')>MaxV)

CCGs=DevVarStringArray([d+'/P'+a for a in '12' for d in DEVICES if 'VG' in d])

AlarmP=DevBoolean(any(Q(c) for c in CCGs))

AnyP=DevBoolean(MaxP or AlarmP)

It allows to use commands and attributes **within a Tango device**.

Parsed/executed by Device commands, tied to **DevImpl** class methods

Variables **isolated** between instances, **kept only when needed**

### fandango.tango.TangoEval (PyAlarm , Panic UI)

MaxP=BL24/VC/VGCT-01/P1>3e-5

AlarmP=any(q in (ATTR\_ALARM,) for q in FIND(BL/VC/VG-\*/P\*.quality))

AnyP=MaxP or AlarmP

**Same eval** object within **devices and clients**

Simplified syntax, enabling **wildcards** and **delta/quality/time** comparison

Keeps cache and values for all evaluated objects, **shared between instances**

Both approaches use lazy **polling by expiration date** (CachedAttributeProxy)





# The Panic Module

Panic contains the python AlarmAPI for managing the [PyAlarm](#) device servers from a client application or a python shell. The panic module is part of the Panic bliss package.

```
import panic
alarms = panic.api()
```

## Browsing existing alarms

The AlarmAPI is a dictionary-like object containing Alarm objects for each registered Alarm tag. In addition the AlarmAPI.get method allows caseless search by tag, device, attribute or receiver:

```
alarms.get(self, tag='', device='', attribute='', receiver='')
```

```
alarms.get(device='boreas')
```

```
Out[232]:
[Alarm(BL29-BOREAS_STOP:The BakeOut controller has been stop),
 Alarm(BL29-BOREAS_PRESSURE_1:),
 Alarm(BL29-BOREAS_PRESSURE_2:),
 Alarm(BL29-BOREAS_START: BL29-BOREAS bakeout started
 ...]
```

```
alarms.get(receiver='eshraq')
```

```
Out[234]:
[Alarm(RF_LOST_EUROTHERM:),
 Alarm(OVEN_COMMS_FAILED:Oven temperatures not updated in the last
 5 minutes),
 Alarm(RF_PRESSURE:The pressure in the cavity exceeds Range),
 Alarm(OVEN_TEMPERATURE:The Temperature of the Oven exceeds
 Range),
 Alarm(RF_EUROTHERM:),
 Alarm(RF_LOST_MKS:),
 Alarm(RF_TEMPERATURE_MAX2:),
 ...]
```

```
alarms['RF_LOST_MKS'].receivers
```

```
Out[237]: 'SRUBIO,%ESHRAQ,%VACUUM,%LOTHAR,%JNAVARRO'
```

**Panic API**  
Package for Alarms and Notification of Incidences from Controls

home | examples | screenshots | documentation » previous | next | modules | index

### Panic API

panic – The Package for Alarms and Notification of Incidences from Controls

class panic.AlarmAPI(filters=PyAlarm/\*)  
Panic API is a dictionary-like object

load(filters=None)  
Reloads all alarm properties from the database ;param filters: is used to specify which devices to be loaded

purge(device, tag, load=False)  
Removes any alarm from a device matching the given tag. Database must be reloaded afterwards to update the alarm list.

remove(tag, load=True)  
Removes an alarm from the system.

rename(tag, new\_tag="", new\_device="")  
Renames an existing tag, it also allows to move to a new device.

save\_tag(tag)  
Shortcut to force alarm update in database

panic.SetProxy  
The \_proxies object allows to retrieve either DeviceProxy or DeviceServer objects.

