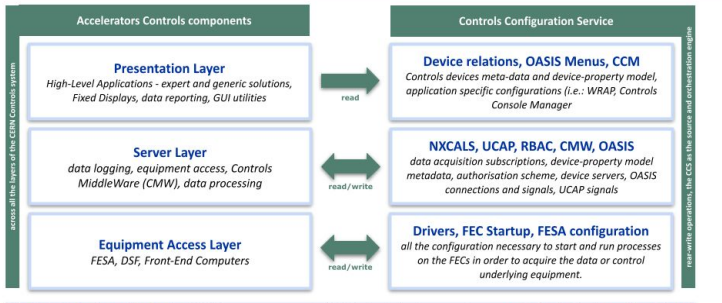
MOPV043

CERN Controls Configuration Service – Event-Based Processing of Controls Changes

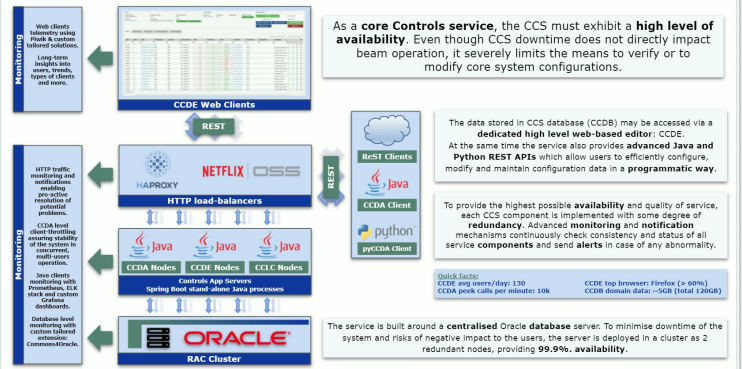
B. Urbaniec †, L.Burdzanowski, CERN, Geneva, Switzerland

ICALEPCS 2021

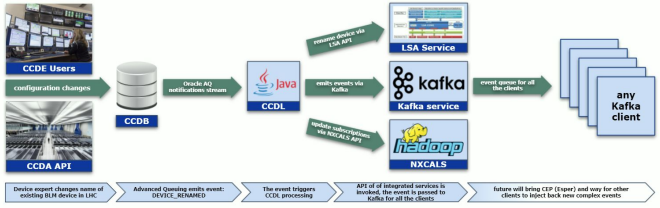
CCS and Controls system configuration



High-level architecture of CCS



Event-based processing of configuration changes



Every change to Controls configuration may **impact** one or more other **Controls system** components. Due to the overall Controls **complexity**, manual adaptations in related subsystems 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.

Event	Occurrences
Device migration (class change)	21k
Device attribute changed	280k
Device added	100k
Device deleted	90k
NXCALS subscription changed	250k
NXCALS subscription added	300k
NXCALS subscription deleted	130k

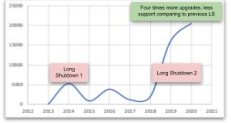
Events produced in 2020

Conclusions and future plans

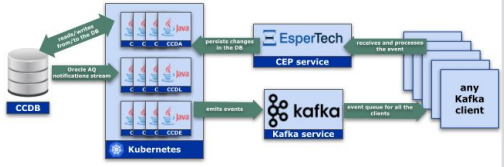
Introduction of Event-Based Processing of Controls Changes allows to:

- **reduce the system support** due to automatisation of processes and better coherency
- **decrease time** when subsystems of Controls system are in **desynchronised state**
- **speed-up propagation** of changes between different parts of Controls system
- **increase system stability** due to automatic validations and simulations before each change
- plug new **services** in **easy** and **coherent** way
- **incorporate user specific logic** without modifications of core components

