ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-MOPV043

CERN CONTROLS CONFIGURATION SERVICE – EVENT-BASED PROCESSING OF CONTROLS CHANGES

B. Urbaniec, L.Burdzanowski, CERN, Geneva, Switzerland

Abstract

The Controls Configuration Service (CCS) is a core component of the data-driven Control System at CERN. Built around a central database, the CCS provides a range of client APIs and graphical user interfaces (GUI) to enable efficient and user-friendly configuration of Controls. As the entry point for all the modifications to Controls system configurations, the CCS provides the means to ensure global data coherency and propagation of changes across the distributed Controls sub-systems and services. With the aim of achieving global data coherency in the most efficient manner, the need for an advanced data integrator emerged.

The "Controls Configuration Data Lifecycle manager" (CCDL) is the core integration bridge between the distributed Controls sub-systems. It aims to ensure consistent, reliable, and efficient exchange of information and triggering of workflow actions based on events representing Controls configuration changes. The CCDL implements and incorporates cutting-edge technologies used successfully in the IT industry. This paper describes the CCDL architecture, design and technology choices made, as well as the tangible benefits that have been realised since its introduction.

INTRODUCTION

The Controls Configuration Service (CCS) is a core component of CERN's Control system, serving as a central point for the configuration of all Controls sub-domains. CCS ensures that the data provided to other services is done in a coherent and consistent way. CCS is used by a diversified group of users, including installation teams (configuring Controls hardware), equipment experts (configuring processes and applications), and Accelerator operators. All CCS users interact with the service at various points in time, to verify or define appropriate configurations.

The service is built around a centralised Oracle database server. To minimise downtime of the system and risks of negative impact to the users, the server is deployed in a cluster as 2 redundant nodes, providing 99.9% availability. The data stored in the CCS database (CCDB) may be accessed via a dedicated high level web-based editor - Controls Configuration Data Editor (CCDE) [1]. At the same time the service also provides advanced Java and Python REST APIs which allow users to efficiently configure, modify and maintain configuration data in a programmatic way.

The CCS service has been an integral part of the Controls system for many years. The first version was created in the late 80s, during operations of the Large Hadron Collider's predecessor - The Large Electron-Positron Collider (LEP). Since that time the service has evolved and been consolidated multiple times. The last renovation started 4 years ago [2] to match the CCS technology stack with technologies widely used in the software industry. All CCS components are based on Java (currently version 11) and the Spring framework. From the Spring framework, Spring Boot has been selected as a solution to establish a common architecture among all the applications in a simplified and unified way. As mentioned, CCS also provides a high-level web interface - CCDE. The CCDE is based on the AngularJS framework (provided by Google) and is augmented with a web components framework developed in-house and encapsulating common functionality and integration with CERN services such as SSO. Communication between the Java back-end and the AngularJS front-end is implemented using the REST architectural pattern.

CERN CONTROLS CONFIGURATION SERVICE

As a core Controls service, the CCS must exhibit a high level of availability. Even though CCS downtime does not directly impact beam operation, it severely limits the means to verify or modify core system configurations. To provide the highest possible availability and quality of service, each CCS component is implemented with some degree of redundancy. For example service-side processes are stateless and deployed in a multi-node set-up. In the rare case of a failure for one of the nodes, the system remains operational without impacting users. Advanced monitoring and notification mechanisms continuously check the consistency and status of all service components and send alerts in case of any abnormality. This allows service managers to react before the CCS becomes unavailable.

Since its inception, the CCS has evolved regularly, in-synch with the significant evolution of CERN's accelerator complex and multiple sub-systems. During the last 30 years, new advanced accelerators and major upgrades have triggered a need for a more sophisticated and powerful configuration platform. The global Controls architecture is realised as a layered system, reflected in CCS configuration domains, which can be simplified as follows:

- Low-level aspects covering kernel driver configurations and hardware types.
- Front-End Computers (FEC) with their modules (of hardware types, mentioned above), on which,

different acquisition and configuration processes are running.

- Software processes which represent physical Devices deployed in specific Accelerators, together with definitions of their interfaces, known as "Device Classes".
- Role-based authorisation schemes, both for access to the aforementioned Device processes and to high-level applications.
- Configuration of Device data acquisitions and logging, to have data available for further analysis.

CONTROLS CHANGES

Every change to Controls configuration, no matter how small, may impact one or more other Control system components. Due to the overall Controls complexity, manual adaptations in related sub-systems to account for changes, are highly time consuming and error prone, and risk to result in discrepancies between different Controls sub-systems. It is therefore crucial, to capture changes, notify related services, and even trigger specific actions, in an automatic way.

To ensure that changes are accepted and parsed correctly, they are treated as named events which may occur during the configuration lifecycle. Each such event may be acknowledged, analysed, and propagated to appropriate services to trigger necessary actions. Once such operations are complete, the user that triggered the change event should be notified of the outcome such as success or failure (including relevant details). The behaviour described above falls into the software architecture paradigm known as an "Event Driven Architecture" and aims to address all implications provoked by change requests.

