ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-THPV013

WRAP – A WEB-BASED RAPID APPLICATION DEVELOPMENT FRAMEWORK FOR CERN'S CONTROLS INFRASTRUCTURE

E. Galatas[*], A. Asko[†], E. Matli[‡], C. Roderick[§], CERN, Geneva, Switzerland

Abstract

To ensure stable operation of CERN's accelerator complex, many Devices need to be controlled. To meet this need, over 500 custom Graphical User Interfaces (GUI) have been developed using Java Swing, Java FX, NetBeans, Eclipse SWT, etc. These represent a high maintenance cost, particularly considering the global evolution of the GUI technology landscape. The new Web-based Rapid Application Platform (WRAP) provides a centralized, zero-code, drag-n-drop means of GUI creation. It aims to replace a significant percentage of existing GUIs and ease new developments. Integration with the Controls Configuration Service (CCS) provides rich infrastructure metadata to support application configuration, whilst following the associated equipment lifecycle (e.g. renames, upgrades, dismantling). Leveraging the CERN Accelerator Logging Service (NXCALS) and the Unified Controls Acquisition and Processing (UCAP) platform, allows WRAP users to respectively, create GUIs showing historical data, and interface with complex data-stream processing. The plugin architecture will allow teams to further extend the tool as needed. This paper describes the WRAP architecture, design, status, and outlook.

INTRODUCTION

Over the past decade, a large number of expert applications have been developed to provide a way of controlling and monitoring thousands of Devices present within the CERN accelerator complex. The need for control and data visualization applies not only to production equipment, but is also essential for developing and testing new Devices. However, the ecosystem of technologies and platforms used for the development of such applications is quite fragmented. A lot of applications evolved organically based on individual needs. At the same time, the desktop graphical application tool kits traditionally used at CERN have evolved, whilst in parallel facing a major decline of community interest over the years. The Java Swing toolkit has been used since the late 90's, with Java FX coming on the scene several years ago. Most recently, given the direction of Oracle's support for Java as a graphical user interface solution, PyQt has been adopted for some graphical applications [1]. Unsurprisingly, the proliferation of custom Graphical User Interfaces (GUI) combined with an relatively rapid evolution of GUI technologies has lead to a worrying situation backed by extensive technical debt.

Instead of forcing a large and diverse community at CERN to continue learning new GUI technologies and develop their own applications, the following question was raised: "Can we turn the situation around and provide a central, data-driven GUI platform, which experts can use to configure their applications, based on their domain knowledge, without worrying about how to develop, build, deploy, and maintain it, using technologies which will undoubtedly continue to evolve?". A new Web-based Rapid Application Development Framework (WRAP) was designed as the answer to this question, and aims to replace many of the legacy applications over time. WRAP will provide a centralized solution around a common platform, integrated with core Controls sub-systems such as the Accelerator Logging Service (NXCALS) [2] and Controls Configuration Service (CCS) [3]. Since the inception of this project comes after years of application development in the organization, there is a large data-set of use-cases, feedback, and design decisions to analyse and improve upon. Leveraging this knowledge can shape WRAP into a single, unified solution for the vast majority of use-cases.

PROJECT GOALS

There are currently over 500 custom Graphical User Interfaces (GUI) relating to Device control and corresponding to different needs of Operation. Since WRAP encompasses many diverse use-cases and clients, it is important to have a clearly defined set of goals, based on which, decisions can be made regarding individual features.

The most vital attribute WRAP must exhibit, is ease of use. An intuitive User Interface (UI) is required, and beyond improving user productivity, it must encapsulate the complexity of Device modeling and communication. Wherever possible no-code, drag-and-drop configuration should be preferred, allowing experts without a programming background to work at a higher level. This level of abstraction must not, however, come at the cost of performance. Many parallel, real time visualizations should be supported.

Being a centralized service concentrates a lot of complexity on the platform itself, creating a barrier that may inhibit contributions from other teams. To offset this, contributions will instead be supported through a plugin architecture. A stable public application programming interface (API) will be made available giving access to the core functionalities of the platform. Those must in turn have high test coverage and be very conservatively altered.

Lastly, WRAP should leverage device metadata from the CCS to the highest possible degree. Not only does this remove redundant configuration steps, but also enables the restriction of configuration options to only those that are compatible with any given Device. This will help users to configure applications within WRAP whilst avoiding configuration compatibility errors that typically stem from a lack

[*] epameinondas.galatas@cern.ch
[†] anti.asko@cern.ch
[‡] aemnuele.matli@cern.ch
[§] chris.roderick@cern.ch

of detailed knowledge of the underling Device interfaces and behaviour.

