12th Int. Workshop on Emerging Technologies and Scientific Facilities Controls PCaPAC2018, Hsinchu, Taiwan JACoW Publishing
ISBN: 978-3-95450-200-4 doi:10.18429/JACoW-PCaPAC2018-WEP25

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

EVOLUTION AND CONVERGENCE OF ALARM SYSTEMS IN TANGO

S.Rubio-Manrique, G.Cuni, F.Fernandez, R.Monge
ALBA Synchrotron, Cerdanyola del Vallès, Barcelona, Spain
G.Scalamera, Elettra-Sincrotrone, Basovizza, Trieste, Italy

Abstract

The technology upgrade that represents Tango 9 has triggered the evolution of two of the most used Tango tools, the PANIC Alarm System and the HDB++ Archiving. This paper presents the status of the collaboration between Alba and Elettra Synchrotron sources for the convergence of its both alarms systems under the IEC62682 [1] standard, and the usage of HDB++ tools for logging and diagnostic of alarms. Relevant use cases from the user point of view has been added to the paper as a validation of the benefits of this control system evolution.

INTRODUCTION

Alarms in Control Systems

Alarm Systems have been a common part of control system toolkits for decades. In the Synchrotron community some of the most common tools are PANIC [2] and AlarmHandler [3] for Tango Control System [4], as well as BEAST Alarm System for EPICS.

PANIC and AlarmHandler systems have coexisted within the Tango community for years, but at some point new members of the community asked whether to choose one or the other for their specific domain. This question triggered an effort to compare both systems, extract the best features of them and explore how they could complement each other.

This effort required from us to redefine what an Alarm System was, and what it was expected to do.

What's an Alarm System?

SKA and Elettra institutions proposed to the Tango community to adopt a common terminology and behaviour based on an international norm, the IEC 62682:2014 "Management of Alarm Systems for the Process Industries"[5-6]. The norm states that:

- 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, which must have separate tools following its own regulation.
- The Alarm System is part of Operator Response, thus it's part of the HMI (including the non-graphical part of it).

These three statements clarified what our Alarm Systems were expected to do; providing a common ground to start the collaboration on merging both projects.

Alarms within the Tango Control System

It is needed to introduce several concepts on how alarms are developed within a Tango Control System. The following terms describe the object hierarchy:

- Tango Host or Database: the central database where all devices are registered and configuration stored.
- Device Server: Each independent software process distributed across the system, managing one or several Tango devices.
- Device: a Tango Device is an entity that can be typically identified to a hardware piece or software process (e.g. a vacuum pump, a PLC, a motor, a voice synthesizer). Each device exports to the control system its Attributes (process variables), Commands (actions), Properties (for configuration) and States.
- Attributes: Each of the process variables exported by a Tango Device. They support both synchronous or asynchronous reading and writing. At each attribute reading it exports its value, timestamp and quality.
- Attribute Quality: The attribute quality accompanies each value to express the process conditions. Quality can be VALID, INVALID, WARNING or ALARM and it can be set on runtime by a Tango user specifying which ranges of operation/warning/alarm will trigger a quality change.

Quality-based vs Formula-based Alarms

The most primitive scope of Alarm Systems in Tango just included the logging of those attributes qualities in ALARM. But this approach didn't apply when it was required the interaction between multiple devices.

The Tango Alarm System (AlarmHandler device server and its alarm database) was developed [3] to enable logical and arithmetical operations on attribute values, thus extending the alarm triggering from the simple matching of an attribute value within valid ranges.

PANIC [7] was developed by ALBA Synchrotron in 2007 [8] as a Python [9] alternative to the Tango Alarm System. It was based on similar principles but applying a distributed architecture (see Fig. 1) and trying to add annunciator features and more flexibility [10] in alarms declaration, allowing to execute python code for both the formula and the resulting action [11]. This enabled the usage of wildcards for attribute selection, and reusing the data from the alarm evaluation to generate rich-text emails or SMS messaging.