The evolution of physical Controls hardware requires corresponding adaptations at various layers of the Control system. One example is the meta-data describing the programmatic interface of the equipment ("Device Classes", mentioned above) which must naturally evolve following the hardware. This process of a Device changing Class is called "migration". It is a critical event in the Controls configuration life-cycle, as it may impact a number of core Controls sub-systems such as LSA[3] (for settings management), NXCALS[4] (for time-series data logging), UCAP (for data processing), and end-user applications. To control this impact and minimise risks of inadvertent changes, the "Controls Configuration Data Lifecycle manager" (CCDL) is used.

A typical example of such a change, would be an evolution of a hardware component, such that the meta-data describing the component's properties needs to be updated with respect to its data type (e.g. from a *scalar integer* to a *double array*). Such a change needs to be properly propagated to other Controls sub-systems to assure operational continuity e.g. consistent application of settings or data acquisition. The window within which such changes are carried out (both in terms of underlying hardware and software) is normally limited to either Accelerator technical stops or shutdown periods.

To ensure the correct propagation of changes within the CCS and across related Controls sub-systems, a sub-component of CCDL, the migration "orchestrator" is employed. The orchestration engine reacts to the named configuration change events and processes them by orchestrating the corresponding changes in all related Controls sub-systems. This processing must be done in a logical order, reflecting how the sub-systems depend on one another. End-users are provided with live insights into the event processing and are also notified about any problems. The outcome of each distinct step in the migration processing chain is recorded as a status event. Status events are also injected back into the processing system for use by the orchestrator for optional conditional processing. This solution is deployed in production and Table 1 shows the number of processed events in 2020.

Table 1: 2020 Events Generated Based on User Actions

TabEvent	Occurrences
Device migration (change of Class)	21k
Device attribute changed	280k
Device added	100k
Device deleted	90k
NXCALS subscription changed	250k
NXCALS subscription added	360k
NXCALS subscription deleted	130k

Considering the long history of the CCS and the need to cope with the complexity of the Accelerator Controls domain, the CCS architecture and design are tailored to solve a wide variety of possible problems in a configurable and data-driven manner. Below, the core CCS components are described, together with specific examples of an event-based definition of business use-cases as seen from the perspective of Controls users.

ARCHITECTURE

The main component of the event-based processing of changes in the system is the CCDL - Controls Configuration Data Lifecycle manager. The manager acts as an orchestration and integration point amongst core CCS components (CCDA – Controls Configuration Data API, CCDE – Controls Configuration Data Editor) and a limited number of clients using direct database access with dedicated schemas or PL/SQL APIs. The CCDL is composed of two main components: CCLC (the Lifecycle manager) and the so-called "integrated Device migrator". Supporting components of the architecture include:

- Apache Kafka – a distributed data store optimised for ingesting and processing streaming data in real-time.
- Oracle AQ (Advanced Queuing) – a proprietary feature of Oracle databases, which delivers database-integrated message queuing functionality.
- System events – representing configuration changes, are modelled as simple tuple objects, generated at the persistency (database) layer.

System events, modelled as data tuples, consist of meta-data such as operation types, timestamps, aggregation topics (i.e. database transaction ID), as well as the event data (i.e. configuration value before and after the change). The events are designed to be sufficiently rich for the processing system to minimise the need for querying additional data. At the database level the events are stored in log tables, structured with attention to data partitioning and indexing. In normal operations of the system, events are processed continuously which minimises the number of physical reads at the level of the database, thus contributing to overall performance operation of the system. In case when historical events need to be processed, indexing and table partitioning at the level of event type and creation timestamp lowers latency when accessing the data.

The decision to use Apache Kafka was driven by a frequent need for a loosely coupled integration of 3rd party clients with CCS (e.g. other Controls sub-systems). System events are fed to Kafka and made readily available to all subscribed clients, including internal clients like CCLC. This solution enabled a plug-and-play model (see Fig. 1), where external clients may integrate only at the level of Kafka data-objects (key-value pair tuples or JSON objects), or if necessary use a complete CCLC client API, tailored to a specific client domain (e.g. NXCALS or UCAP).

CONCLUSIONS

Before CCDL was put in place, many of the configuration updates described above needed to be performed manually by a limited number of system experts. The manual nature meant that users had to request changes well in advance, such that experts of each of the possibly impacted Controls sub-systems had sufficient time to internally verify what would be the impact, and if necessary, prepare appropriate steps to mitigate risks and ensure a global consistency of the distributed configuration elements. This situation required careful planning and coordination to minimise errors and discrepancies between Controls sub-systems.

The introduction of CCDL helped to considerably reduce support and manual interventions. All changes can now be executed directly by the users, at any moment, without a need for specific actions from the various Controls sub-system experts. Feedback gathered during the last 3 years since the automatic Device migration process has been in place, shows that the work of the users has been simplified and became more efficient. At the same time, thanks to the fully automated process, the time when the global Control system configuration is in an inconsistent state, has been reduced to a minimum.

