STAR Online Meta-Data Collection Framework: Integration with the Pre-existing Controls Infrastructure

D Arkhipkin, J Lauret
Physics Department, Brookhaven National Laboratory, Upton, NY 11973-5000 USA
E-mail: arkhipkin@bnl.gov, jlauret@bnl.gov

Abstract. One of the ST AR experiment’s modular Messaging Interface and Reliable Architecture framework (MIRA) integration goals is to provide seamless and automatic connections with the existing control systems. After an initial proof of concept and operation of the MIRA system as a parallel data collection system for online use and real-time monitoring, the ST AR Software and Computing group is now working on the integration of Experimental Physics and Industrial Control System (EPICS) with MIRA’s interfaces. This integration goals are to allow functional interoperability and, later on, to replace the existing/legacy Detector Control System components at the service level. In this report, we describe the evolutionary integration process and, as an example, will discuss the EPICS Alarm Handler conversion. We review the complete upgrade procedure starting with the integration of EPICS-originated alarm signals propagation into MIRA, followed by the replacement of the existing operator interface based on Motif Editor and Display Manager (MEDM) with modern portable web-based Alarm Handler interface. To achieve this aim, we have built an EPICS-to-MQTT [8] bridging service, and recreated the functionality of the original Alarm Handler using low-latency web messaging technologies. The integration of EPICS alarm handling into our messaging framework allowed ST AR to improve the DCS alarm awareness of existing ST AR DAQ and RTS services, which use MIRA as a primary source of experiment control information.

1. Overview of the STAR Detector and the MIRA Framework

The ST AR [1] experiment at the Relativistic Heavy Ion Collider (RHIC [2]) is one of the foremost experiments in the study of the medium created in heavy-ion collisions. It consists of 18 subsystems as exemplified by Fig. 1, each of which performs a specific task. To name a few - the Time Projection Chamber is used for tracking and particle identification, the Time Of Flight detector is used for triggering, the Barrel Electromagnetic Calorimeter measures the electromagnetic shower energy of high-$p_T$ particles. For the heavy quarkonium studies the recently installed Heavy Flavor Tracker and Muon Telescope Detector are important. Almost six hundred scientists and engineers are working on the extraction of physics results from
the data taken by STAR detector. At the heart of the STAR’s data acquisition system, there is a Messaging Interface and Reliable Architecture (MIRA) framework (see Fig. 2 for an overview), built specifically to integrate the online services and streamline the meta-data collection, processing, storage and visualization. STAR’s experimental control system had 40,000 process variables (PVs) at start (circa 1996), and has grown to, today, 80,000 PVs. Detector’s online ecosystem includes Data Acquisition (DAQ), Run-Time Services (RTS), Control System (CS), Alarm Handler (ALH), Monitoring, subsystem-developed services – all talking to each other. Due to the continual upgrade process of STAR subsystems, an obvious need for the modern service integration bus was clear. The MIRA framework was created to answer that need.

**Figure 2.** A high-level layered architecture overview of the MIRA framework. CDEV (Common DEVice, [9]) is a C++ and Java framework for developing portable control applications - interface which feeds RHIC data into STAR control system. AMQP - Advanced Message Queuing Protocol, used along with MQTT protocol in MIRA for service orchestration purposes. MQ 2 DB - set of services, which archive MQ streams to the persistent storage backend (MySQL [6] and MongoDB [7] databases). ESPER engine ([5]) is a Complex Event Processor engine, used to preprocess raw MQ streams.

As reported in our previous work ([3],[4]), the architecture of our framework is based on a few concepts that together help achieve the goals of STAR collaboration. MIRA is using modern messaging technologies (AMQP, MQTT), and features a scalable architecture, inter-operable low-overhead message format, payload-agnostic messaging to enable legacy service support, and quality of service regulation. It is a distributed, reliable, and a high-availability system for the efficient collection, aggregation and moving of large amounts of meta-data from many different sources to a centralized data store.
MIRA supports Complex Event Processing (CEP) via pluggable interface to the ESPER engine. Responding to the needs of the online experts, MIRA has recently expanded its coverage to the Control System realm and experimental Alarms Handling in particular. Details of the CS and ALH upgrade processes are in the focus of this paper’s report.

