MANAGING ALARMS AND (LOG) MESSAGES – THE CSS WAY

Matthias R. Clausen, Jan Hatje, Gongfa Liu, Markus Moeller, Helge Rickens, Bernd Schoenbeurg, DESY, Hamburg, Germany

Abstract

The alarm system receives alarm messages from the control system and presents this information in a convenient way to the operators. Each function of the alarm system is handled by a separated component. At the beginning are applications collecting status changes from EPICS front end controllers through a persistent store, archive and filters to client applications with user interfaces. All applications are based on the platform Control System Studio that provides common interfaces and remote management for headless applications. Most applications are redundant to ensure a high reliability. The communication is managed by Java Message Service a specification by Sun. Because the message system is based on a common service it can also handle any kind of messages.

JAVA MESSAGE SERVICE

The Java Message Service (JMS) is an API from Sun Microsystems [1]. The components of the message system for alarm and log messages are using JMS for communication. JMS defines interfaces for asynchronous, loosely coupled and reliable communications between two or more clients.

JMS provides two communication models.

- **Publish and subscribe**: One or more receivers are subscribed on a topic where senders publish their messages. Sender and receiver do not know about each other but durable subscription ensures that a receiver gets all messages published in a topic.
- **Point-to-point**: A particular sender posts messages to a queue where a particular receiver reads and acknowledges the messages.

The JMS server for the alarm system is an implementation from Apache, ActiveMQ [2]. It is open source, covers the whole JMS API and the performance is compared to other open source implementations high.

For a high reliability the alarm system is using two instances of the ActiveMQ server. The management of multiple JMS servers is integrated in the ActiveMQ implementation.

CONTROL SYSTEM STUDIO

Control System Studio (CSS) [3] is a platform for control system applications with common interfaces for data types, a Data Access Layer (DAL), management services and centralised connection to external data sources. CSS is built on the Eclipse Rich Client Platform that is implemented in Java and therefore operating system independent. The applications handling alarm and log messages are build on CSS.

CSS inherits the plug-in structure of Eclipse. The separation of logical parts in components respectively plug-ins makes it easy to adjust CSS installations for different requirements. Depending on the environment CSS integrates another subset of plug-ins. Besides the installation as an operator interface CSS can run without the UI plug-ins in a headless mode as a server. CSS instances in headless mode are controlled from a CSS management instance with remote commands.

STRUCTURE OF DESY ALARM SYSTEM

The main issue of an alarm system is to decide which alarm message should be forwarded to which destination. Important alarms are sent via SMS to operators, alarms of low priority are just displayed on a terminal and stored in an archive.

Alarms are generated by the control system whenever a value exceeds a defined alarm limit. The IOC forwards each message to the Interconnection Server (ICS) via a direct TCP/IP connection (Fig 1). The ICS that receives messages from several IOCs, converts them in JMS message format and sends them to the JMS server.

From the JMS server alarm and log messages are distributed to all applications handling alarms. On the one hand there are the headless CSS instances. JMS2Ora writes all messages in a database for archiving. The components for the Alarm Message System (AMS) filter the messages and if they are important send them by SMS, Mail, … to the operators. On the other hand are the CSS applications that display the status of the alarm system.

ALARM SOURCES

The sources of alarm messages are the front end controller of the control system, IOCs for EPICS and D3 PCMs for D3.

On EPICS IOCs the software IOC Log Client holds the connection to the Interconnection Servers (ICS) and checks by beacons if the server is still online. If more than one ICS is connected to the IOC the IOC Log Client selects one ICS as the active one. Each new alarm state is sent by the IOC Log Client via a TCP/IP connection to the ICS as a text message in a specific format. To ensure the receipt of alarm messages by the ICS each message is acknowledged. Beside the alarm messages IOC Log Client send also caPutLog and SNL-log messages.

The ICS converts the message from the IOC in a JMS message of the format MapMessage. MapMessage is a definition of JMS and consists of key value pairs. Because MapMessage is a common format the whole communication system can be used for all kinds of messages like log etc.

The ICS is built on CSS and runs as a server in headless mode. It implements the interfaces for remote management that each CSS instance with plug-ins for remote management can control ICS instances.

The other sources for alarms are D3 PCMs similar to EPICS IOCs. The alarm messages generated by D3 are appended to a file. This file is observed by the D3 Alarm Reader and each new entry is converted in a JMS MapMessage and sent to the JMS server. Like the ICS the D3 Alarm Reader is built on CSS.

Because the components of the alarm system are generic also CSS instances are using the system. Beside logging on the console or in files they can send log messages in JMS format.

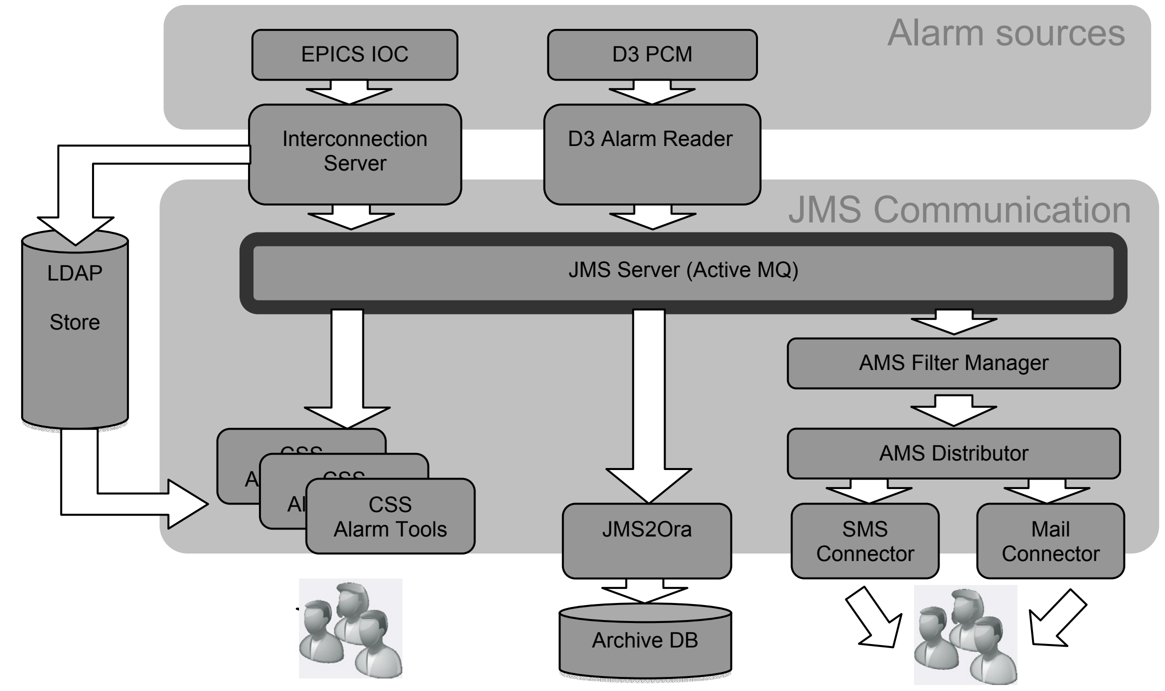


Figure 1: Overview of the alarm System. The top part shows the sources of specific alarm messages. They are converted in JMS format and distributed by the JMS Server to the components of the alarm system. Note that just the logical way of alarm messages are shown. Technically each message is sent via the JMS server.

PERSISTANT STORE AND ARCHIVE

The message system only forwards alarms from the sources to the clients. To get information of the current alarm state of the system or previous alarm messages a persistent store and archive is necessary.

The persistent store for the alarm status is a LDAP-(Lightweight Directory Access Protocol) [4] directory. The channels with their alarm status and additional information are organized in a hierarchical way predefined by the structure of the IOCs. The ICS updates the LDAP directory every time a new alarm message is received.

All alarm and log messages are stored in a SQL database by the application JMS2Ora. A SQL database as an archive source makes it very easy to retrieve alarms of a certain period or search for specific properties.

CSS ALARM VIEWER

CSS shows the status of the alarm system with the Alarm Tree Viewer and the Alarm Table Viewer. From both applications the user can acknowledge alarms. The acknowledgement is distributed as a JMS message from the CSS instance where it is set to update the other CSS instances, persistent store and archive.

Alarm Tree Viewer

The CSS application Alarm Tree Viewer shows channels of the control system with their alarm status ordered in a tree structure. By default the structure is oriented by IOCs but the user can order all channels for their own requirement.

The alarm state of a channel is indicated by the colour of the according icon. Previous alarm states are displayed by a background icon. To show alarms of channels in a not expanded sub tree they are propagated to the root of the tree. Thus the root shows always the alarm with the highest severity in the system.

Alarm Table Viewer

The Alarm Table Viewer is message oriented and displays the alarm and log messages in a table. Because the message content is not fixed the number and titles of table columns are configurable. The keys of the JMS MapMessage are mapped to the table columns and the according values are displayed in a row.