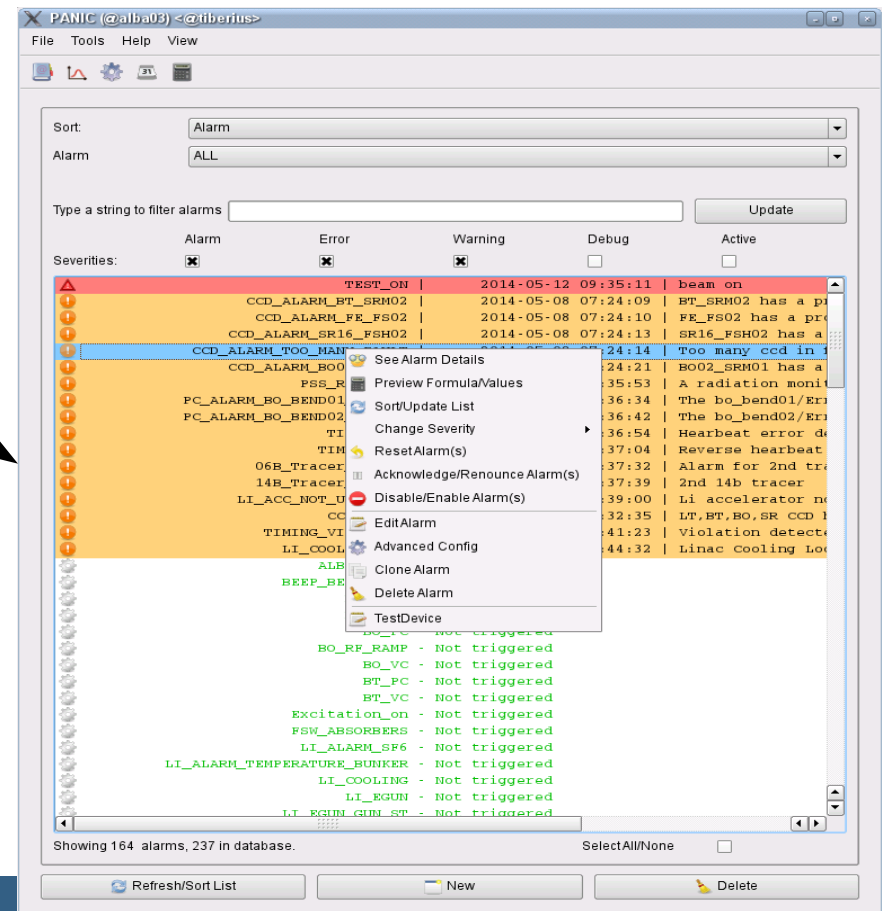
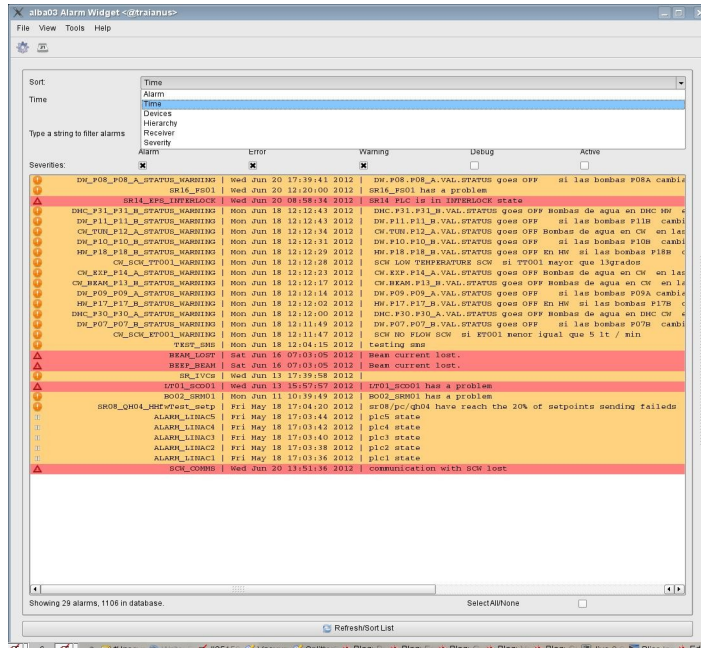
Table of Contents  
Panic API  
■ panic – The Package for Alarms and Notification of Incidences from Controls  
Previous topic  
Welcome to Panic's documentation!  
Next topic  
PyAlarm Tango Device Server  
This Page  
Show Source  
Quick search  
Enter search terms or a module, class or function name.