2. Experiment Control System: Upgrade Process

STAR online systems evolve at different rates, the protocols and formats used by a given subsystem can change every year as devices are being phased in and out or upgraded, often irrespective of the systems around them. So, the usual procedure for a system upgrade involves studies of the online ecosystem, its components and dependencies, development of the robust upgrade process and, finally, the execution of the staged upgrade plan. The final output of this process is a continual upgrade scheme that needs to be reviewed and maintained periodically.

MIRA was designed to connect what is essentially a distributed system of components that are (at best) tightly coupled or not even designed to work together. It helps tackle these interactions via an API, that make subsystems amendable to cooperation and exchange of cleaned and pre-processed meta-data streams. Let’s review the upgrade process on the example of the legacy Slow Controls infrastructure integration into MIRA.

The current STAR Detector Control System is largely based on Experimental Physics and Industrial Control System (EPICS), and has been developed to operate independently of any other service in the online domain. Here, we have four major components to review in a course of our upgrade plan: device drivers, Input-Output Channels (IOCs, also performing a role of local database), Graphical User Interface (GUI) and the EPICS Alarm Handler application. In addition, we must keep in mind that new detector hardware is sometimes shipped with EPICS drivers only. Also, STAR manpower is sparse at times, and we simply cannot upgrade everything at once (e.g. during the downtime periods). In order to cope with the seamless upgrade and to avoid drastic changes in well-defined experimental procedures, we decided to gradually replace the identified components with newer technologies provided by MIRA’s API and services.

MIRA has proven to be an extremely flexible platform to enable the aforementioned process.

3. Bridging EPICS and STAR Online services using MIRA API

The overview of MIRA services and corresponding data flow is presented on Fig. 3. Let’s take the Time Projection Chamber (TPC) as a specific subsystem for our upgrade plan. TPC is controlled by EPICS-based TPC IOC, which is also attached to the STAR Alarm Handler. Ideally, we want to migrate all TPC controls from EPICS to MIRA, but we need to do this in parallel to the existing EPICS-based setup.

The upgrade goal is achieved by designing a set of microservices, that replicate the functionality of the existing components, allowing us to have a one-to-one substitute in terms of API and deployment. This approach offers important benefits to the upgrade process, as it takes care of hiding the differences of the underlying technologies. Also, scalable, loosely coupled microservice architecture has several important advantages over monolithic architectures - it is easier to deploy new versions of services frequently, each service can be developed and deployed independently, and it eliminates any long-term commitment to a technology stack.

Our first component is the link between EPICS IOC and Alarm Handler. To propagate EPICS PV changes, we wrote a three-tier microservice, which is essentially a bi-directional translator (bridge) between EPICS PV format and MIRA messaging format. This service was coded as javascript runtime (Node.js service), with the intent of easier maintenance and support by future generations of STAR Software and Computing group members. Compact translator instances could be deployed where needed, which eliminates the common problem of cross-network boundaries typical for EPICS setups.
Having EPICS-to-MIRA translator in place, our next step is to replace the IOC itself, removing the overhead of protocol conversion. Again, here we only need to mimic the most basic functionality of IOC, namely set / get methods, supporting only a subset of PV data types. Since legacy codes were not optimized for low latency or extreme data rates, our straightforward-programmed service was able to provide the required functionality.

Finally, we needed to upgrade the User Interface (UI) of detector controls and alarm handlers. While there are several UI frameworks available on a market, we decided on making our own framework using HTML5, JavaScript, Node.js and MongoDB, which comprises a full development stack, with the intent to provide modern web-based UI. We did review the recent developments of the community in this area, namely tools and frameworks allowing to expose EPICS to internet users. First, we reviewed WebPDA, which is a protocol for accessing real-time data streams using WebSocket technology. While it does provide rich web interface functionality, it requires intermediate java application server (extra overhead), and it was not clear how to allow bi-directional talk of non-EPICS services. Second, there is a EPICS Boy toolkit. It is an attractive opportunity because it is developed and maintained externally by EPICS experts, but it provides potentially limited scalability options, and introduces strong bias towards specific data exchange protocol and data format. All reviewed tools were found to be very task-specific, rarely (if at all) mentioning service integration, which is crucial for the long-running STAR experiment’s infrastructure.

4. Use-Case Example: Alarm Handler Upgrade
We illustrate a specific upgrade use-case on the example of the STAR Alarm Handler. Historically, STAR relied on the Motif Editor and Display Manager (MEDM)-based Alarm Handler, provided with the EPICS distribution. It is presented as single instance of local application, running at STAR Control Room on the dedicated PC and configured via plain text file residing at specific network-mounted node.

Our task was to replace the Alarm Handler graphical interface, along with the configuration file by MIRA-provided services, keeping the usual look and feel for the detector operators. This task is logically divided into three subtasks: alarm propagation, alarm configuration, and
user interface. First subtask was fulfilled by using streams of MIRA-provided events, exported
by EPICS-to-MIRA bridge microservices (described above). Alarm configuration was replaced
with basic MIRA’s storage service, and now is using MongoDB as its backend engine. User
Interface was re-created as web-based application written in HTML5/JavaScript, and attached
to message-queuing server via MQTT over WebSockets connector.

Figure 4. Alarm Handler interface comparison: (left) old MEDM interface, (right) new HTML5 interface.
New interface supports broad range of application packaging formats, including mobile versions.

Propagated alarms use the familiar hierarchy of [system] / [subsystem] / [component], and
reach Alarm Handler application directly via MQTT bus. Depending on the configured bridging
modes, events could be sent back to EPICS from the UI, or from the Event Processor-emitted
streams - for example, collective states of components or modules.

New UI allowed us to not only keep the familiar look (as illustrated by Fig. 4), but also to
provide new functionality. First, the new UI has an option to suppress nuisance alarms, resulting
from fluctuating border states of detectors. Second, Alarms can now be easily accessed by STAR
experts using their desktop PC, cellphones and tablets virtually from anywhere in the world,
reducing the time to respond to critical issues. Note that the upgraded Alarm Handler UI can be
used ‘as-is’ via a browser, or it could be packaged as standalone application for Linux, Windows,
MacOS, iOS or Android platforms using NW.js and other frameworks — in other words, the
work on providing a new UI opens the possibility to access it from any devices including mobile
ones.

5. Summary and Outlook
STAR started with its Messaging Interface and Reliable Architecture to archive, redirect, filter
and act (CEP) upon streaming meta-data collected online. MQ-based service bus provided
STAR with stable and scalable solution, and enabled near-real-time view of the experimental
status for all collaborators.

In this work we showed how we could implement and replace the EPICS Alarm Handler UI
by a web interface relying on our recently developed EPICS-to-MIRA bridge approach. STAR
Online service integration is now greatly improved for Data Acquisition and Run-Time systems,
as our integrated processing of the alarms produced by various STAR subsystems is completed.
Future plans for the MIRA framework development include stripping EPICS-to-MIRA
bridging and EPICS IOC emulator codes from the experiment-specific dependencies. This will
allow STAR to provide reusable generic implementations which could be employed in other
HEP/NP experiments. Such development is greatly supported by the interest in our work from the Collider-Accelerator community, as well as the prospect of future Nuclear Physics experiments.

Acknowledgments
This work was supported by the Office of Nuclear Physics within the U.S. Department of Energy’s Office of Science.

References
[1] K. Ackermann et al. (STAR Collaboration), STAR detector overview Nucl. Instrum. Meth. A 499, 624 (2003).
[2] M Harrison, T Ludlam and S Ozaki 2003, RHIC project overview Nucl.Instrum.Methods Phys.Res.A 499 235244
[3] D Arkhipkin, J Lauret, W Betts and G Van Buren 2012. Online Meta-data Collection and Monitoring Framework for the STAR Experiment at RHIC J. Phys.: Conf. Ser. 396 012002
[4] D Arkhipkin, J Lauret 2015, STAR Online Framework: from Metadata Collection to Event Analysis and System Control J. Phys.: Conf. Ser. 608 012036
[5] "Esper: an open source event stream processing and event correlation engine [Online], http://www.espertech.com/products/esper.php"
[6] "MySQL: the world's most popular open source database [Online], http://www.mysql.com/"
[7] "MongoDB: scalable, high-performance, open source NoSQL database [Online], http://www.mongodb.org/"
[8] "MQ Telemetry Transport [Online], http://mqtt.org/"
[9] "CDEV - Common DEVice [Online], http://www.jlab.org/cdev/"