DEVELOPMENT OF THE TANGO ALARM SYSTEM

L. Pivetta
Sincrotrone Trieste, Trieste, Italy

ABSTRACT

Alarms are a key part of every control system. They provide asynchronous notification of status changes and let the machine operator be warned of possible problems related to the controlled devices. The development of a flexible, highly configurable alarm system integrated in the TANGO [1] structure is underway at ELETTRA.

INTRODUCTION

An alarm is an asynchronous notification that some event has happened or that a given state has been reached in the plant. The alarm system is in charge of providing a fast and effective method to manage this information. It should be flexible enough to allow some processing to be made, for example to compare a variable with a predefined value or to evaluate some status and, according to the result, decide whether a condition is met or not, namely the alarm is active or not. The most effective way to gather all of the necessary input information from the control system is to exploit the event-driven communication paradigm available within TANGO.

THE TANGO CONTROL SYSTEM

TANGO is an object oriented control system framework based on CORBA [2]. Designed having in mind system integration as the main goal, TANGO uses a subset of the CORBA middleware specifically targeted for control systems and introduces the so called "Device" paradigm [3]. All the control objects are derived from the "Device" type, thus sharing the same basic interface and functionalities. Among these there are device commands, attributes and properties. Commands correspond to actions made by the control system on the device. Attributes are device variables that can be associated to physical quantities and are normally used to perform readings from the field. Device default settings and permanent data are stored into specific variables called properties.

Each device server, the actual implementation of control objects, will then deal with the specificity of the interface and the behaviour of the controlled device.

The alarm system has been developed as a special TANGO device server, named the Alarm Collector.

REQUIREMENTS

A distributed control system is usually made of a number of computers. A possible approach could be to have a distributed alarm system; each computer in the control system could be in charge of monitoring and evaluating the alarm rules associated to its own controlled devices. This approach has the advantage of distributing the load, but lacks in flexibility; evaluating an alarm rule whose input values come from multiple computers may be tricky. Instead, a centralized system allows the management of complex rules involving several plant variables in a simpler manner. The possibility of using alarm rules defined by means of formulas is desirable.

In summary, the requirements for the new TANGO alarm system are:

- easily configurable, even at runtime
- alarm rules based on values from multiple hosts
- support for complex rules
- logical and binary operators
- basic mathematical functions
- consistency between multiple clients

ALARM RULES

At the basis of the whole alarm system are the rules that specify each alarm condition. Each rule is made of three distinct fields: the alarm name, the alarm formula and an optional message. The alarm name is a self-explaining unique label for each configured alarm. The second field is used to store the alarm formula in the form of a string made of identifiers, numbers, binary and logical operators, as well as simple mathematical functions. Each alarm rule also allows for storing an optional text field, containing the alarm explanation or a specific message for the machine operator. As an example, the following alarm rule will generate an alarm when the power supply of an insertion device correction coil is turned off:

sr/pscid/s1.1/off   ({sr/pscid/s1.1/stat} & 0x40)   "Correction coil power supply OFF"

sr/pscid/s1.1/off is the alarm name and *{sr/pscid/s1.1/stat}* is the reading of the power supply status; the braces contain the values to be acquired from the plant, i.e. the attributes read from the TANGO device servers.

Available binary and logical operators include &, |, ^, ~, *, +, -, (), <<, >>, <=, >=, <, >, !=, &&, ||, having the same meaning as in C programming language. Round brackets allow for writing more complex formulas, involving multiple arbitrarily combined conditions. An example is:

sr/pscid/s1.1/off   (({sr/pscid/s1.1/stat} & 0x40) && ({sr/carid/s1.1/stat} & 0x100))

or

sr/psch/s1.1/highthr   (({sr/psch/s1.1/stat} & 0x80) && ({sr/psch/s1.1/curr} > 15.0))

The first triggers the alarm when a correction coil power supply is turned off and the insertion device feed-forward orbit correction loop is enabled; the second checks if both a corrector magnet power supply is switched on and the supplied current is higher than 15 A.

The alarm rules are stored into the TANGO database as device attribute properties of the Alarm Collector TANGO device server. The database maintains a consistent configuration for the whole alarm system.

ALARM MESSAGES

As a result of the formula evaluation the alarm status could assume two values, ALARM or NORMAL, stating whether the alarm is active or not. When an alarm is active the Alarm Collector composes a formatted text string, the alarm message, to be sent to the client.

The alarm message contains the time stamp, the alarm identifier (or alarm name), the alarm status, the acknowledge flag and the optional text message. The acknowledge flag shows whether an alarm has been received by at least one visualization client and acknowledged by the machine operator, who should realize the meaning of the alarm and take the necessary actions. The acknowledge flag could have the values ACK or NOT_ACK, with obvious meaning.

Table 1 summarizes the available combinations of status and acknowledge flag of a given alarm.