# Panic User Interface

The Panic tool shows the list of active or declared alarms. It provides several filters to search alarms: by state (active/inactive), severity, subsystem, receiver or historic values.

A text search is also provided that allow to locate alarms by any of the attributes used in formula or words used in description.

For each alarm the menu allows to Reset the alarm or show the attribute values that triggered it.



# UI Editor and Preview

ALARM: CCD\_ALARM\_TOO\_MANY\_FAULT <@tiberius>

Name:

Status: ALARM

Disabled:

Acknowledged:

Device:

Severity:

Description:

Receivers:

**Formula:**

```
mach/di/ccdmonitor-01/FaultCameras > 0  
AND mach/di/ccdmonitor-01/FaultCamerasList != None
```

**Result:**

CCD\_ALARM\_TOO\_MANY\_FAULT Alarm Formula Preview <@ti>

**Formula:**

```
mach/di/ccdmonitor-01/FaultCameras > 0  
AND mach/di/ccdmonitor-01/FaultCamerasList != None
```

**Result:**

True

.....

Values of attributes used in the Alarm formula:

mach/di/ccdmonitor-01/FaultCameras	<span style="background-color: green; color: white; padding: 2px;">1</span>
mach/di/ccdmonitor-01/FaultCamerasList	<input type="button" value="Show"/>

# Record/View Alarm History using Snaps

If alarm history is enabled, (**SNAP Receiver** or **UseSnaps+CreateNewContext** property) then attribute values will be recorded every time that the alarm is triggered.

The alarm will create a context in the Tango Snapshotting database with all the attributes that appear in the formula. It can be modified later to include additional attributes.

Alarm history can be viewed either from Panic or PyTangoArchiving.widget.snap widgets.

The image shows two overlapping windows from a PyTango-based application. The background window is 'Alarm History Viewer' showing a table of alarm events. The foreground window is 'SnapSaver' showing details for a specific snapshot.

**Alarm History Viewer Table:**

Date	Alarm	Comment
2012-06-18 12:12:00	DHC_P30_P30_A_STATUS_WARNING	ALARM: DHC.P30.P30_A.VAL.STATUS goes OFF Bombas de agua en DHC CW en las bombas P30A de los intercambiadores E06 si cambia el estado a OFF.
2012-03-22 14:50:20	DHC_P30_P30_A_STATUS_WARNING	ALARM: DHC.P30.P30_A.VAL.STATUS goes OFF Bombas de agua en DHC CW en las bombas P30A de los intercambiadores E06 si cambia el estado a OFF.
2012-03-22 12:37:51	DHC_P30_P30_A_STATUS_WARNING	ALARM: DHC.P30.P30_A.VAL.STATUS goes OFF Bombas de agua en DHC CW en las bombas P30A de los intercambiadores E06 si cambia el estado a OFF.
2012-03-22 12:19:50	DHC_P30_P30_A...	

**SnapSaver Window Details:**

- Filter: Name
- Context: AL\_AFS\_STO\_LVL\_LOLO\_READ\_WARNING [
- Author: AlarmAPP
- Reason: ALARM
- Description: BUILDING/CT/SQLSERVER/AL\_AFS\_STO\_LVL\_LOLO\_READ == 1
- Snapshot: 2014-04-07 09:44:27 - ALARM: TOWER TANKS ALARM si en el deposito de Sar

**Snapshot 2 - Actual values Table:**

Attribute Name	RV1	WV1	RV2	WV2	diff1	diff2
building/ct/sqlserver/AL_AFS_STO_LVL_LOLO_READ	1.0	X	0.0	X	1.0	X