ARCHITECTURE OVERVIEW

The WRAP architecture (Fig. 1) needs to accommodate the goals mentioned above, whilst also ensuring that it can scale in terms of functionality, complexity and performance. A clear separation between the rendering logic and underlying data management has to exist, while delegating some key points of complexity to relevant CERN Controls subsystems.

The platform's front-end (user interface layer), is created using web technologies rather than as a native desktop application. Running in a web browser removes cross-platform compatibility concerns, simplifies distribution and updates, and facilitates use from mobile devices. JavaScript's ever improving speed also removes the need for performance trade-offs in order to facilitate the WRAP platform use cases. Thanks to its immense popularity, the web stack appeals to a lot of potential candidates applying to CERN, it provides mature frameworks and UI design systems, and incorporates comprehensive debugging tools. A high degree of flexibility is also present; while the platform and most graphical widgets are being developed in the Angular framework, 3rd party widgets can utilize any preferred solution. These widgets can then be bundled and integrated as native Web Components. Web Components can encapsulate presentation and behaviour in a custom HTML element, creating the basis of a plugin architecture. Each widget has a specific, predefined structure that is modeled in the WRAP database. This model is used to validate widget states and to provide constraints on what types of data can be visualized.

In order to provide actual Device data to be rendered and processed, a variety of other Controls sub-systems are used. The information about the available Devices and their data structures is managed within the Controls Configuration Service (CCS) [3] from where it is served to WRAP via a REST API. After the desired Devices and their Properties are identified, the WRAP back-end subscribes to live updates of data values, using the CERN Java API for Parameter Control (JAPC) [4]. As values become available, they are aggregated and propagated from the WRAP server to corresponding WRAP clients via HTTP/2 push events.

Device Evolution

As operational needs or underlying hardware evolve, corresponding Controls software Devices follow. This can mean a simple change of Device name or a more complex change of Device interface. A requirement for WRAP is to be resilient to such events by design. This is achieved by integrating with the CCS and it's Controls Configuration Data Lifecycle (CCDL) manager [5]. Knowing the dependencies between Device data and WRAP widgets, throughout all applications configured within WRAP allows a WRAP API to be exposed to the CCDL. Using this API, the CCDL can interrogate WRAP to check for compatibility of CCS changes to Device configurations and, in most cases propagate the changes to WRAP - automatically keeping the applications working over time. Very complex changes such as data type alterations or Device deletions will at least result in notifications to the relevant application owners in order that they can make a manual intervention.

Historical Data

The architecture explained so far is able to visualize live Device data, however historical values are also important. For example, a graph widget would require data for the displayed time window to be pre-filled, before receiving updates based on current values. The visualisation of historical data is possible thanks to the integration with the CERN Accelerator Logging Service (NXCALS) [2], providing a unified solution for logging of time-series data acquired from accelerator Controls Devices.

Data Processing

For some use cases, Device data requires some form of processing before being displayed. This processing can range from simple filtering of specific values to aggregations that involve a large number of data subscriptions and require complex computations. The Unified Controls Acquisition and Processing (UCAP) [6] framework provides a means to facilitate and streamline acquisition and processing of Controls data. Data from multiple Devices are processed with pre-existing or user defined Python or Java code, with incoming values aggregated based on the desired event sourcing strategy. The result is then exposed as a Virtual Device, for which WRAP can utilize the standard infrastructure explained above to access Device metadata, live data and historical values.

Fault Tolerance

Given the foreseen critical nature of WRAP as a service (e.g. exposing critical GUI applications), it must have a high degree of fault tolerance. The WRAP front-end is received by the client as a Single Page Application (SPA). When loading a user application instance within WRAP, metadata has to be fetched in order to: render the included widgets, establish data source subscriptions, and fill historical values where necessary. Once initialized, WRAP applications only depend on the back-end infrastructure to exchange values, for which the corresponding server nodes are highly redundant. In the event of failure, such as a Device disconnection, wrong metadata, missing history, etc. parts of the application can fail gracefully, providing appropriate information to the user, without breaking the application as a whole.

CURRENT STATUS

WRAP development has now entered its second year. The processes for structuring and saving user applications have stabilized, and development is focused on adding editor features and extending the widget library. In this section, some

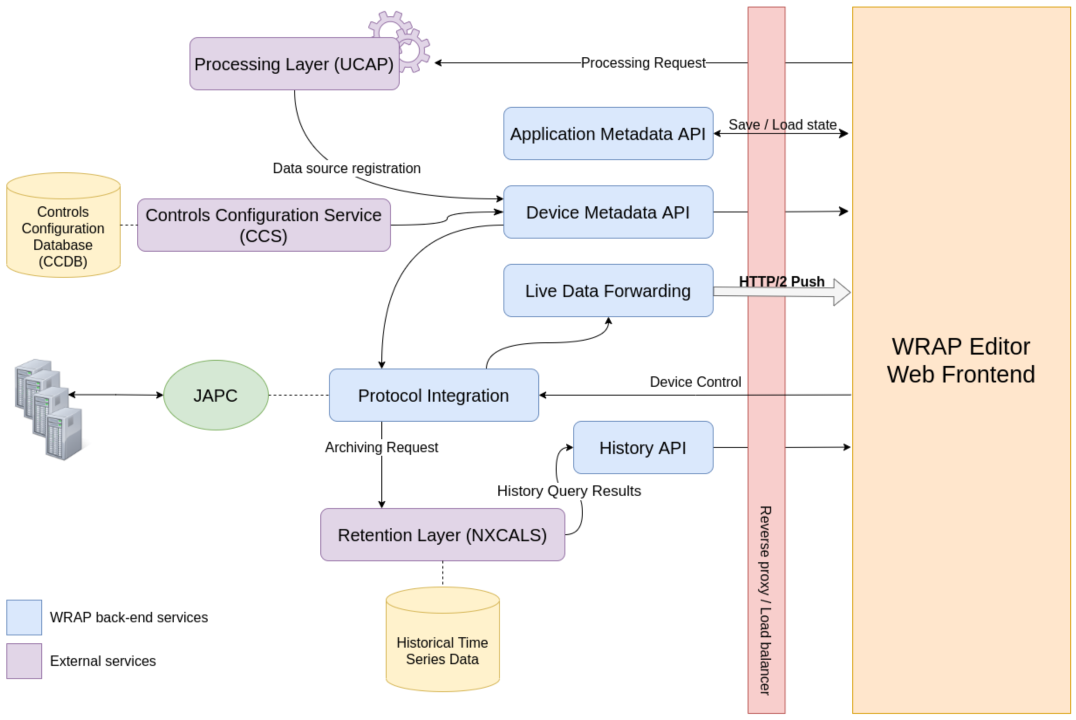


Figure 1: WRAP architecture overview.

of the existing capabilities will be highlighted coupled with some examples of finished products.

Figure 2 shows the WRAP editor. On the top, a tab-like list of recently visited applications is present. These tabs allow a fast navigation between applications. They also include information about the application title and owner, as well as an option to make a new copy - helpful when starting a new application based on an existing one. When an application owner enables the edit mode, a blue bar appears below with options to alter the application name and description, make it publicly visible, undo/redo functionality, and an option to enable a scroll-able / report-like view (instead of being limited to one screen).

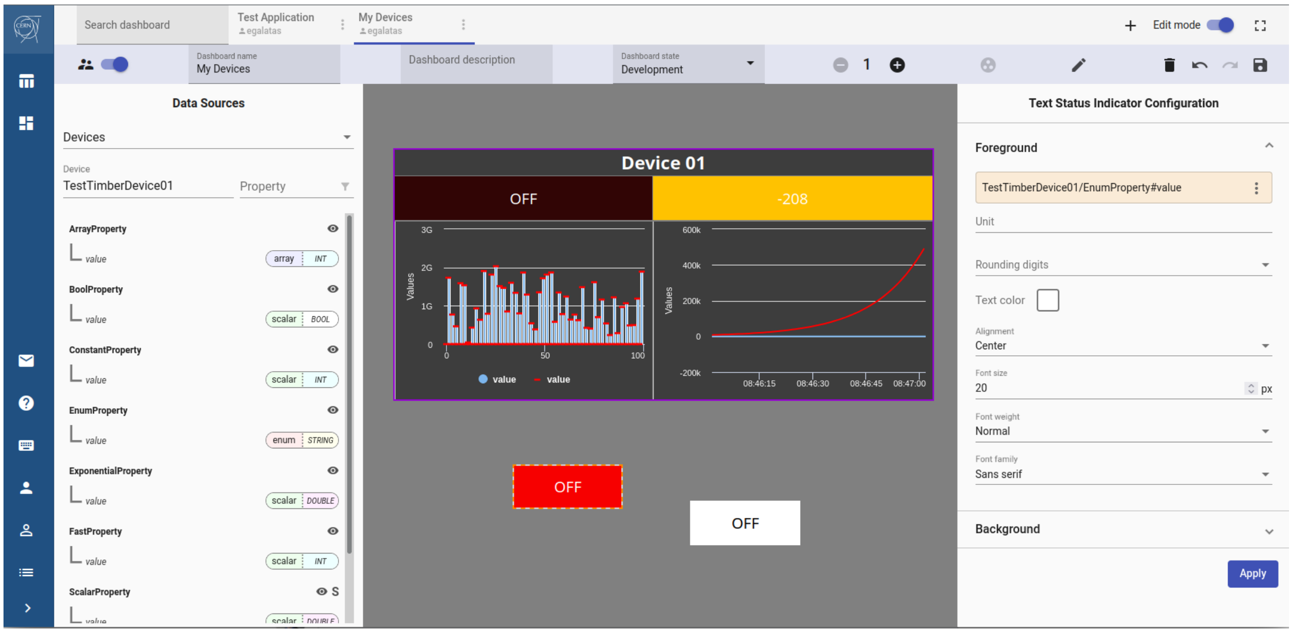
On the left side is the Entry Point menu which allows the lookup of Devices (with some additional filters). Once a Device has been selected, a list of it's Properties appear, indicating the data type and additional flags dictating its behaviour. A Property value can either be used to instantiate a new widget that references it, or it can be assigned to an existing widget. In both cases, constraints are applied based on the correlation between the Property value type and corresponding widgets (e.g. boolean, string or enum can be shown on status indicators, numbers may be plotted on a chart, etc.).

Selecting a widget on the central application canvas will open a panel on the right side, from where the related attributes can be modified. Widgets can also be grouped together, with the resulting group having a union of the aspects of its children (Device source, colours, relative dimensions, etc.). This effectively enables bulk copying, moving and editing of related widgets. The bulk features around groups can make it very efficient to configure applications showing multiple aspects, of multiple Devices, of a similar type. For such cases, the user configures the individual widgets for a single Device, groups them together, copies them, then changes the Device binding in bulk for all widgets in the new group. Currently, the implemented widgets include graphs, labels, and status indicators, with many others foreseen.

ADOPTION

At CERN there are around 60 so-called ”Fixed Displays” or ”Vistars” which are applications that typically display important information about the status of accelerator operations and specific systems. These critical applications, are based on an aging custom framework, legacy Java Swing technology, and domain specific code. They are under the responsibility of a diverse group of people, who are only involved with software as an occasional, part-time activity, typically during periods of accelerator shutdown. It is not feasible for such people to learn the latest graphical technologies in their spare time. This is one of several reasons that it is foreseen to migrate all of these Fixed Display applications to the WRAP platform, thus lowering the technical debt and limiting the responsibility of the people associated with the applications to the domain knowledge and not the software maintenance.

A progressive migration of the Fixed Displays is already underway. Figure 3 shows one such application related to the LHC experiments. The applications created so far, aggregate data from multiple Devices via UCAP, and then visualize the resulting Virtual Device data from within WRAP. Currently, an effort is placed on evaluating how these new applications behave in the operational environment, while additionally



Dashboard name: My Devices
Dashboard description
Dashboard state: Development

Data Sources

Devices
Device: TestTimberDevice01 Property

ArrayProperty		
value	array	INT
BoolProperty		
value	scalar	BOOL
ConstantProperty		
value	scalar	INT
EnumProperty		
value	enum	STRING
ExponentialProperty		
value	scalar	DOUBLE
FastProperty		
value	scalar	INT
ScalarProperty		
value	scalar	DOUBLE

Text Status Indicator Configuration

Foreground
TestTimberDevice01/EnumProperty#value
Unit
Rounding digits
Text color
Alignment: Center
Font size: 20 px
Font weight: Normal
Font family: Sans serif
Background
Apply

Figure 2: WRAP application builder.

evaluating possible cases of performance degradation or improvements with respect to the legacy ones. In order to give a desktop like experience, it is foreseen to evaluate technologies such as Electron [7], in order to avoid launching the WRAP applications via a full-blown web browser.

MAJOR CHALLENGES

As a system that combines many different services and provides solutions for very diverse use cases, WRAP presents a number of technical challenges that have to be addressed. Some of the most notable challenges currently experienced when developing WRAP are listed below.

Scalability and Performance

CERN's accelerator complex consists of hundreds of thousands of Devices, some of which emit very large data sets at very high frequencies. Creating visualizations that can cope with such cases requires optimizations both on data delivery and front-end (user interface) performance. Any inefficiency in either, however small, will be magnified and create bottlenecks in sufficiently complex user applications. A number of strategies have been employed to counter this, such as avoiding duplicate data subscriptions and bundling rendering events. Despite this, thorough regression testing will have to be incorporated for all widgets and major editor features in order to guarantee consistent performance across subsequent WRAP builds.

Evolving Technologies

Achieving stability on an evolving technological landscape is a challenge faced in all software. WRAP is particularly exposed to this problem, as it has to follow the evolution of not only 3rd-party front-end web frameworks and back-end technologies, but also the CERN Controls sub-systems with which it interacts. A consistent and well documented API between different parts of the platform, and a modular design across the entire stack, helps alleviate this problem. Even if breaking changes cannot be avoided, affected parts can be addressed separately, containing the scope of any potential refactoring and improving the accuracy of related cost estimations.

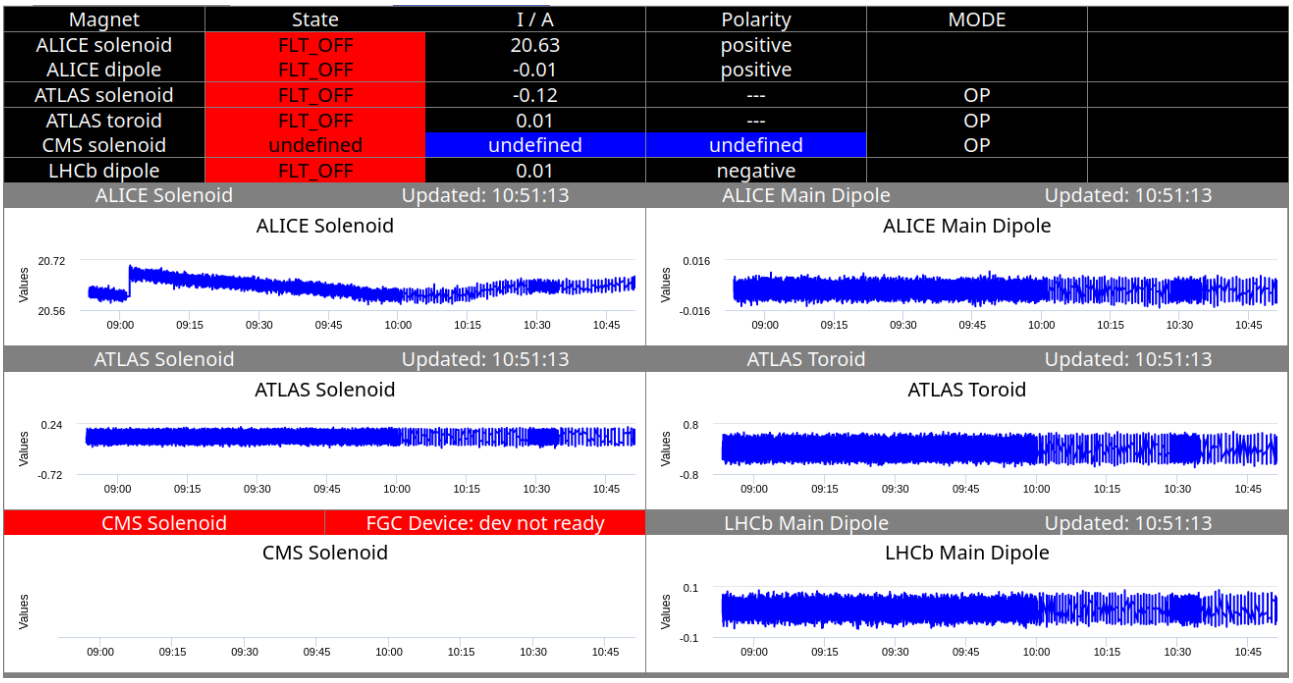
Web Limitations

As mentioned above, the web as an application run-time provides a number of advantages with respect to native applications, including efficient development times, easier debugging (especially around the visual elements), and a trivial distribution of the final product. There are however, some trade-offs that have to be considered.