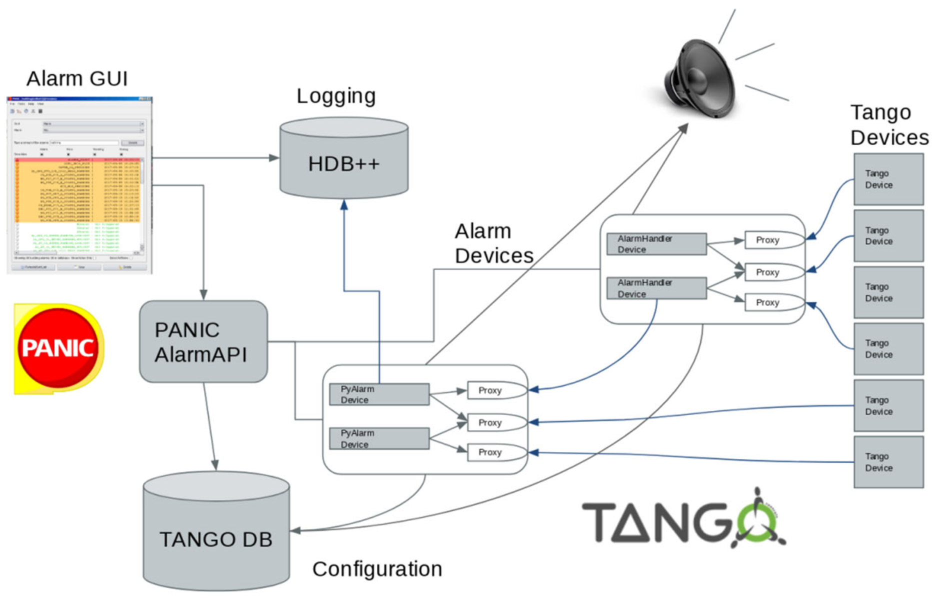

Figure 1: PANIC Architecture, showing alarm device servers, configuration and logging databases, and annunciators.

ALARM SYSTEMS CONVERGENCE

Existing Alarm Systems in Tango have been already described in previous papers [2-3], as well as the changes required [12] to adapt them to the IEC62682. This paper focus instead in the changes that allowed their convergence into a single toolkit.

Unifying the AlarmHander and PANIC device servers forced us to unify criteria on:

- Alarm Formula database
- Alarm attributes
- Classification of alarms
- Annunciators specification and triggering
- Priorities
- Exporting attributes to HMI
- Alarm Summaries

The criteria for convergence have been triggered by the chosen norm (that fixed terminology and state conditions), the Tango control system architecture (centralized database / distributed devices) and the possibilities to expand the HMI to the web (choosing JSON as the default data format for Alarm Summaries).

Intended to reduce the number of different tools required for having a functional alarm system, the Alarm database has been dropped to use Tango Properties instead. Keeping the alarm formulas and configuration in the Tango database helps to unify the quality-based and formula-based alarm approaches, thus storing the configuration for both types of alarms in the same database.

Following the same principle, it has been discarded the idea to maintain a separate Alarm logging, thus reusing the new HDB++ archiving for Tango [13].

HDB++ archiving allows to record alarms on state change, and at the same time records all the events received by the attributes involved in the formula (including both value and quality changes). Thus, alarm changes and attribute changes can be queried and represented from the same database using already existing tools.

ALARMS IMPLEMENTATION

Reactive Alarms vs Polled Alarms

The PANIC API is currently connecting to two types of device servers, PyAlarm, developed in Python by ALBA Synchrotron, and AlarmHandler, developed in C++ by Elettra Sincrotrone.

Both device servers acquire lists of formulas from the Tango Database Properties, connect to the attributes appearing on the formulas and evaluate the results on each attribute value change; triggering actions when required by an alarm state change. Devices differ on how they connect to attributes and react to value change.

PyAlarm is polled-based, so it means that attributes are read periodically, updating the formula result at each attribute reading. Attribute changes are cached and alarm is not triggered until the result is True for a number of iterations (the AlarmThreshold property). With a threshold of 1, response time can be as short as 100 ms; but still this minimal latency time is needed. In exchange, complex formulas are enabled and the polling buffer allows to use the average, the max peaks or the delta change of the last N values acquired.

AlarmHandler instead is event-based. It means that the device is just waiting for events sent from the Tango Control System, reacting immediately when a change event is received. In this case, latency is minimal, and actions are executed immediately. Alarm formula parsing is done in C++, and it is restricted to basic arithmetical and logical operations between attribute values.

As demonstrated in the field, both systems are complementary, as the AlarmHandler can be dedicated to fast reaction on critical conditions while PyAlarm flexibility

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.

provides its best usability on interacting with multiple devices or evaluating attribute evolution in time.

Alarm System Scalability

Another factor that triggered convergence of Alarm Systems in Tango has been the need to increase system scalability and performance for the new SKA project.

Alarms Systems can be scaled using different approaches. PANIC allows scaling by exporting each evaluated alarm as a new attribute. Thus, new alarms can be written using the result of the previous ones already evaluated. It allows having a detailed alarm for each subsystem of a sector, and then summarize all values in a single sector alarm. As example, vacuum alarms will contribute to the alarm state of a sector, and at the same time are grouped in a vacuum alarm for all sectors.

Alarm summaries can be done at client level (Alarm View) or at device server level (Alarm Group). When created at alarm level, it will behave as any other alarm, triggering actions and exporting a new attribute, so the system can be scaled to the next level. They can be accessed even from other Tango Control Systems, so the hierarchy can be expanded indefinitely. Propagation times at each level can be tuned using the AlarmThreshold and PollingPeriod properties.

Alarm Annunciators

Besides its initial features (email, SMS, logging), the current version of PANIC allows now to trigger any kind of Tango command or python code script and provides the capability of reusing alarm data and values in the execution of these commands. As example, it allows to send alarm description to speakers, include attribute values in an email or html report, move a motor a fixed number of steps depending on a calculation, etc.

