THE BEST EVER ALARM SYSTEM TOOLKIT*

Kay Kasemir, Xihui Chen, Ekaterina Danilova, ORNL, Oak Ridge, TN 37831, U.S.A.

Abstract

Learning from our experience with the Experimental Physics and Industrial Control System (EPICS) alarm handler (ALH) as well as a functionally similar approach based on script-generated operator screens, we developed the Best Ever Alarm System Toolkit (BEAST). It is based on Java and Eclipse on the Control System Studio (CSS) platform, using a relational database (RDB) to store the configuration and to log actions. It employs the Java Message Service (JMS) for communication between the modular pieces of the toolkit, which include an Alarm Server to maintain the current alarm state, an arbitrary number of Alarm Client user interfaces (GUI), and tools to annunciate alarms or log alarm related actions. Web reports allow us to monitor the alarm system performance and spot deficiencies in the alarm configuration. The Alarm Client GUI not only gives the end users various ways to view alarms in tree and table format, but also makes it easy to access guidance information, related operator displays and other CSS tools. It also allows the alarm configuration to be modified online from the GUI. Coupled with a good "alarm philosophy" on how to provide useful alarms, we can finally improve the configuration to achieve an effective alarm system.

INTRODUCTION

Before the using of the Best Ever Alarm System Toolkit (BEAST), it was very hard to manage and reduce the hundreds of alarms daily generated at the Spallation Neutron Source (SNS), an installation with more than 300000 Process Variables (PVs). The SNS controls group had tried a number of different approaches to alarm handling, starting with the EPICS alarm handler (ALH) [1], a Unix/X11 tool whose layout is fixed to a tree view of alarms and a legend. The tree view requires several mouse clicks to reach the actual alarm(s). ALH lacks ways to handle multiple selected alarms at the same time. There are no interfaces to other control system tools. Another attempt was a soft-IOC-based alarm handler [2], which basically creates EPICS operator displays from ALH configuration files together with soft-IOCs that implement the ALH logic. This provided ALH functionality within the main SNS operator display, but still required navigation down many levels of subscreens, each again in a fixed layout, to determine the actual alarms. In both systems, configuration changes were hard. Starting with a less than perfect alarm system configuration, using tools which made it hard to improve soon led to end user frustration with the alarm system.

The tools presented in this paper not only give end users various graphical ways to view or handle current alarms, including access to guidance information on how to handle a specific alarm, related operator displays or other control system tools, they also allows us to monitor the alarm system performance, for example to determine which alarms trigger most often. Most important they allow online configuration changes from a graphical user interface so we can easily improve the configuration. This is coupled with an overall "philosophy" on how to provide useful alarms [3].

ARCHITECTURE

Inspired by the DESY alarm management system [4] and sharing many of the same CSS components [5,6], the BEAST was designed in a Client/Server architecture with tools for annunciation, logging and web report generation (see Fig.1). The modular design improved its flexibility, stability and reusability significantly at a small sacrifice of additional work on installation.

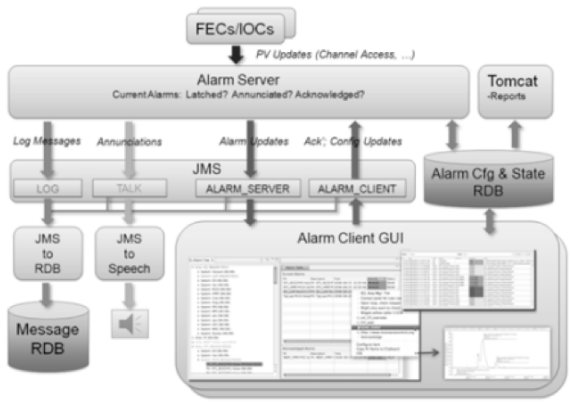

Figure 1: The system architecture of BEAST.

At the core of the BEAST is the Alarm Server. It reads the alarm configuration from the RDB, connects to all the requested PVs, monitors their state changes and generates alarms, handling acknowledgement, annunciation, latching, and some amount of filtering. It allows several GUI clients to connect simultaneously. The Alarm Server and Client GUI will be discussed later in detail.

The BEAST uses a relational database to store the configuration and to log actions. The configuration includes information for the Alarm Server (what PVs to monitor, whether to latch or annunciate alarms) as well as the Alarm Client GUI (user guidance on an alarm, related display links). The current states of all alarms are also stored in the configuration database, supporting both MySQL and Oracle.

JMS, specifically Apache ActiveMQ, is employed for communication between the modular pieces of the toolkit, using JMS topics with distinct purposes. The

* SNS is managed by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy

ALARM_SERVER topic is used by the Alarm Server to publish alarm updates and periodic idle message. The ALARM_CLIENT topic allows client GUIs to notify the alarm server about alarm acknowledgements and configuration updates. The TALK topic is dedicated to annunciation messages.

The tools for annunciation and logging are separate from the Alarm Server and GUI clients. JMS2Speech annunciates messages from the TALK topic. JMS2RDB can listen to all JMS topics, writing received messages to an RDB message log. They are otherwise generic JMS clients, independent from the message origin, so both of them could be reused for other JMS based applications.

Finally, web reports based on Java Server Pages (JSP) are provided to monitor the alarm system performance and spot deficiencies in the alarm configuration by reading and analyzing the data in configuration and logging databases. They play a very important role in improving the alarm configuration at the SNS.

ALARM SERVER

The Alarm Server is the central place for maintaining alarm states based on the configuration from the RDB. It connects to all requested PVs, monitors the severity of each PV and then computes the alarm state of PVs based on the alarm logic configuration for each PV, which includes enablement, latching, delay, count and a filter expression. The alarm state can also change via operator acknowledgement performed on a client GUI.

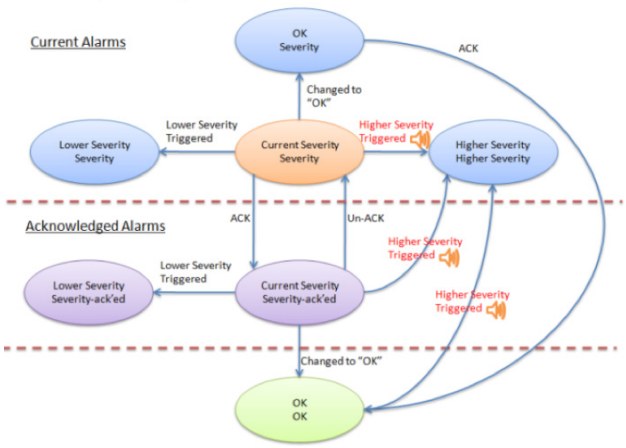

Figure 2: The alarm state diagram for a latched PV.

When a PV is configured to "latch", the alarm server remembers the highest alarm severity of the PV until it is manually acknowledged. For each alarm, it not only maintains the current alarm severity (Current Severity) which is directly read from PV, but also the remembered severity (Severity). "Current severity" might differ from "Severity" when the underlying alarm PV already recovered but the alarm server was configured to "latch" the alarm until operator acknowledgement. Figure 2 shows alarm state transitions in response to various events.

The “Delay” and “Count” configuration can help to reduce the amount of “nuisance” alarms from noisy PVs. By default, the alarm server will react as soon as it notices a non-OK alarm severity. When adding a delay greater than zero seconds, it will only react when the alarm severity remains for at least this time. When an additional alarm count greater than zero is specified, it will react to alarms that either outlast the delay, or happen at least “count” times within the delay. In practice, however, it is always best to cure alarm noise problems at the source through adequate dead-bands or smoothing.

A filter expression can automatically enable or disable alarms based on the value of other PVs, allowing mode-based alarming, though this is also preferably implemented on the front-end computer that provides the alarm PV.

When a PV is configured to “annunciate”, the alarm server will send the PV's description to the annunciation system (JMS2Speech) whenever the alarm severity of the PV rises.

ALARM CLIENT GUI

The Alarm Client GUI is integrated into Control System Studio (CSS) [5, 6] as three views: Alarm Tree, Alarm Table and Message History. The Client GUI reads the initial alarm configuration with guidance messages and related display links from the RDB. It connects to the Alarm Server via JMS to read alarm updates and to submit acknowledge requests or configuration updates. It shows alarm handling guidance or opens related displays on user request.

Alarm Table

The alarm table provides a tabular view of currently active alarms. It only shows alarms that were actually triggered. It is split into two parts for displaying current alarms (unacknowledged alarms) and acknowledged alarms respectively. The alarms can be sorted by any one of the columns: PV name, Description, Time, Current Severity, Severity, Status or Value.

From the context menu or toolbar, the user can acknowledge or un-acknowledge one or multiple selected alarms in the table.

Alarm Tree

The alarm configuration is hierarchically arranged by Area, System, optional Subsystems and finally PVs. Alarm tree components “inherit” the guidance and display info of their parent items. The Alarm Tree allows access to this hierarchy in a tree-like structure, which by default includes all configured alarms, active or not. Optionally, one can also choose to show only active alarms. Active alarms can be acknowledged as in the table view.

The alarm state of each tree item is reflected by a dynamically colored icon and a three-word annotation (Current PV Severity/Alarm Severity/Alarm Status). The alarm state of an area or system summarizes the alarm states of its children.

From the context menu of an alarm tree item one can easily configure, add, move or remove a component. The

Alarm Tree is synchronized with the configuration in RDB. When a user changes the configuration in an Alarm Tree, the configuration in the RDB is updated and vice versa. The configuration update will take effect immediately on both the Alarm Server and Client GUIs without a restart. The simplicity of online configuration modifications make it possible to finally improve the configuration to provide useful alarms and have guidance for every alarm based on a good alarm "philosophy" [3].

Message History

The message history is a generic tabular browser for the CSS message log. It reads logging data from the message RDB upon requests. In the BEAST, it is used to view the history of alarms and actions.

Interoperability

From the context menu of items in the Alarm Table or Tree it is easy to access guidance information, related operator displays and web pages, an electronic logbook or other CSS tools such as Probe, Data Browser, EPICS PV Tree etc. The guidance information of a PV can instruct operators on how to deal with an alarm. Related operator displays and web pages can help to diagnose the problem or perform corresponding operations. Operators can also submit information about an alarm event with comments to an electronic logbook. This integration with other CSS tools is one of the inherent advantages of the CSS framework. It helps users to access other PV-related tools simply from the context menu of an alarm. Figure 3 shows the three steps to plot archived data for an alarm PV in the Data Browser.

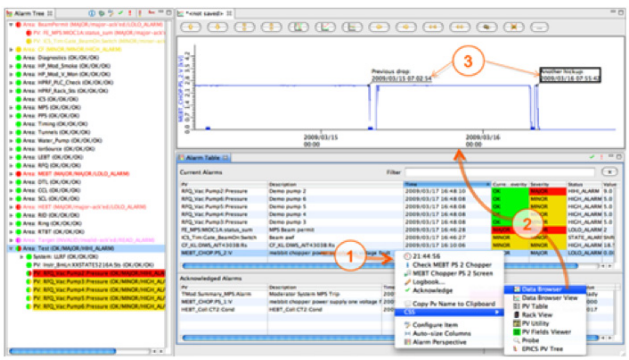

Figure 3: Interoperability with other CSS tools.

Security

The CSS authentication and authorization strategy was applied to BEAST to restrict privileges for alarm acknowledgement and configuration updates. Only authorized users are able to login and perform protected operations. An LDAP server is used to maintain information on users and their privileges.

WEB REPORTS

Web reports (Fig.4) utilize alarm data automatically saved in the RDB with JSP running on Tomcat technology. Dynamically created charts and tables are based on user choice of time period, PV name or pattern, and severity or current severity. The reports are accessed either immediately, through one button click, or through interactive charts. The charts present alarm statistics, and allow users to drill down for more information, as from a "Totals per Day" chart to a chart showing the "Top" 10 PVs with the highest number of alarms with cumulative percentages, to detail reports which contain not only statistics on minimum, maximum, average, most frequent and extreme alarm durations, alarms by severity, and 10 minute slices on a time line chart, but also detailed information about every alarm state change.

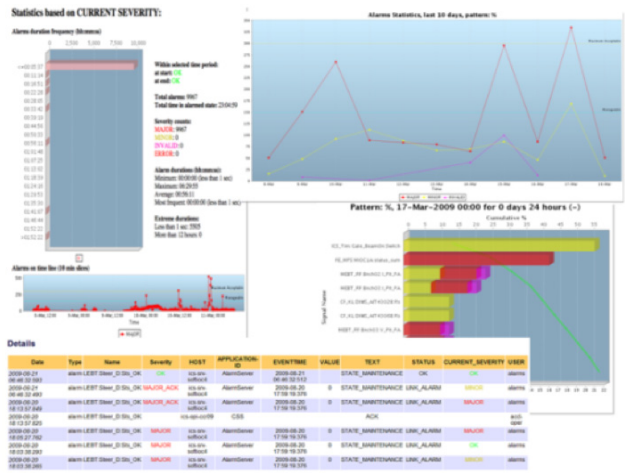

Figure 4: Web Reports.

SUMMARY

The BEAST has been operational at the SNS since February 2009. It has been stable through IOC reboots, online configuration changes and Oracle updates. It is very helpful in improving the configuration. Starting by importing a previous ALH setup of about 300 PVs with no guidance and few related displays, the configuration now has 400 PVs and all of them have guidance, related displays or links to operational procedures. The Alarm Client GUI indeed received the "best ever" praise from SNS operators.

REFERENCES

[1] http://www.aps.anl.gov/epics/extensions/alh.
[2] P.Gurd et al, "The New Soft-IOC-Based Alarm Handler at the Spallation Neutron Source", ICALEPCS07, Knoxville, October 2007.
[3] Karen White, Kay Kasemir, "Alarms Philosophy", ICALEPCS09, Kobe Japan, October 2009.
[4] M.Clausen, "Alarm Management System", PCaPAC, Ljubljana, Slovenia, Oct. 2008.
[5] DESY CSS web page: http://css.desy.de
[6] SNS CSS web page: http://ics-web.sns.ornl.gov/css