The performance of complex applications on aging hardware can degrade the user experience. For WRAP this problem will become more apparent as more features are requested and more sophisticated applications are created. Solutions like Web Assembly and Web Workers have been considered and seem promising in eliminating these problems in the future.

Web applications might sometimes be limited due their inability to access native desktop features. They are also perceived differently by end-users, when faced with loading and running in a browser, instead of being installed as a standalone application available via a shortcut. However, all these point can be addressed by bundling WRAP with Electron [7] allowing the same behaviour and access to native APIs as a traditional desktop application. This also enables individual applications created within WRAP to be accessible via distinct shortcuts.

Finally, some problems are inherently more difficult to solve on the Web compared to native applications. For example, installation and communication across multiple windows, while possible, is a lot more involved. Creating multi-



Magnet	State	I / A	Polarity	MODE	
ALICE solenoid	FLT_OFF	20.63	positive		
ALICE dipole	FLT_OFF	-0.01	positive		
ATLAS solenoid	FLT_OFF	-0.12	---	OP	
ATLAS toroid	FLT_OFF	0.01	---	OP	
CMS solenoid	undefined	undefined	undefined	OP	
LHCb dipole	FLT_OFF	0.01	negative		

ALICE Solenoid Updated: 10:51:13
ALICE Main Dipole Updated: 10:51:13
ATLAS Solenoid Updated: 10:51:13
ATLAS Toroid Updated: 10:51:13
CMS Solenoid FGC Device: dev not ready
LHCb Main Dipole Updated: 10:51:13

Figure 3: Example LHC Fixed Display implemented in WRAP.

windowed applications via WRAP is presenting a significant challenge. However, the easier development of most of the other parts of WRAP on the Web does, so far, greatly offset the small subset of features that prove problematic. Choosing any technology to build a sufficiently complex software will invariably highlight some deficiencies in some aspects of it.

NEXT STEPS AND FUTURE OUTLOOK

The aim is to use WRAP to replace as many as possible of the hundreds of CERN accelerator controls standalone graphical applications. From the outset, it is clear that WRAP, which is a data-driven application, will not be suited to replace the most complex legacy applications. Nevertheless, based on an in-depth analysis performed in 2020, it is expected that between 40-70 percent will be suited to WRAP. This will clearly require a lot of features to be present in the WRAP editor together with significant expansion to the widget collection. It is expected that the gradual adoption of WRAP by a large community will help establish collaborations with other teams (certainly inside CERN, and hopefully outside), which in turn can contribute improvements, extensions or even just valuable feedback to help WRAP become a more complete and useful end-product.

SUMMARY

WRAP introduces a new paradigm of developing applications for control and visualization of Device data. The new platform has received a lot of interest from the user community and, while development is at an early stage relative to the ambitious goals that where initially set, early adopters have shown promising results on the long-term viability of the solution. The overall architecture, consisting of modern and tested industry practices, coupled with the use of open-source software used by large communities, will help ensure the stability of the overall platform. WRAP is on-track towards it's objectives and it is expected that this endeavour will pay off, helping to substantially reduce technical debt, and minimising the need for diverse experts to acquire new software application development skills, in order to develop and maintain their graphical applications.

REFERENCES

[1] Z. Kovari *et al.*, "New Timing Sequencer Application in Python with Qt - Development Workflow and Lessons Learnt", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper THPV015, this conference.

[2] J. Wozniak *et al.*, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", in *Proc. ICALEPCS'19, New York, USA*, Oct. 2019. doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[3] L. Burdzanowski *et al.*, "CERN Controls Configuration Service — A challenge in usability", in *Proc. ICALEPCS'17*, Barcelona, Spain, Oct. 2017. doi:10.18429/JACoW-ICALEPCS2017-TUBPL01

[4] V. Baggiolini *et al.*, "JAPC - the Java API for Parameter Control", in *Proc. ICALEPCS'05*, Geneva, Switzerland, Oct. 2005.

[5] B. Urbaniec, "CERN Controls Configuration Service – Event-Based Processing of Controls Changes", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper MOPV043, this conference.

[6] L. Cseppentő *et al.*, "UCAP: A Framework for Accelerator Controls Data Processing at CERN", presented at ICALEPCS'21, Shanghai, China, Oct. 2021, paper MOPV039, this conference.

[7] Electron, https://www.electronjs.org