# Alarm Life Cycle

- Condition: CIRCE\_PRESSURE:BL24/VC/VGCT-01/P1>3e-5

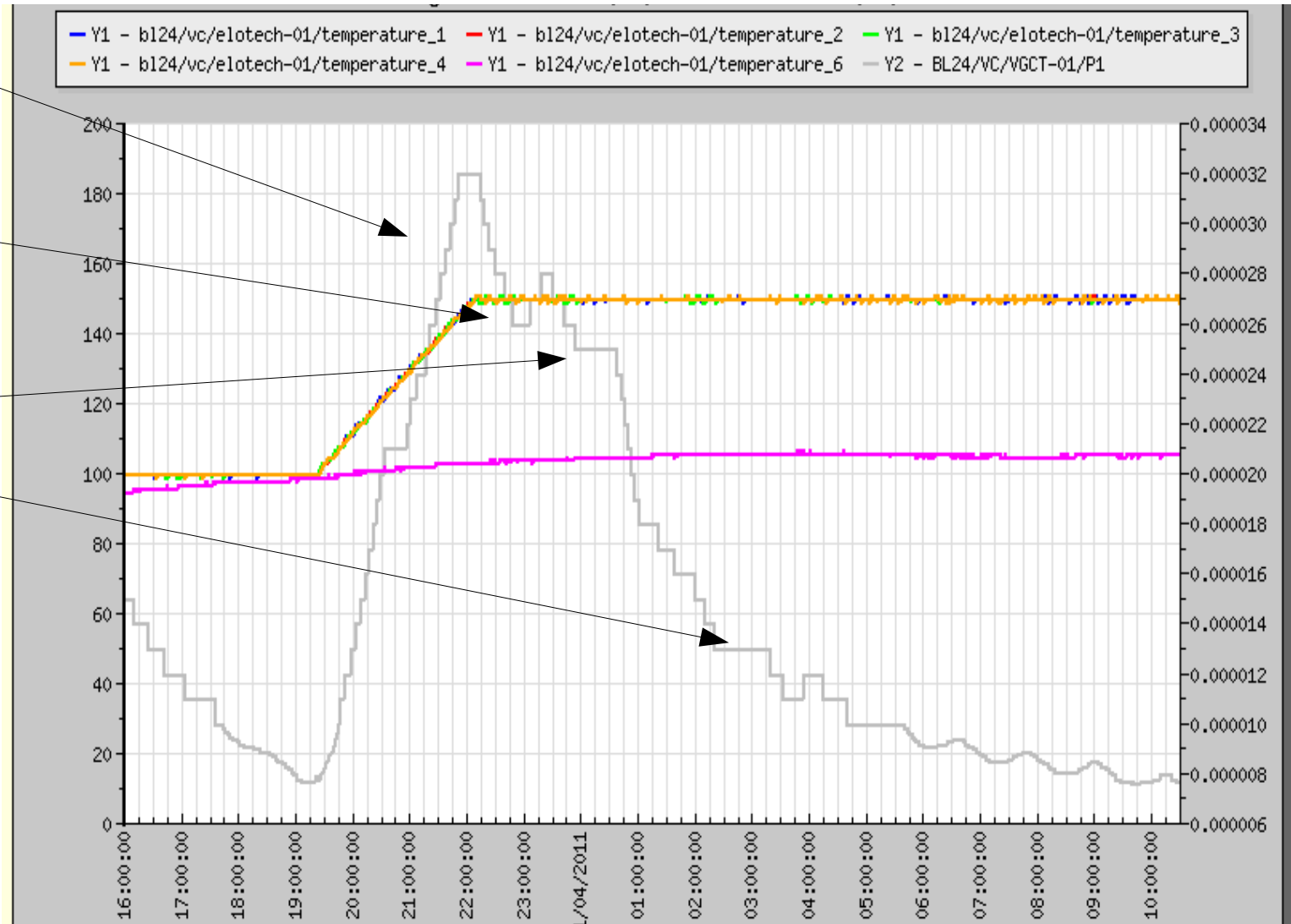
- Incidence  
- Notification  
- Archiving

- Recovery  
- Reminder

- Acknowledge

- AutoReset

- Actions?