It also expands the logging mechanisms. HDB++ and the Tango DB are used to keep history of alarm and attribute changes and its configuration; but via external commands other messaging or logging systems can be easily integrated. It allowed PANIC to interact with some popular applications like Telegram or Kibana and generate web reports based on JSON files [14].

CONCLUSIONS AND FUTURE

PANIC is a key tool for the daily operation of ALBA Synchrotron Accelerators and Beamlines, it is the first diagnostic tool used in order to check any incidence. It offers an overview of currently triggered alarms, allowing a full comprehension of the problem. Its second potential resides in the customization of the alarms applications that allows to suit the specific needs of each user group and, in its integration with Taurus [15-16], eases a fast response and troubleshooting in a user-friendly environment.

Having a common interface with Elettra's AlarmHandler allow to expand PANIC functionality to a new level, as now two complementary Alarm engines are provided and scalability can be achieved combining performance and flexibility at each level.

PANIC Use Cases

This are some practical examples of PANIC use cases that have been enabled by the changes introduced in the latest releases:

Protection of infrared mirrors; in this case an alarm was required to, in case of a fast increase of temperature, to be able to move a mirror outside of the vacuum chamber in less than 200 ms. For this application alarms on delta change, response to events and precise motor movement execution were required.

Injection Permits: In the last upgrades of our Synchrotron facility, the number of injection modes are increasing as well as the complexity of the rules that allow to inject in the storage ring. PANIC has been used as an easy tool to compile permission rules and convert it into attributes that can be used as enable/disable of different operations.

Vacuum Systems: it is very important the continuous monitoring of the pressure and equipment state, but besides this, there are many variables belonging to other sub-systems which could affect the vacuum status. PANIC allows to extend the views and notifications of each alarm to include additional information or parameters that help to diagnose the problem. It also allows to crosscheck temperatures against different ranges depending on the current operation status (injection, maintenance, bakeout); checking not only the value but its proper behaviour.

Testing and Deployment

Testing of PANIC is critical in to ensure its reliability and scalability, guaranteeing that any modification in the evaluation engine does not break the compatibility with previous versions. A test suite has been developed [17] and its currently expanded with the help of tango-simlib project. Testing will be introduced in our continuous integration/deployment cycle based on Debian packages.

ACKNOWLEDGEMENT

The latest release of PANIC contains contributions from many people: MaxIV, ESRF, ALBA, Elettra and Solaris institutes as well as some private companies like S2Innovation, IK and TCS.

The work of validating and suffering each new release is shared with our users, the Operation and Vacuum groups at ALBA. They contributed with their experience to refine and improve alarm formulas and syntax.

REFERENCES

[1] *Management of alarms systems for the process industries,* IEC-62682, IEC, 2014.

[2] S. Rubio-Manrique et al., "PANIC, A Suite for Visualization, Logging and Notification of Incidents.", in *Proc. PcaPAC'14*, Karlsruhe, Germany, Oct. 2014, paper FCO206.

[3] L. Pivetta, "Development of the Tango Alarm System", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005, paper WE3b.1-70.

[4] TANGO, `http://www.tango-controls.org`

[5] *Engineering Equipment and Material,* Users'Association (EEMUA) issued publication, 191, University of Manchester, United Kingdom, 1999.

[6] M. Tennant, "Implementing Alarm Management Per the ANSI/ISA-18.2 Standard", *Control Engineering*, September 2013.

[7] PANIC, https://github.com/tango-controls/panic

[8] S. Rubio-Manrique et al. "Extending Alarm Handling in Tango", in *Proc. ICALEPCS'11*, Grenoble, France, Oct. 2011, paper MOMMU001.

[9] D. Fernández et al. "Alba, a Tango based Control System in Python", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper WEPMU005.

[10] S. Rubio et al., "Dynamic Attributes and other functional flexibilities of PyTango", in *Proc. ICALEPCS'09*, Kobe, Japan, Oct. 2009, paper THP079.

[11] http://www.pythonhosted.org/panic/recipes.html

[12] G. Scalamera, L.Pivetta, S.Rubio-Manrique, "New developments for the Tango Alarm System", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper TUPHA165.

[13] L. Pivetta et al., "New developments for the HDB++ Tango Archiving System", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper TUPHA 166.

[14] M. Broseta, D. Roldan, S. Rubio, A. Burgos, G. Cuni, "A web-based report tool for Tango Control Systems via websockets", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper TUPHA 173.

[15] S. Rubio et al., "Unifying all Tango Services in a single Control Application", in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015. paper WEPGF148.

[16] Taurus Library, http://www.taurus-scada.org

[17] S. Rubio-Manrique et al., "Reproduce Anything, Anywhere: A Generic Simulation Suite for Tango Control Systems", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, paper TUDPL01.

Content from this work may be used under the terms of the CC BY 3.0 licence (© 2018). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI.