Improving User Information by Interfacing the Slow Control's Log and Alarm Systems to a Flexible Chat Platform

Michael Ritzert (Heidelberg University, Heidelberg)
For the Belle II PXD collaboration.

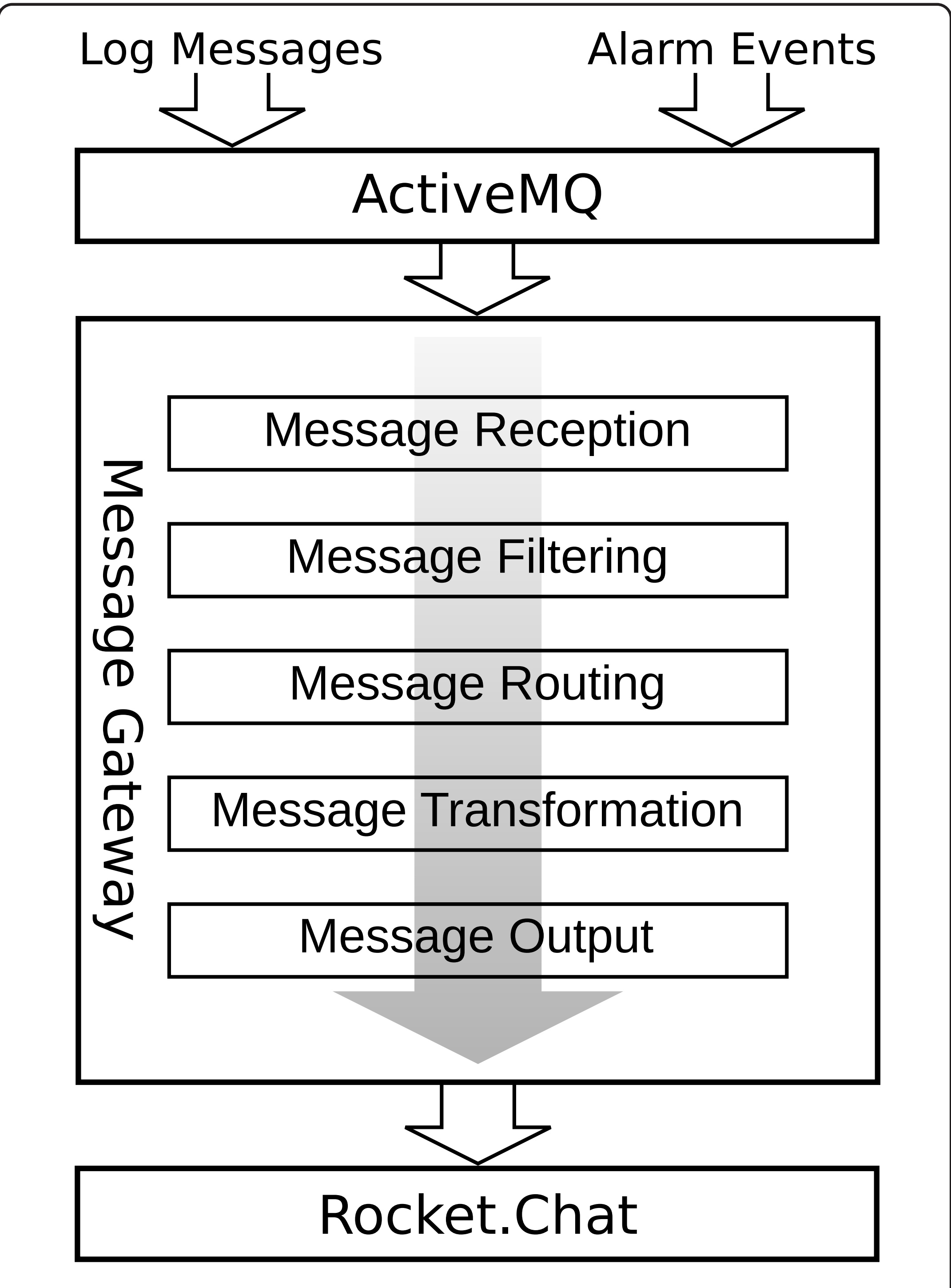
17th International Conference
on Accelerator and Large Experimental
Physics Control Systems
Brooklyn, NY, USA
October 2019

Abstract

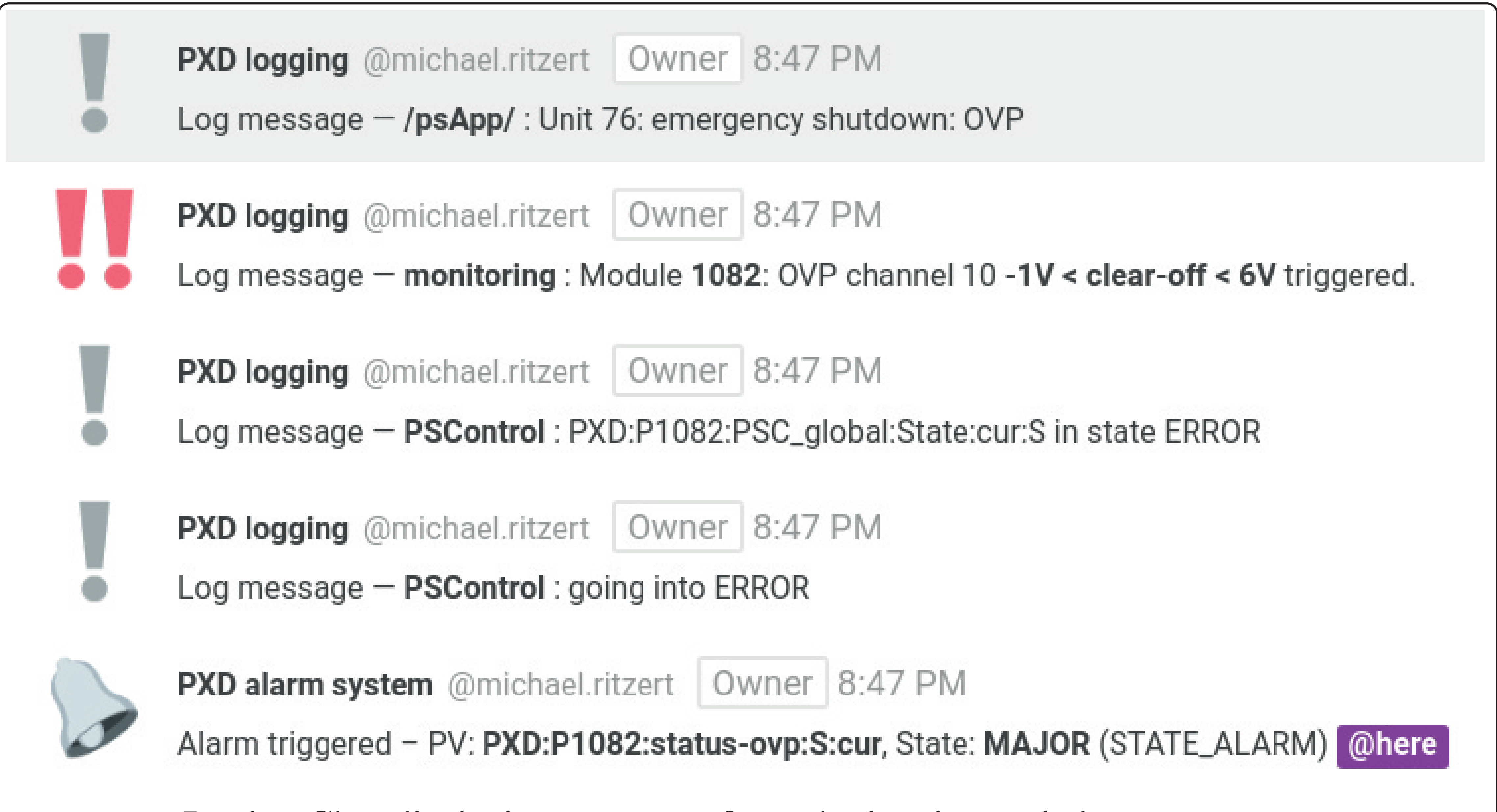
Research groups operating large experiments are often spread out around the globe, so that it can be a challenge to stay informed about current operations. We have therefore developed a solution to integrate a slow control system's alarm and logging systems with the chat system used for communication between experimenters. This integration is not intended to replace a control screen containing the same information, but offers additional possibilities:

- Instead of having to open the control system's displays, which might involve setup work (VPN, remote desktop connections, …), a web interface or an app can be used to track important events in the system.
- Messages can easily be filtered and routed to different recipients (individual persons or chat rooms).
- Messages can be annotated and commented on.

The system presented uses Apache Camel to forward messages received via JMS to Rocket.Chat. Since no binding to Rocket.-Chat was available, this interface has been implemented. On the sending side, a C++ logging library that integrates with EPICS IOCs and interfaces with JMS has been designed.



Overview of the Implemented System.


PXD logging @michael.ritzert Owner 8:47 PM
Log message — **/psApp/** : Unit 76: emergency shutdown: OVP

PXD logging @michael.ritzert Owner 8:47 PM
Log message — **monitoring** : Module **1082**: OVP channel 10 **-1V < clear-off < 6V** triggered.

PXD logging @michael.ritzert Owner 8:47 PM
Log message — **PSControl** : PXD:P1082:PSC_global:State:cur:S in state ERROR

PXD logging @michael.ritzert Owner 8:47 PM
Log message — **PSControl** : going into ERROR

PXD alarm system @michael.ritzert Owner 8:47 PM
Alarm triggered – PV: **PXD:P1082:status-ovp:S:cur**, State: **MAJOR** (STATE_ALARM) @here

Rocket.Chat displaying messages from the logging and alarm systems.

System Overview

The gateway combines data from the BEAST alarm system and the message log. Both systems are configured to publish their messages via the ActiveMQ message broker, that is central to the distribution of all messages in the system. The gateway is registered as a listener on the respective channels. It receives all messages, filters them and forwards to the Rocket.Chat server.

The gateway is implemented based on Apache Camel, a message routing engine implemented in Java. Received messages are passed through several modules connected to build a route.

For **Message Reception**, a module to receive from ActiveMQ is readily available.

Filtering is performed in a Java module to accept only messages related to the triggering of an alarm condition, and messages with high log levels.

Messages can then be **routed** to different recipients (channels or individual persons). The routing is implemented in Java and has full access to all message metadata.

For posting to Rocket.Chat, the messages have then to be **transformed** from the native BEAST and log formats with all metadata to plain text.

Finally, the messages are **output** to Rocket.Chat. In case the messages have been tagged accordingly, i.e. with @here, the Rocket.Chat server notifies the users.

Logging Library

We implemented a C++ library for logging via ActiveMQ. The STOMP protocol is used to send the message to Active-MQ, where it is converted to the JMS format also used by the alarm system.

When included in an EPICS IOC, IOC Shell commands are available to configure the logging to different sinks (console, logfile, STOMP). The minimum log levels per message source and sink can be adjusted at runtime.

A threaded design is used to decouple the main thread from the potentially time-consuming output operations, and avoid blocking the normal operation of the IOC.

Rocket.Chat Integration

For the output to Rocket.Chat, a new module has been implemented in Java. It uses the REST API to post messages. Since the protocol is stateless, it is easy to implement. There is only a single HTTP request per posted message.

The actual implementation consists of two parts: A small, generic module implementing the posting of messages to Rocket.Chat, and a second module that wraps around it to provide the interfaces as required by Camel.