Status	*Acknowledge flag*
ALARM	NOT_ACK
ALARM	ACK
NORMAL	NOT_ACK

Table 1. Alarm status and acknowledge flag matching.

Acknowledging an alarm in NORMAL status will remove it from the active alarms table. In the same manner, an alarm that has already been acknowledged will disappear from the list when switching to NORMAL.

THE ALARM COLLECTOR

Overview

As already noted, the Alarm Collector is a special TANGO device server written in C++ language, which makes use of the TANGO API. It is based on a double client/server architecture: acting as a client the Alarm Collector gathers the necessary input values from the involved TANGO devices, as a server it provides alarm notification to any client interested in receiving alarms. It relies on the TANGO event system to collect input values as well as to provide alarm notifications. The main interactions between the Alarm Collector, the TANGO framework and a number of different alarm clients, along with the mechanisms involved, are shown in Figure 1.

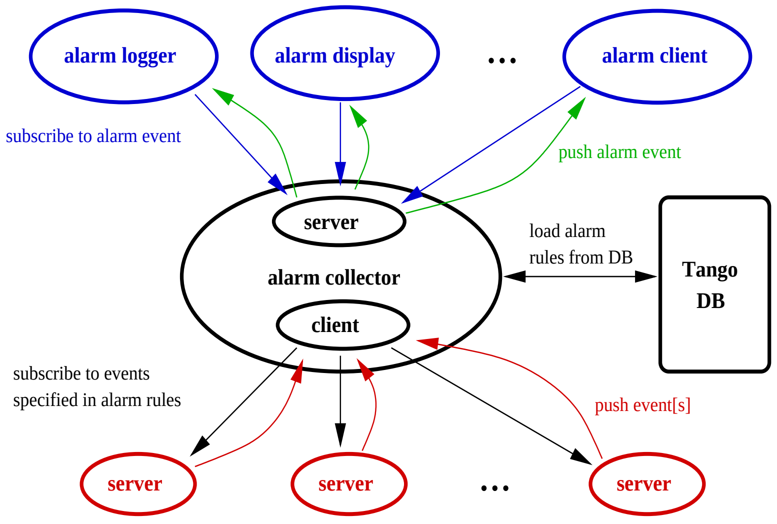


Figure 1. The Alarm Collector layout

The event system is the TANGO implementation of the “event-driven” communication paradigm. In this paradigm, instead of polling the server for updating a value, the client registers its interest in the value just once and the server provides the current value by sending an “event” when a given condition is met. Although the “change” event is the most suitable for alarm purposes, other types of events are implemented in TANGO: change, quality, periodic, archive and user event. The “change” event is generated when the attribute changes significantly. Absolute and relative “delta” thresholds can be specified for analog attributes. In addition to writing the alarm rules, setting up the alarm system clearly involves choosing the proper attribute event thresholds, having in mind the peculiarity of the real device and its physical variables.

During normal operation the Alarm Collector will be idle, waiting for events. As soon as en event is received, the server evaluates the alarm formulas containing the attribute corresponding to the received event and updates the alarm value. In case the alarm status changes an event notification is eventually sent to every interested client.

Internals

A block diagram describing the operation of the Alarm Collector is shown in Figure 2. At startup, the Alarm Collector loads from the database the configured alarm rules, builds the required data structures and, for each attribute contained into the rules, subscribes to the change event. The *event handler routine* is in charge of collecting the events and storing them into a FIFO (First In First Out) queue. Access to the event FIFO is regulated by a mutex, whereas a condition variable allows for an efficient queue management. One dedicated thread, named *alarm evaluation thread*, normally sleeping on the condition variable, is then spawn to perform the alarm formula evaluation; when the FIFO is not empty it is woken up and, for each event in the FIFO, evaluates all the alarm rules containing the related attribute. The results, with all the relevant details such as the event time stamp, are then stored into the server internal *alarm table*.

The Alarm Collector tries to ensure alarm status consistency in case of server shutdown by saving the internal status into the database. A TANGO device property is timely created by the server before shutting down and checked at startup in order to address possible alarm status changes that may occur during server shutdown.

The server also keeps an internal table, called *active alarm table*, of all the active alarms, i.e. those with an ALARM or NOT_ACK status. The *alarm table* is periodically compared to the *active alarm table* to build a list of changed alarms which is defined as a TANGO device attribute. Client programs interested in receiving alarms can subscribe to the change event related to this attribute.