Evolution of device upgrade events



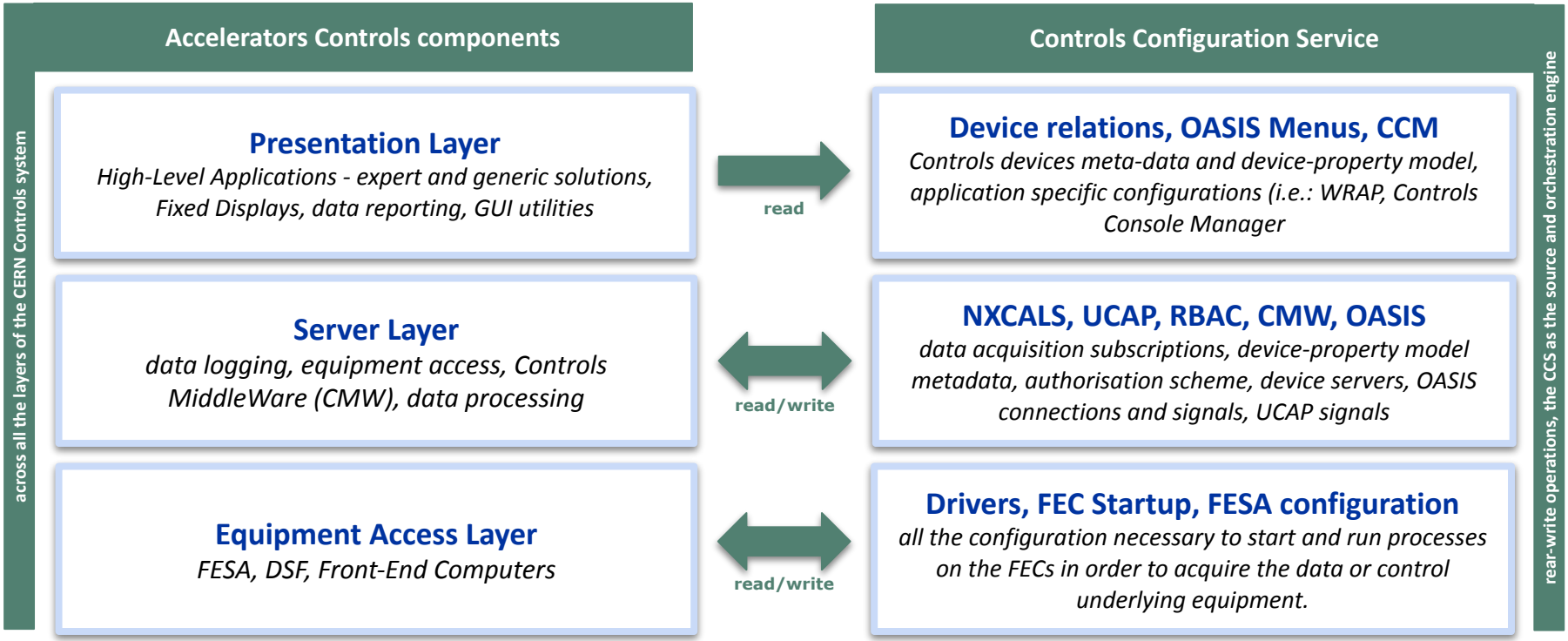
Future architecture



REFERENCES

1. L. Burdzanowski et al, "CERN Controls Configuration Service - a challenge in usability", Barcelona, Spain, 8 - 13 Oct 2017, pp.TUBPL01
2. L. Burdzanowski et al, "The Renovation of the CERN Controls Configuration Service", Melbourne, Australia, 17 - 23 Oct 2015, pp.MOPGF006
3. D. Jacquet et al, "LSA - the High Level Application Software of the LHC - and its Performance During the First Three Years of Operation", San Francisco, CA, USA, 6 - 11 Oct 2013, pp.thppc058
4. J. Wozniak et al, "NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service", New York, United States, 5 - 11 Oct 2019, pp.WEPHA163
5. https://www.espertech.com/esper/esper-faq

CCS and Controls system configuration



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.

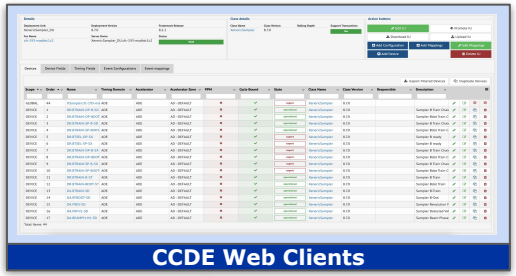
CCS is used by a **diversified group of users**, including installation teams (configuring Controls hardware), equipment experts (configuring processes and applications), and Accelerator operators. The users interact with the service to verify or define appropriate configurations.

High-level architecture of CCS

Monitoring

Web clients telemetry using Piwik & custom tailored solutions.

Long-term insights into users, trends, types of clients and more.



CCDE Web Clients

REST

Monitoring

HTTP traffic monitoring and notifications enabling pro-active resolution of potential problems.

CCDA level client-throttling assuring stability of the system in concurrent, multi-users operation.

Java clients monitoring with Prometheus, ELK stack and custom Grafana dashboards.

Database level monitoring with custom tailored extension: Commons4Oracle.

HAPROXY NETFLIX OSS

HTTP load-balancers

REST

ReST Clients

Java CCDA Client

python pyCCDA Client

Java Java Java

CCDA Nodes | CCDE Nodes | CCLC Nodes

Controls App Servers
Spring Boot stand-alone Java processes

ORACLE

RAC Cluster

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 to modify core system configurations.

The data stored in CCS database (CCDB) may be accessed via a **dedicated high level web-based editor**: CCDE.
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**.

To provide the highest possible **availability** and quality of service, each CCS component is implemented with some degree of **redundancy**. Advanced **monitoring** and **notification** mechanisms continuously check consistency and status of all service **components** and send **alerts** in case of any abnormality.

Quick facts:

CCDE avg users/day: 130
CCDA peek calls per minute: 10k
CCDE top browser: Firefox (> 60%)
CCDB domain data: ~5GB (total 120GB)

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**.