Device Attribute Changer <@tiberius>

mach/alarm/ccds-01

	Attribute Name	Attribute Value
1	AlarmThreshold	3
2	AlertOnRecovery	false
3	AutoReset	3600.0
4	CreateNewContexts	False
5	Enabled	True
6	EvalTimeout	500
7	FlagFile	/tmp/alarm_ds.nagios
8	FromAddress	oncall
9	HtmlFolder	htmlreports
10	IgnoreExceptions	True
11	LogFile	/dev/null
12	LogLevel	INFO
13	MaxAlarmsPerDay	3
14	MaxMessagesPerAlarm	20
15	PollingPeriod	15.0
16	Reminder	0
17	RethrowAttribute	False
18	RethrowState	True
19	SMSConfig	controls@cells:cells.cells
20	StartupDelay	60
21	UseProcess	False
22	UseSnap	True
23	UseTaurus	False

Refresh

## PyAlarm Device Properties

Enabled may be True or a time (e.g. 120 seconds) to ignore alarms already enabled during startup.

PollingPeriod controls the frequency of update.  
 AlarmThreshold controls alarm triggering and .delta  
 AutoReset time will reset the alarm if condition recovers

Reminder/AlertOnRecover for extra notifications  
 FlagFile/LogFile/HtmlFolder controls local logging.

Alarm history controlled by CreateNewContexts/UseSnap  
 UseEvents replaced by UseTaurus; which is set to False by default

IgnoreExceptions/RethrowAttribute/State control whether exceptions should trigger alarm or not or be replaced by None.

UseProcess (python subprocess) still under development



# Alarm Life Cycle

- Condition: CIRCE\_PRESSURE:BL24/VC/VGCT-01/P1>3e-5

- Incidence  
- Notification  
- Archiving

- Recovery  
- Reminder

- Acknowledge

- AutoReset

- Actions?

Legend:

- Y1 - bl24/vc/elotech-01/temperature\_1
- Y1 - bl24/vc/elotech-01/temperature\_2
- Y1 - bl24/vc/elotech-01/temperature\_3
- Y1 - bl24/vc/elotech-01/temperature\_4
- Y1 - bl24/vc/elotech-01/temperature\_6
- Y2 - BL24/VC/VGCT-01/P1

Message Subject: BL24-CIRCE/CT/Alarms: CIRCE\_PRESSURE ALARM

From: controls@ivc00new.cells.es  
 To: vacuum@cells.es, oncall@cells.es, vperez@cells.es, epellegrin@cells.es, srubio@cells.es  
 Date: 2011-04-20 21:46

TAG: CIRCE\_PRESSURE  
 Alarm at Wed Apr 20 21:46:04 2011  
 Description: Chamber pressure exceeds limit  
 Formula: tb12401:10000/BL24/VC/VGCT-01/P1>3e-5 OR tb12401:10000/BL24/VC/VGCT-01/P2>3e-5

Values are:

tb12401:10000/bl24/vc/vgct-01/p2:	1.4e-06
tb12401:10000/bl24/vc/vgct-01/p1:	3.1e-05

Alarm receivers are:  
 vacuum@cells.es  
 oncall@cells.es  
 vperez@cells.es  
 epellegrin@cells.es  
 srubio@cells.es

Other Active Alarms are:  
 CIRCE\_PRESSURE:Wed Apr 20 21:46:04 2011:tb12401:10000/BL24/VC/VGCT-01/P1>3e-5 OR  
 tb12401:10000/BL24/VC/VGCT-01/P2>3e-5  
 CIRCE\_STARTED:Wed Apr 20 18:51:41 2011:any(t>50 for t in  
 tb12401:10000/bl24/vc/elotech-01/Temperature\_All)

[10:12:48] Transmission for account srubio@cells.es complete. No new messages.