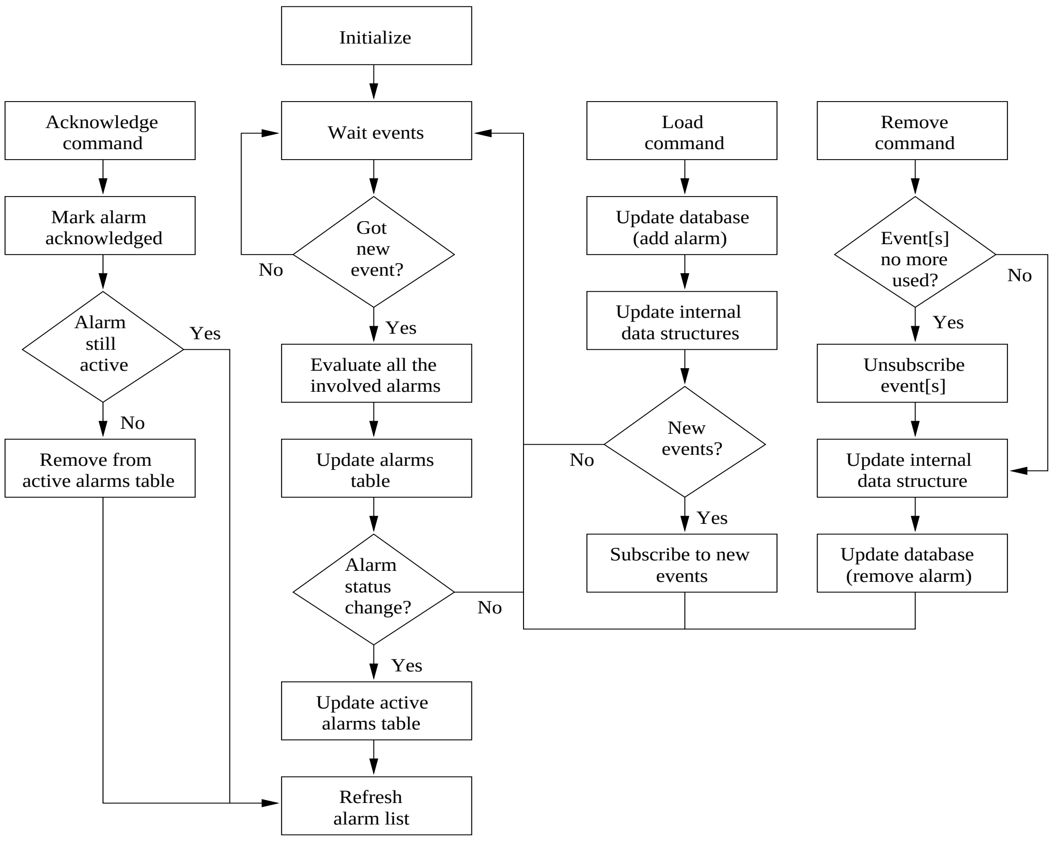


Figure 2. Alarm Collector block diagram

The Alarm Collector server interface currently exports five methods; two attributes and three commands. The first attribute, named "*alarm*", is a list of strings, each containing an alarm message; the Alarm Collector populates this list with the results of the alarm rules evaluation. The second attribute, named "*show*", allows to retrieve the complete list of all the alarms configured in the device server. The "*acknowledge*" command is used to mark an alarm as "seen by the operator"; this command will also update the *active alarm table*, removing those that are in NORMAL and ACK status. "*load*" and "*remove*" commands can be used to add and remove an alarm contained in the alarm table, updating at the same time the database. Removing an alarm is only allowed when it is in NORMAL and ACK status. When removing an alarm the Alarm Collector will eventually unsubscribe from attribute events no longer used.

Through the mentioned methods any TANGO client can configure and monitor the alarm server, including the "swiss army knife" Jive (a graphical tool used to configure the TANGO database) and

Python scripts. A graphical interface for alarm configuration is foreseen.