Event-based processing of configuration changes

configuration changes

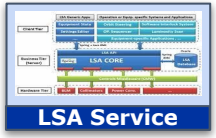
Oracle AQ notifications stream

Java
CCDL

rename device via LSA API

emits events via Kafka

update subscriptions via NXCALS API



kafka
Kafka service

hadoop
NXCALS

event queue for all the clients

any Kafka client

Device expert changes name of existing BLM device in LHC → Advanced Queuing emits event: DEVICE_RENAMED → The event triggers CCDL processing → API of of integrated services is invoked, the event is passed to Kafka for all the clients → future will bring CEP (Esper) and way for other clients to inject back new complex events

Every change to Controls configuration may **impact** one or more other **Controls system** components. Due to the overall Controls **complexity**, manual adaptations in related subsystems 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.

Event	Occurrences
Device migration (class change)	**21k**
Device attribute changed	**280k**
Device added	**100k**
Device deleted	**90k**
NXCALS subscription changed	**250k**
NXCALS subscription added	**360k**
NXCALS subscription deleted	**130k**

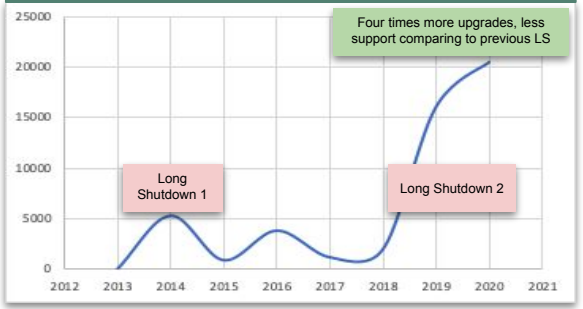
Events produced in 2020

Conclusions and future plans

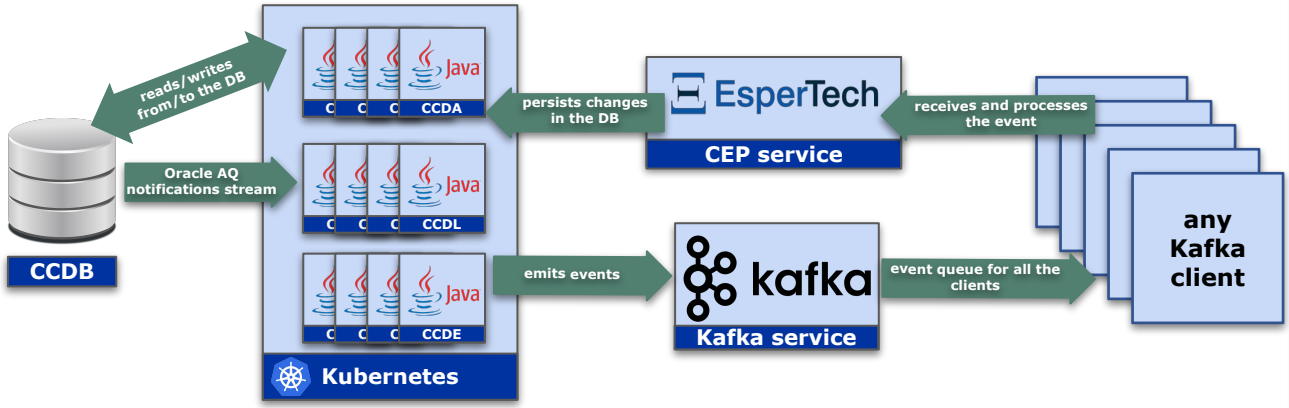
Introduction of Event-Based Processing of Controls Changes allows to:

- **reduce the system support** due to automatisation of processes and better coherency
- **decrease time** when subsystems of Controls system are in **desynchronised state**
- **speed-up propagation** of changes between different parts of Controls system
- **increase system stability** due to automatic validations and simulations before each change
- plug new **services** in **easy** and **coherent** way
- **incorporate user specific logic** without modifications of core components

Evolution of device upgrade events



Future architecture



REFERENCES

1. L. Burdzanowski et al, “CERN Controls Configuration Service - a challenge in usability”, Barcelona, Spain, 8 - 13 Oct 2017, pp.TUBPL01
2. L. Burdzanowski et al, “The Renovation of the CERN Controls Configuration Service”, Melbourne, Australia, 17 - 23 Oct 2015, pp.MOPGF006
3. D. Jacquet et al, “LSA - the High Level Application Software of the LHC - and Its Performance During the First Three Years of Operation”, San Francisco, CA, USA, 6 - 11 Oct 2013, pp.thppc058
4. J. Wozniak et al, “NXCALS - Architecture and Challenges of the Next CERN Accelerator Logging Service”, New York, United States, 5 - 11 Oct 2019, pp.WEPHA163
5. https://www.espertech.com/esper/esper-faq