# Declaring Receivers in Phonebook

Receivers can be set in AlarmReceivers property. When using tags previously declared in PyAlarm.Phonebook class, then each tag will be replaced by its mailing list..

```
%BEEP:ACTION(alarm:command,mach/alarm/beep/play,$DESCRIPTION)
```

```
%CONTROLROOM:operators@cells.es,SMS:+34646291497
```

```
%CTRLMV:oncall@cells.es,SMS:+34682793983
```

```
%CTRL3:oncall@cells.es,SMS:+34682793983
```

```
%PLCMV:plc@cells.es,SMS:+34638420276
```

# Receivers: Executing Actions and/or External Notifications

AlarmsList:

## **BL\_FE\_OPEN:**

bl/ct/plc-01/FE\_AUTO and  
**host:10000**/chan/ct/fe/value and  
bl/ct/plc-01/BL\_READY and not bl/ct/plc-01/fe\_open and  
not bl/ct/-plc-01/fe\_control\_disabled

AlarmsReceivers:

## **BL\_FE\_OPEN:**

**ACTION(alarm:attribute**,bl/ct/plc-01/OPEN\_FE,1),

**ACTION(alarm:command:test/notif/blmachine/popup**  
**,\$ALARM,\$DESCRIPTION,15)**

# Related Projects

## ProcessProfiler

- Provides CPU stats (cpuUsage, memUsage, ...)
- Can be used to trigger alarms (complementary to Nagios)

## FestivalDS

- Beeping (using OS)
- **Speech synthesizer** (using festival linux package)
- Beep+Speech
- **Pop-up** notifications, using libnotify
- FestivalDS must run with the same user that is managing the desktop

e.g. ACTION(alarm:command:test/notif/controls01/popup,\$ALARM,\$DESCRIPTION,15)

## Fandango.tango.\*

- Caseless **regular expression** parsing/sorting/matching (cached, offline when possible)  
e.g. get\_matching\_[device/attributes/labels/alias/properties/hosts/...](regexp)
- TangoEval (evaluation of alarm-like code; with user-macros like FIND or GROUP)
- Smart singletons: TangoCommand, CachedAttributeProxy, TangoedValues
- ProxiesDict when UseTaurus = False

# Alarm Pop-ups

The image shows a software interface for configuring an alarm. The main window, titled "ALARM: TEST\_STATE", contains the following fields and controls:

- Name:** TEST\_STATE
- Status:** ALARM (indicated by a red bar)
- Buttons:** Last Report, Reset
- Disabled:**
- Acknowledged:**
- Device:** test/alarms/alarms-test
- Severity:** WARNING
- Description:** Just trying to see what happens!!!
- Receivers:** ACTION(alarm:command:test/notif/pc148/popup,\$ALARM,\$DESCRIPTION,15)
- Formula:** test/test/alarms-test/State not in (OFF,UNKNOWN,FAULT,ALARM)
- Result:** (empty field)
- Buttons:** Edit, Preview

On the right side, a code editor shows a snippet of Python code:

```
f,req_type=None)  
self, attr)  
ttr)  
ns(self, attr)  
, attr)
```

Below the main window, a pop-up notification is displayed with the title "TEST\_STATE" and the message "Just trying to see what happens!!!". The notification includes a red "Panic" button icon.

The Windows taskbar at the bottom shows various application icons, including Blender, Firefox, and several instances of PyAlc.



# Summary

- **PyAlarm** running at ALBA since 2007
- Panic UI deployed in the machine since 2011
  
- Panic is both **Distributed and System-consistent**
- **DB optional** (may work as a no-db device server)
- Dynamic, formulas and **devices modifiable online**
- **Regular expression** matching
- Full Attribute value access (value, time, **quality, delta, exception**)
- Selective **exception management**
  
- **PyAlarm triggers** 4 different messages/actions:
  - ALARM, REMINDER, ACKNOWLEDGE, RESET
  
- It also uses 4 different **severities**:
  - INFO, WARNING, ERROR in DEBUG
  - DEBUG alarms don't trigger receivers
  
- Logging in local files and in the **Snapshotting Database**
- **Extended** by interaction with other Tango devices.



More info and SVN links at [www.tango-controls.org](http://www.tango-controls.org)