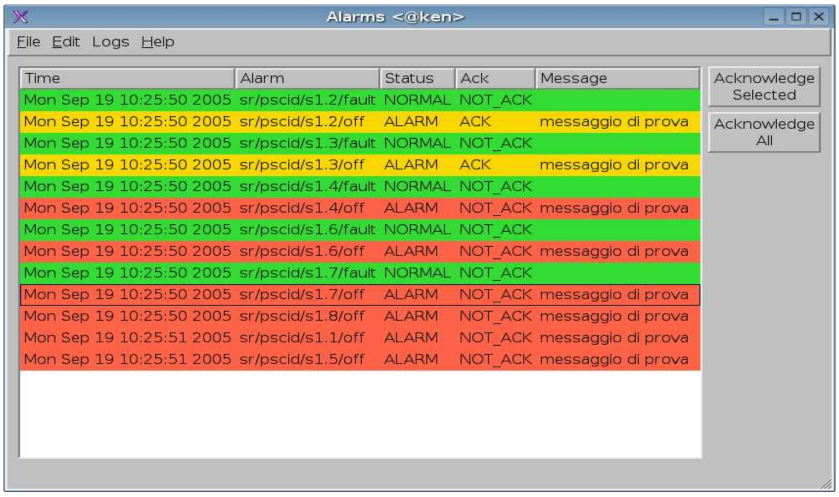
Scanner and parser

A lexical scanner, generated with GNU Flex [4], is used to initialize the server internal data structures at startup. Each alarm rule is split into three known tokens: *alarm*, *formula* and *message*. Then, in order to evaluate the alarm formulas during normal operation, another lexical scanner together with a parser, generated with GNU Bison [5], is used. This scanner is able to split each alarm formula into its basic tokens: operators, numbers and attributes. Three types of numbers are foreseen: decimal, hexadecimal and floating point. Each token is assigned a unique value that the scanner returns to the parser while scanning the alarm formula. With this information the parser can apply to the operands the rules specified in its grammar and finally evaluate the whole formula. The result is a boolean flag that can assume the ALARM or NORMAL values, showing whether the alarm under evaluation is currently active or not.

ALARM CLIENTS

Alarm presentation panel

A simple client application has been developed for testing the Alarm Collector device server. The visualization client, written in Python using the Qt libraries [6], subscribes to the "*alarm*" event. The alarm server will send an updated list of alarms each time the status of at least one of them changes. The visualization client uses colours to show alarm messages according to their status: red for ALARM/NOT_ACK status, yellow for ALARM/ACK status and green for NORMAL/NOT_ACK status. The panel also provides two push buttons to acknowledge either the selected alarms or the whole alarm list.



Time	Alarm	Status	Ack	Message
Mon Sep 19 10:25:50 2005	sr/pscid/s1.2/fault	NORMAL	NOT_ACK	
Mon Sep 19 10:25:50 2005	sr/pscid/s1.2/off	ALARM	ACK	messaggio di prova
Mon Sep 19 10:25:50 2005	sr/pscid/s1.3/fault	NORMAL	NOT_ACK	
Mon Sep 19 10:25:50 2005	sr/pscid/s1.3/off	ALARM	ACK	messaggio di prova
Mon Sep 19 10:25:50 2005	sr/pscid/s1.4/fault	NORMAL	NOT_ACK	
Mon Sep 19 10:25:50 2005	sr/pscid/s1.4/off	ALARM	NOT_ACK	messaggio di prova
Mon Sep 19 10:25:50 2005	sr/pscid/s1.6/fault	NORMAL	NOT_ACK	
Mon Sep 19 10:25:50 2005	sr/pscid/s1.6/off	ALARM	NOT_ACK	messaggio di prova
Mon Sep 19 10:25:50 2005	sr/pscid/s1.7/fault	NORMAL	NOT_ACK	
Mon Sep 19 10:25:50 2005	sr/pscid/s1.7/off	ALARM	NOT_ACK	messaggio di prova
Mon Sep 19 10:25:50 2005	sr/pscid/s1.8/off	ALARM	NOT_ACK	messaggio di prova
Mon Sep 19 10:25:51 2005	sr/pscid/s1.1/off	ALARM	NOT_ACK	messaggio di prova
Mon Sep 19 10:25:51 2005	sr/pscid/s1.5/off	ALARM	NOT_ACK	messaggio di prova

Figure 3. Alarm visualization panel

Alarm logger

The "alarm logger" is in charge of collecting the alarm messages and save them either in a file or in the database. A graphical tool will be developed to browse the archived alarm messages.

FUTURE IMPROVEMENTS

In order to improve the integration into the TANGO framework and to extend the support for simpler alarm configurations, some enhancements are foreseen. In particular, the support for the TANGO "quality" event, in addition to the existing "change" event, is planned. Moreover the introduction of a "*type*" or "*family*" field into the alarm messages could allow a sort of hierarchical visualization of the alarms.

CONCLUSIONS

This paper has presented the requirements a flexible alarm system should fulfil to be deployed into an accelerator control system. The main aspects of its implementation have been reviewed and the proposed solutions explained. The alarm system has been successfully tested at ELETTRA and will serve the new booster control system [7].

ACKNOWLEDGMENTS

ELETTRA Machine Controls Group, the TANGO Team at ESRF and Soleil.

REFERENCES

[1] The TANGO Team, "The TANGO control system manual", Version 5.1, 13th January 2005, http://www.esrf.fr/Infrastructure/Computing/tango/documentation
[2] OMG home page, http://www.omg.org
[3] A. Goetz et al., “TANGO – An Object Oriented Control System Based on CORBA”, ICALEPCS'99, Trieste, October 1999.
[4] “Flex, a fast scanner generator”, version 2.5, Free Software Foundation, March 1995.
[5] “Bison user manual”, version 1.875d, Free Software Foundation, 2004.
[6] Trolltech home page, http://www.trolltech.com
[7] M. Lonza et al., “The Control System of the ELETTRA Booster Injector”, these proceedings.