There are three types of viewers:

- **Log Viewer**: All messages are displayed in this table whatever they are of the type alarm, log etc. The messages are ordered by timestamp, the latest at top. If a configurable number of messages are exceeded the oldest messages will be deleted.
- **Alarm Viewer**: The Alarm Viewer shows only alarm messages ordered by the severity and in the second step by timestamp. If a more recent message for the same channel is received the previous one is greyed out and the oldest one will be deleted.
- **Archive Viewer**: Messages in the archive data base can be accessed with the Archive Viewer. To set a time period and conditions for message properties a filter is available.

A headless CSS instance with the data model and JMS part of the Alarm Table View provides the data by a servlet for an IPhone application. The alarms are displayed in a simplified alarm table and the user can check the state of process variables that have generated the alarms.

ALARM MANAGEMENT SYSTEM

Operators running facilities do not want to see every alarm message. They are only interested in the important ones and depending on the situation they want to be informed on different ways.

With the Alarm Management System (AMS) users can define complex filters to get only the messages they are interested in. To each filter one or more action can be assigned that specify if the message should be forwarded by Mail, SMS, etc.

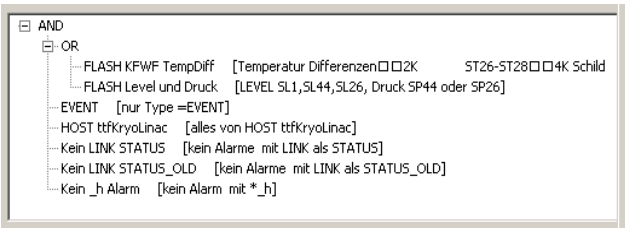
AMS consists of three components that execute the messages one after another.

Filter Manager

The Filter Manager receives all alarm messages and checks if there is a filter match (Fig. 1). If so a JMS message is generated with the original message, the id of the matched filter and sent to the distributor.

A filter is a composition of conditions connected with AND and OR relations. Conditions are simple equations of message content with parameters. For example 'is value of message key NAME equals to VALVE_01' or 'is value of message key VALUE greater than 50'. Fig. 2 shows the structure of a filter with basic conditions in the user interface of the AMS Configurator.

New filter configurations set in AMS Configurator are stored in an Oracle database. The Filter Manager observes the database for updates and copies the configuration in a local Derby database exclusive used from the Filter Manager. With the local database the Filter Manager enhances its reliability because the connection to Oracle is not necessary.



```
⊟ AND
   ⊟ OR
      FLASH KFWF TempDiff    [Temperatur Differenzen□□2K        ST26-ST28□□4K Schild
      FLASH Level und Druck    [LEVEL SL1,SL44,SL26, Druck SP44 oder SP26]
   EVENT    [nur Type =EVENT]
   HOST ttfKryoLinac    [alles von HOST ttfKryoLinac]
   Kein LINK STATUS    [kein Alarme  mit LINK als STATUS]
   Kein LINK STATUS_OLD    [kein Alarme  mit LINK als STATUS_OLD]
   Kein _h Alarm    [kein Alarm  mit *_h]
```

Figure 2: The user interface of Filter Manager shows the conditions of a filter in a tree structure.

Distributor and Connectors

The Distributor prepares the messages from the Filter Manager and forwards them to one of the Connectors according to the related action.

Like filters actions are configured with the AMS Configurator and the settings are stored in the Oracle database. For now there are only actions to forward messages on different ways. But the system is easy extensible by new action types and connectors that for instance set new values on control system channels if an alarm message is received.

Receivers of alarm messages can be singular receivers or groups of receivers. In groups members can be set active or inactive, a minimum number of active members can be set and the delay for acknowledges.

Messages for Connectors prepared by the Distributor contain just the necessary information for the contact e.g. mobile number or mail address and the content for the receiver. For now there are Connectors for SMS, Mail, Voice Mail and JMS topic available. The JMS topic Connector can be used to filter messages and post them in a JMS topic. In this way messages for the Alarm Table Viewers can be filtered.

CONCLUSION

AMS is now in use at DESY for one year. One month ago also Alarm Table and Tree are used by the operators. Apart from some small bugs the whole system runs successfully.

The redundant layout of the most components makes the system very reliable and because the components are based on CSS they are easy manageable from each CSS instance. Another benefit of the system is the generic design that we can use it for any kind of messages.

REFERENCES

[1] Java Message Service API Overview, http://java.sun.com/products/jms/overview.html.
[2] Apache ActiveMQ, http://activemq.apache.org/.
[3] M. Clausen, DESY. CSS - Control System Studio, 7. ICALEPCS 2007.
[4] Lightweight Directory Access Protocol (LDAP): The Protocol. RCF 4511.