Extending Phoebus Data Browser to alternative data sources

Mihnea Romanovschi[1], Ivan Finch[1], Gareth Howells[1]

1. ISIS Neutron and Muon Source, Harwell Campus, Oxfordshire, OX11 0QX, UK

Introduction

The Phoebus user interface to EPICS is an integral part of the upgraded control system for the ISIS Neutron and Muon Source accelerators and targets. Phoebus can use the EPICS Archiver Appliance, which has been deployed as part of the transition to EPICS, to display the history of PVs. However, ISIS data historically has and continues to be stored in the InfluxDB time series database. To enable access to this data, a Python application to interface between Phoebus and other databases has been developed. Our implementation utilises Quart, an asynchronous web framework, to allow multiple simultaneous data requests. Google Protocol Buffer (Protobuf), natively supported by Phoebus, is used for communication between Phoebus and the database. By employing subclassing, our system can in principle adapt to different databases, allowing flexibility and extensibility. Our open-source approach enhances Phoebus's capabilities, enabling the community to integrate it within a wider range of applications.

The components of the entire system can be divided roughly into four main components following a Model-View-Controller approach in design:

- **Phoebus Application:** it represents the view side of the system; it displays the control screens and the information from our archiving database. It also performs requests to the controller regarding what information is to be displayed.
- **Quart Server:** an asynchronous server that acts as the controller. It unpacks the HTTP requests and delivers data from the model in an asynchronous manner. This approach minimizes the latency between processing information from a database and presenting it to Phoebus, ensuring optimal performance.
- **Model:** Performs the data conversion from the format that the databases API responds into the format that Phoebus expects, in this case a **Protobuf** binary.
- **Database:** We use **InfluxDB** for data collection and **CouchDB** for metadata lookup. They may also perform statistical analyses as needed.

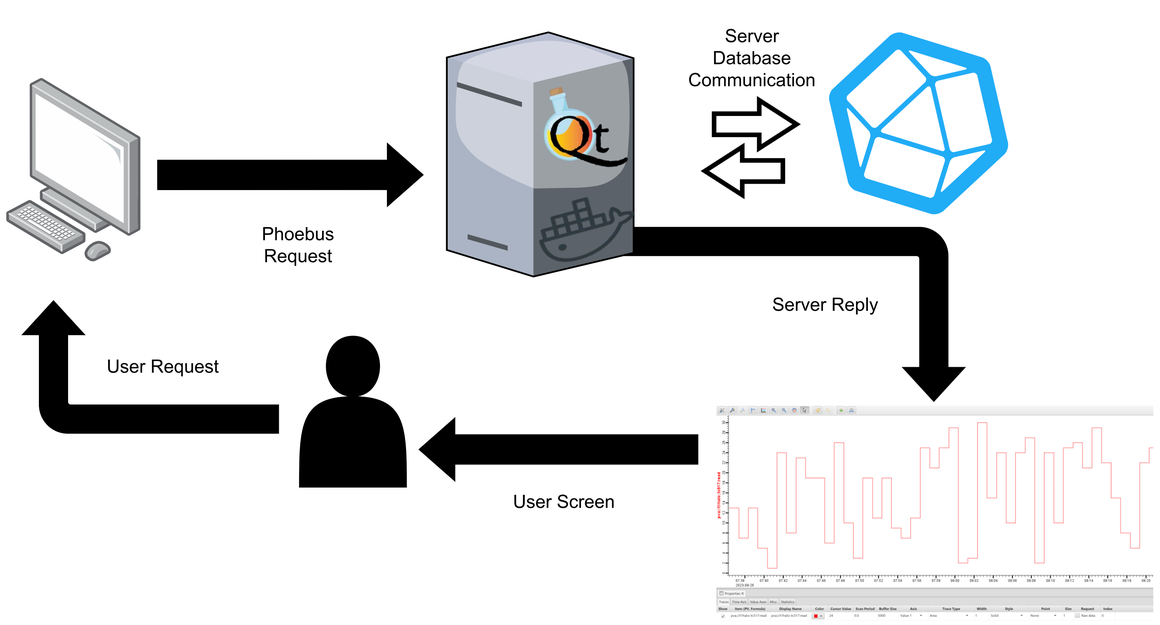

Figure 1: High level overview of architecture

Server Architecture

The server is an application separate from Phoebus and the databases that it communicates with. It runs in a Docker environment using the Green Unicorn HTTP Server, a platform capable of handling multithreaded operations, although this multithreaded functionality has yet to be used by the current application.

Our current workflow for introducing a new Process Variable (PV) is as follows:

1. Define metadata parameters, such as name, type, precision and other elements for the display. To track this data, we use **CouchDB**.
2. Add it to our archiving tool, **EPICS Archiver Appliance** or **InfluxDB.**

Figure 2 illustrates the internal operation of the application. It seamlessly handles incoming requests from Phoebus, mimicking the Epics Archiver Appliance API. These requests are then transformed into the appropriate API calls for the chosen database. When metadata retrieval is required, the server connects to **CouchDB** to retrieve the information, converting it into the **Protobuf** protocol. Subsequently, this metadata is incorporated as a header in the response received from **InfluxDB**.

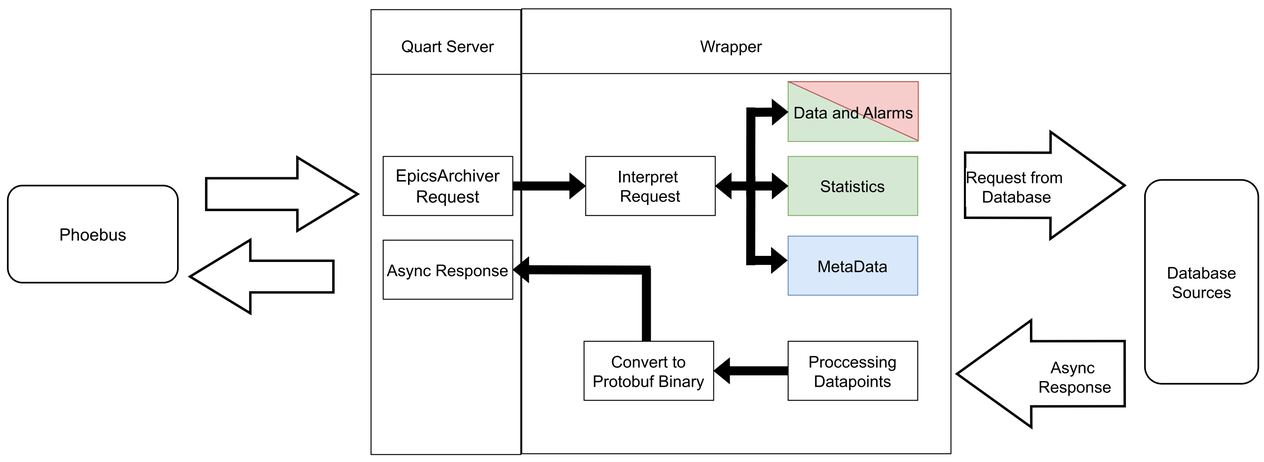

Figure 2: Internals of how the server operates.

By default, the EPICS Archiver Appliance begins logging each PV using incremental timestamps based on the first observation of the PV. If you wish to access the data then an appropriate conversion from the queried time to the required incremental timestamp must be performed. In the case of **InfluxDB** this would require adding a new field to your data points. This can be avoided by setting the year to 1970, the start of the UNIX Epoch.

Figure 3 breaks down the Protobuf binary that the Quart server sends to Phoebus. The components are as follows:

- **Header (Blue):** Represents the metadata, information about the PV/Channel that we request. The **type** is sent in both the header and body of the response. The header is separated from the body by a new line.
- **Body (Green and Red):** The body contains every point we sample from the archiver separated by new lines. It also contains the status and the severity of any alarms that have been registered at a given time. It can be either a field from Influx or a completely different data source, for example an **ElasticSearch** server.

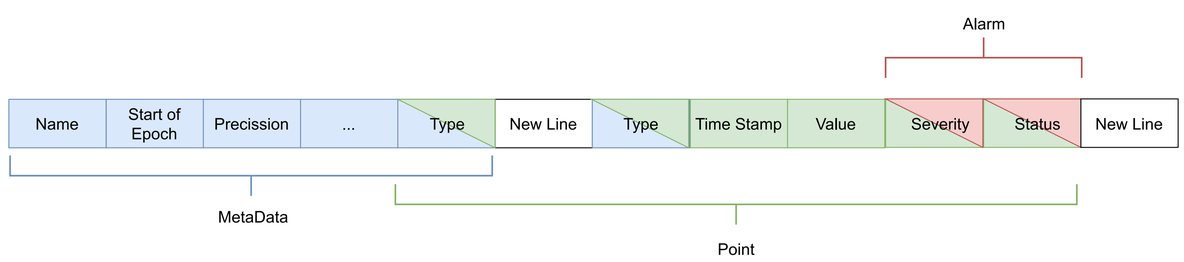

Figure 3: Breakdown of the Protobuf binary.

Further Work

Future development is planned to focus on three areas:

1. **Modularity:** Currently the application only implements support for one database. We plan to make it easier to change between databases and allow for multiple data sources to be connected to allow for a smooth transition between systems, for example from the **EPICS Archiver Appliance** to **InfluxDB.**
2. **Alarm Handling:** The hope is to move to an **ElasticSearch Database** to store and serve alarms to different systems that request it, including the wrapper.
3. **Testing:** The application requires more testing to ensure that the **EPICS Archiver Appliance API** is fully replicated, for example. At present, the ability to efficiently handle simultaneous calls for PV statistics to optimize display is limited and not fully supported.

Conclusion

We plan to contribute to the open-source community of both Phoebus and **InfluxDB** through this project. We have provided an example on how to change the data source used by the Data Browser without the need to modify the Phoebus source code.

Find out more about ISIS