ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUSDSC08

PHOEBUS TOOLS AND SERVICES

K. Shroff, Brookhaven National Laboratory, Upton, NY, USA
R. Lange, ITER Organization, St. Paul lez Durance, France
T. Ford, Lawrence Berkeley National Laboratory, Berkeley, CA, USA
G. Weiss, European Spallation Source ERIC, Lund, Sweden
K. Kasemir, Oak Ridge National Laboratory, Oak Ridge, TN, USA
T. Ashwarya, FRIB, East Lansing, MI, USA

Abstract

The Phoebus toolkit consists of a variety of control system applications providing user interfaces to control systems and middle-layer services. Phoebus is the latest incarnation of Control System Studio (CS-Studio), which has been redesigned replacing the underlying Eclipse RCP framework with standard Java alternatives like SPI, preferences, etc. Additionally, the GUI toolkit was switched from SWT to JavaFX. This new architecture has not only simplified the development process while preserving the extensible and pluggable aspects of RCP, but also improved the performance and reliability of the entire toolkit.

The Phoebus technology stack includes a set of middle-layer services that provide functionality like archiving, creating and restoring system snapshots, consolidating and organizing alarms, user logging, name lookup, etc. Designed around modern and widely used web and storage technologies like Spring Boot, Elasticsearch, MongoDB, Kafka, the Phoebus middle-layer services are thin, scalable, and can be easily incorporated in CI/CD pipelines. The clients in Phoebus leverage the toolkit's integration features, including common interfaces and utility services like adapter and selection, to provide users with a seamless experience when interacting with multiple services and control systems.

MOTIVATION

The workflows of operators, engineers, and scientists interacting with control systems like EPICS [1] often involve interacting with multiple applications, each designed for specific use cases. Some applications are used to display real-time data from various signals, some visualize historical data or consolidated alarms, while others provide a set of operations to write directly to the control system or trigger actions in middle-layer services, such as setting predefined values from a snapshot.

Phoebus/Control System Studio (CSS) [2] was developed to streamline these workflows by offering a suite of integrated applications and a framework that simplifies the development of such applications. For end-users, this translates to a collection of applications that are easy to navigate between, where data can be seamlessly transferred from one application to another, ensuring a consistent user experience. For developers, CSS provides a framework for creating applications that seamlessly integrate into this ecosystem without requiring tight dependencies on other applications. It also supports modules that offer access to shared and optimized resources, such as connections to EPICS Process Variables (PVs), REST clients, and more.

Stepping into the Sun

The initial incarnation of Control System Studio (CSS) was built on top of the Eclipse Rich Client Platform (RCP) which provided a great foundation, offering a plethora of essential features required for building extensible, pluggable applications. Over time, most of the core functionalities and capabilities originally provided by the Eclipse RCP framework found their way into the standard Java Development Kit (JDK) library itself. The incorporation of these Eclipse RCP features into the JDK introduced an opportunity to simplify the application development by allowing developers to leverage the standardized Java library, and a more cohesive and streamlined development process that comes with it. This transition was further accelerated by the increasing complexity of the Eclipse RCP framework, which prompted us to embrace the native Java ecosystem for building Phoebus as a modern, efficient, and maintainable application.

PHOEBUS ARCHITECTURE

Java Service Provider Interface

Phoebus/CSS is employed across diverse international scientific and industrial setups, each with its unique requirements. Hence, enabling each site to curate customized products comprising relevant applications and tools within their control environment is crucial. Equally important is Phoebus's seamless integration with existing software stacks.

Phoebus facilitates this adaptability through a list of Service Provider Interfaces (SPIs) for contributing applications. Java SPI is a mechanism to modularize a software framework. In the case of Phoebus, SPI allows applications to register for file extensions, contribute menu or toolbar entries, offer data sources or access to a site-specific logbook. To add a new application, one simply needs to provide an implementation of the app SPI and include it in the classpath. This approach encourages the contribution of new applications and simplifies the creation of site-specific products tailored to each site's specific needs.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUSDSC08

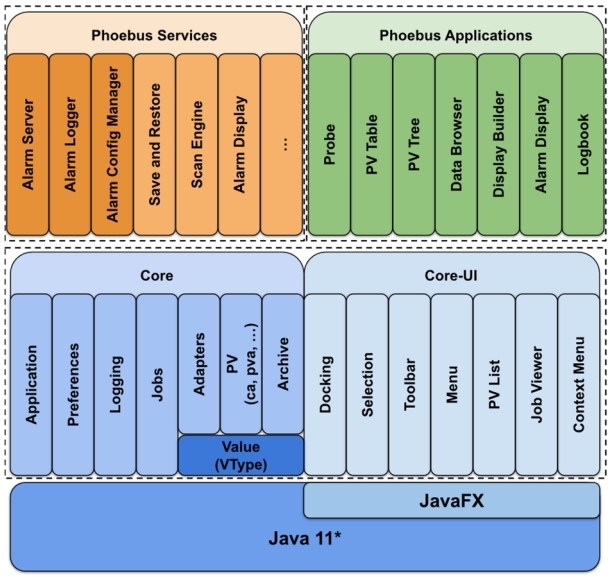

Figure 1: Phoebus Architecture.

Furthermore, Phoebus offers SPIs for extending functionality, such as adding actions to context menus,into a site's existing control stack. These SPIs cover a range of functions, including support for different data menus, toolbars, and more. Additionally, there are SPIs designed for integrating Phoebus applications seamlessly into a site's existing control stack. These SPIs cover a range of functions, including support for different data sources and control protocols, as well as integration with site-specific services like logbooks and archivers.

Phoebus Core Libraries and Service

The Phoebus framework (Fig. 2) simplifies the development of user applications by offering a set of core modules that provide solutions for various aspects of application development. These core modules encompass several key functionalities.

One of the core modules offers common UI elements that can be easily integrated into applications, ensuring a consistent and user-friendly experience. Another core module focuses on data processing pipelines, allowing developers to efficiently manipulate and transform data from various sources. Additionally, Phoebus includes multithreading and job scheduling services, ensuring that tasks are executed in a controlled and organized manner.

Phoebus also incorporates essential core libraries, such as Core-PV and EPICS VType. These libraries simplify application development by providing support for communication over various protocols and standardized abstractions for mapping different protocol messages and data types, reducing complexity.

The framework includes formula functions, that enables users to define pipelines for processing values from data sources like Channel Access (CA) or PVAccess (PVA). Importantly, these operations are executed efficiently off the UI thread, enhancing the responsiveness of UI applications. The core framework also offers services for scheduling and managing jobs, along with the associated execution thread pools.

Additionally, the framework includes a Selection Provider and Adapter Framework that facilitates the mapping of data types between different applications. This functionality enables seamless data exchange between applications without creating rigid dependencies. Adapter Factories within this framework allow developers to define how selections in their application can be mapped to other data types, such as log entries or process variables, without introducing direct dependencies. This can be achieved by registering adapter implementations using Java Service Provider Interface (SPI).

In summary, the Phoebus framework, as shown in Fig. 1, provides a comprehensive set of tools and core libraries that empower developers to create applications efficiently. These modules enhance the development process by ensuring consistent UI elements, efficient data processing, responsive multithreading, and flexible data mapping between applications while minimizing dependencies.

PHOEBUS APPLICATIONS

A group of applications for interacting with control systems and middle layer services.

Display Builder

The Display Builder combines an interactive editor for creating control system displays with a runtime to execute them [3]. It offers a large number of display widgets, including basic graphics (labels, rectangles,..), widgets that monitor the value of a PV (text updates, LED-type indicators, meters, plots, …), widgets to control the value of a PV (text entry, slider, buttons, …), and widgets to aid in the layout and organization of displays (embedded displays, groups, tabs, …). Widgets have been designed such that they can ideally be used “as is” with little customization. For example, a “Text Update” widget only needs to be configured with a PV name to then display the value of that PV combined with units, appropriate numeric precision, and alarms will be indicated via an alarm-sensitive border. If necessary, many visual and behavioural aspects of each widget can be adjusted via a multitude of widget properties. There is also a scripting interface which offers access to the complete Phoebus API, but its use is reserved for advanced users who are familiar with the API and who are also prepared to then maintain these scripts.

The tool can auto-convert displays from other EPICS display tools (EDM, MEDM, BOY), and the converted displays tend to require very little manual adjustments as long as the original displays also contained minimal customizations. The XML-based “*.bob” file format of the Display Builder was designed to me minimalistic and generic, anticipating the need to eventually use them with another generation of display tools, since user interface tools continue to evolve over time. Many “*.bob” files and in fact already be represented in a separately developed web runtime [4].

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUSDSC08

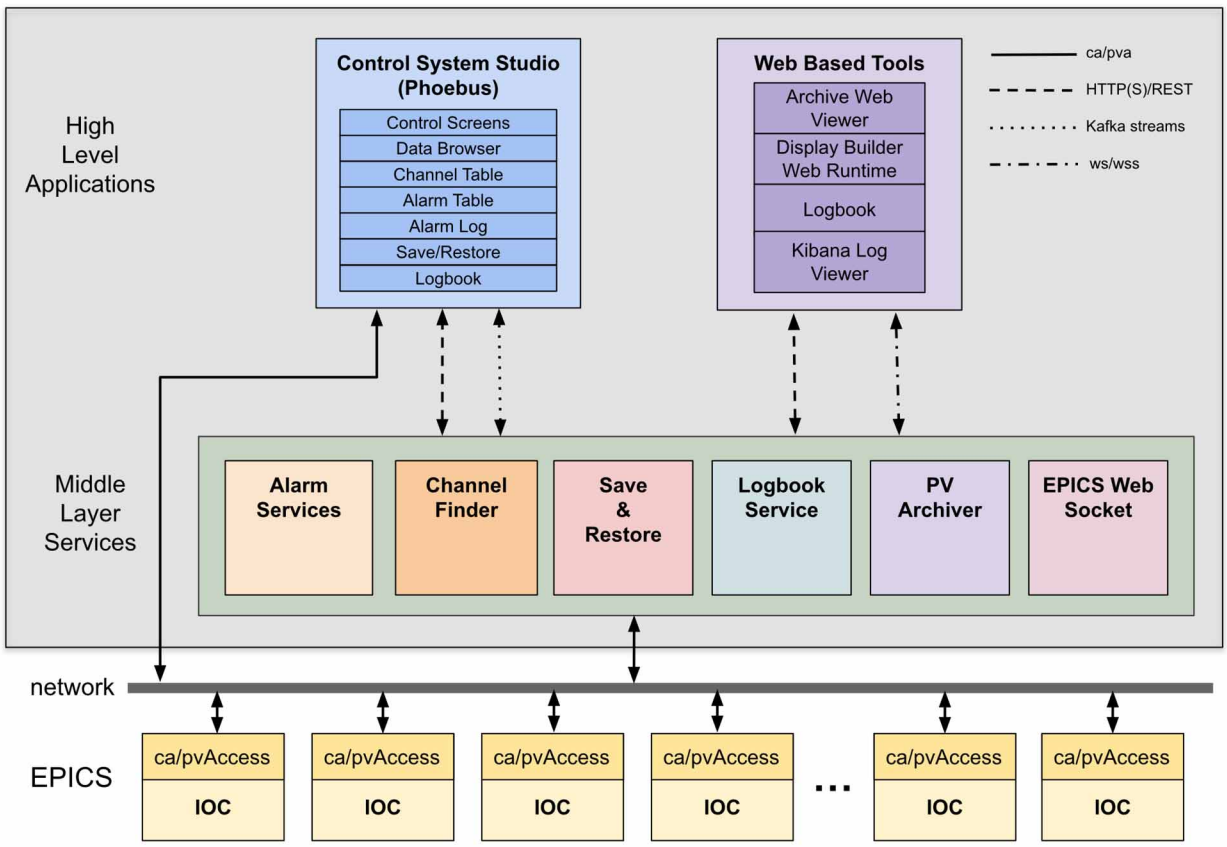

Figure 2: Phoebus Tools and Services Software Ecosystem Overview.

Data Browser

The Data Browser is a trending tool for plotting the values of PVs over time. It supports SPI-based archive data providers, currently including the EPICS Archive Appliance as well as RDB-based history stores and TimescaleDB.

PV Utilities

Several tools support the introspection of control system PVs. Probe is a tool that displays all the information provided by a PV. The PV Table displays value, time stamp and alarm information for a list of PVs, and offers a basic save, compare, restore functionality. The PV Tree can display the hierarchy of input links in EPICS records.

Alarm User Interfaces

Every EPICS PV provides alarm information, and most client tools will by default indicate the alarm state via for example an alarm-sensitive border in live displays. A selected number of alarms, however, should be brought to operator attention even if they are not on a currently open display because these alarms can assist operators in their task of efficiently operating the facility [5]. The alarm user interface includes an "Area Panel" high-level overview of the overall alarm state, an "Alarm Table" list of current alarms, an "Annunciator" for voice representation of new alarms, and an "Alarm Tree" for viewing and editing the hierarchical alarm configuration.

Channel Finder Clients

Valid PV names tend to be difficult to memorize for multiple reasons. To help users to uniquely determine a valid PV name, the Channel Finder clients provide a case-insensitive search tool using substrings of the full name. Search results may include additional meta-data maintained in the Channel Finder service, like IOC name, IOC hostname, address and port, current PV status, record type and IOC developer identity.

Save and Restore User Interface

The save-and-restore application is a general-purpose tool providing means to save a user-defined set of PV values to a remote service. Such snapshot data can at a later point be retrieved from the service and written back in order to restore a subsystem to a known state, or to quickly switch configurations. Users may change the PV values of a snapshot (e.g., using a scaling factor) prior to the restore operation. Multiple snapshots can be combined to facilitate a restore operation on a larger set of PVs.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUSDSC08

Logbook

Several sites use an online logbook to record operational issues and progress. While several sites have adopted the Olog system [6], others rely on a site-specific tool. Phoebus contains a pluggable framework which allows applications to create and view log entries to several different implementations of Logbook services.

PHOEBUS MIDDLE LAYER SERVICES

These services adhere to the principles of the microservices architecture, exhibiting modularity with each service possessing a clearly defined scope and service APIs. This approach enables the creation of lightweight, specialized services, avoiding the need for complex, resource-intensive ones. Clients within the Phoebus framework can readily leverage this integrated ecosystem to deliver users a smoother and more seamless experience.

Alarm Services

Alarm Server The alarm server provides the core alarm system functionality of monitoring a configurable list of PVs and notifying operators via the alarm user interface of new, pending and acknowledged alarms.

Alarm Logger In the Alarm Logger, data can serve a dual purpose: it not only provides a historical timeline of alarms and associated operator actions, like acknowledgements, but also acts as a foundation for generating valuable statistics. These statistics are instrumental in identifying noisy alarms or establishing correlations among multiple alarms over time. This chronological information is pivotal for optimizing the alarm configuration, enabling more efficient and effective alarm management.

Alarm Configuration Manager The alarm configuration manager keeps a history of changes to the alarm configuration, allowing system administrators to track changes and optionally restore older configuration snapshots.

Phoebus Save and Restore

Snapshots and other save-and-restore data objects are persisted by this service, offering a REST API over HTTP(S). Backed by Elasticsearch it offers powerful and efficient search capabilities to locate snapshots based on meta-data such as snapshot name, date, user identity and user-defined tags.

Phoebus Olog

In its most recent incarnation, Phoebus Olog is an online logbook service which relies on Elasticsearch for persistence of log entries, and Mongodb for attachments. Clients may create and access data through a REST API, which includes search based on meta-data and natural language search on the contents of the log entry.

The logbook service offers a range of clients to cater to various user needs. These include an integrated client within the Phoebus framework, ensuring seamless access from other Phoebus applications. Additionally, there is an HTML/JavaScript client designed for web-based access, enhancing accessibility. Furthermore, dedicated clients with specific functionality are available for smartphones, delivering optimized experiences on mobile devices.

Channel Finder

The "Channel Finder" is a simple directory service [7]. Its core purpose is to address the challenge of organizing and accessing channel names within the flat namespace of the EPICS Channel Access protocol. One of its standout features is its "query-by-functionality" approach, allowing users to efficiently locate channels based on specific criteria. Moreover, with the use of tags and properties, Channel Finder enabled the construction of hierarchical views within the EPICS name space. With applications spanning from high-level physics to data management and more, Channel Finder plays a vital role in enhancing the accessibility, flexibility, and efficiency of EPICS control system operations.

NEXT STEPS AND FUTURE PLANS

Backward and Forward Compatibility The framework is designed to embrace both backward and forward compatibility, acknowledging the ever-evolving landscape of software technology and computer graphics.

Migration Path and Backward Compatibility

The framework aims to provide backward compatibility, recognizing the importance of preserving legacy content and ensuring a smooth transition for users [8]. To achieve this, the Display Builder allows for the import of file formats from older EPICS display tools. This approach guarantees that content created using previous tools remains accessible and functional within the new framework.

Forward Compatibility and Adaptability

Anticipating future control system tools and technological advancements, the framework introduces new file formats with a minimalistic and generic design. These formats are purposefully crafted to simplify the adoption of upcoming tools while maintaining compatibility. Moreover, the middle layer services are underpinned by simple, technology-agnostic APIs. This design choice extends the longevity of client support. Even in cases where significant backend changes are made, such as transitioning storage solutions, Phoebus clients remain unaffected. This commitment to forward compatibility ensures that the framework can seamlessly adapt to changes and innovations in the control system landscape.

Track EPICS7 Upgrades and New Features

The current Java implementation of the PV Access protocol (core-pva) that has been developed within the Phoebus project continues to track the ongoing EPICS 7 changes. For example, we recently added support for IPv6 and are involved in adding support for secure communication via SSL/TLS sockets.

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-TUSDSC08

CONCLUSION

Phoebus is the culmination of decades of experience and expertise derived from the CS-Studio collaboration. It's rooted in a profound understanding of the control system community's needs.

This comprehensive ecosystem, as seen in Fig. 2, of control system tools empowers operators, engineers, and scientist to streamline workflows. From alarm detection to inspections of alarm details, viewing historical data of associated PVs, issue resolution, and meticulous documentation, Phoebus provides an all-encompassing solution.

ACKNOWLEDGEMENTS

The Phoebus update, initiated in 2017, represents the evolution of CS-Studio, a project that has been in development since 2006. Throughout this extensive timeframe, numerous individuals have made significant contributions to its growth and success. While the authors of this paper represent only some of the currently active developers, we extend our sincere thanks to the many others who have contributed.

This prolonged journey since 2006 underscores the enduring nature of this project. Rather than allowing CSS/Eclipse to fade away, we have undertaken the task of porting it to a new platform which has been designed with adaptability in mind. This commitment to longevity ensures that the project remains robust, adaptable, and ready for the future.

REFERENCES

[1] EPICS Base releases, https://epics-controls.org/resources-and-support/base/epics-7/

[2] Phoebus, http://www.phoebus.org

[3] K.-U. Kasemir and M. L. Grodowitz, "CS-Studio Display Builder", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017, pp. 1978-1981. doi:10.18429/JACoW-ICALEPCS2017-THSH303

[4] Display Builder Web Runtime, https://github.com/ornl-epics/dbwr/

[5] K.-U. Kasemir, "CS-Studio Alarm System Based on Kafka", in *Proc. ICALEPCS'19*, New York, NY, USA, Oct. 2019, pp. 1511. doi:10.18429/JACoW-ICALEPCS2019-WESH2001

[6] K. Shroff, and G. Weiss, "Phoebus Olog", EPICS Spring Collaboration Meeting 2021

[7] K. Shroff, K. Kasemir, T. Ford, M. Davidsaver, G. Weiss, and R. Lange, "ChannelFinder", EPICS Spring Collaboration Meeting 2023

[8] T. Ashwarya *et al.*, "Upgrading and Adapting to CS-Studio Phoebus at Facility for Rare Isotope Beams", presented at ICALEPCS 2023, Cape Town, South Africa, Oct. 2023, paper TUMBCMO11, this conference