Reduction of the support and automatization of the configuration process is an example of a continuous evolution of the system. The increasingly dynamic nature of the CERN Accelerator Controls manifested by a growing number of data-driven components, indicates a need for software solutions which in place of customised use-case specific code, rely on more abstract and agile solutions. One possible solution is to deliver a platform which allows users to define complex high-level events based on atomic events generated by the system. While the atomic events represent fundamental changes occurring in the system (e.g. a new Device created), the complex event may be used to indicate patterns of behaviour. For

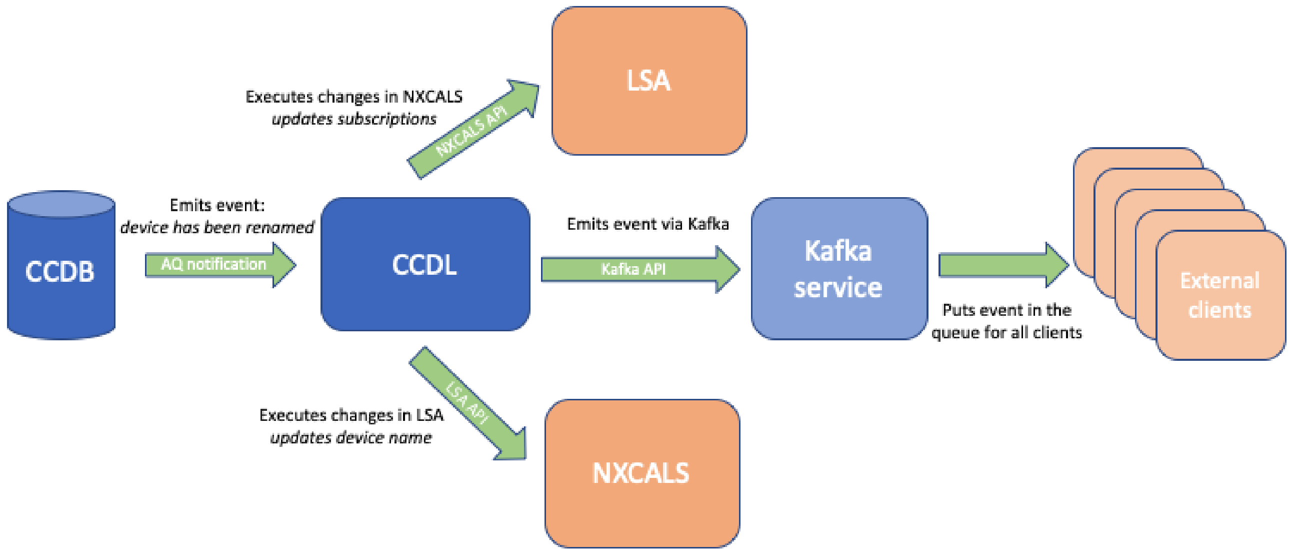

Figure 1: Diagram of event processing with CCDL.

example, a consecutive set of new Device created and Device renamed events may represent a complex event such as swapping of Devices by name while retaining their associated configuration intact. Discovery of such patterns may be achieved with Complex Event Processing engines, for example using Esper CEP framework [5]. The CEP approach provides users with a high degree of freedom in covering different use cases, including dynamic injection of new types of events, without noticeably increasing the amount of support needed from various related Controls sub-system experts.

In the mid-term perspective we plan to integrate ESPER CEP engine with CCDL and consequently expose means to define complex events and EPL (Event Processing Language) queries through a high-level graphical user interface as a part of CCDE. This will provide all the users of the Controls system configuration the means to define new events and give a possibility to look for patterns of events occurring in the system. With such extensions as well as described earlier stream processing engine like Apache Kafka and high-level API like CCDA, we aim to establish practical means to realise a business intelligence solution tailored to the needs of the CERN Accelerator controls and its users, while remaining agnostic to the CERN specific concepts.

REFERENCES

[1] L. Burdzanowski *et al.*, “CERN Controls Configuration Service - a Challenge in Usability”, in *Proc. 16th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'17)*, Barcelona, Spain, Oct. 2017, pp. 159-165. doi:10.18429/JACoW-ICALEPCS2017-TUBPL01

[2] L. Burdzanowski and C. Roderick, “The Renovation of the CERN Controls Configuration Service”, in *Proc. 15th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'15)*, Melbourne, Australia, Oct. 2015, pp. 103-106. doi:10.18429/JACoW-ICALEPCS2015-MOPGF006

[3] D. Jacquet, R. Gorbonosov, and G. Kruk, “LSA - the High Level Application Software of the LHC - and Its Performance During the First Three Years of Operation”, in *Proc. 14th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'13)*, San Francisco, CA, USA, Oct. 2013, paper THPPC058, pp. 1201-1204.

[4] J. P. Wozniak and C. Roderick, “NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service”, presented at *the 17th Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'19),* New York, NY, USA, Oct. 2019, pp. 1465-1469. doi:10.18429/JACoW-ICALEPCS2019-WEPHA163

[5] Esper FAQ – EsperTech, https://www.espertech.com/esper/esper-faq