

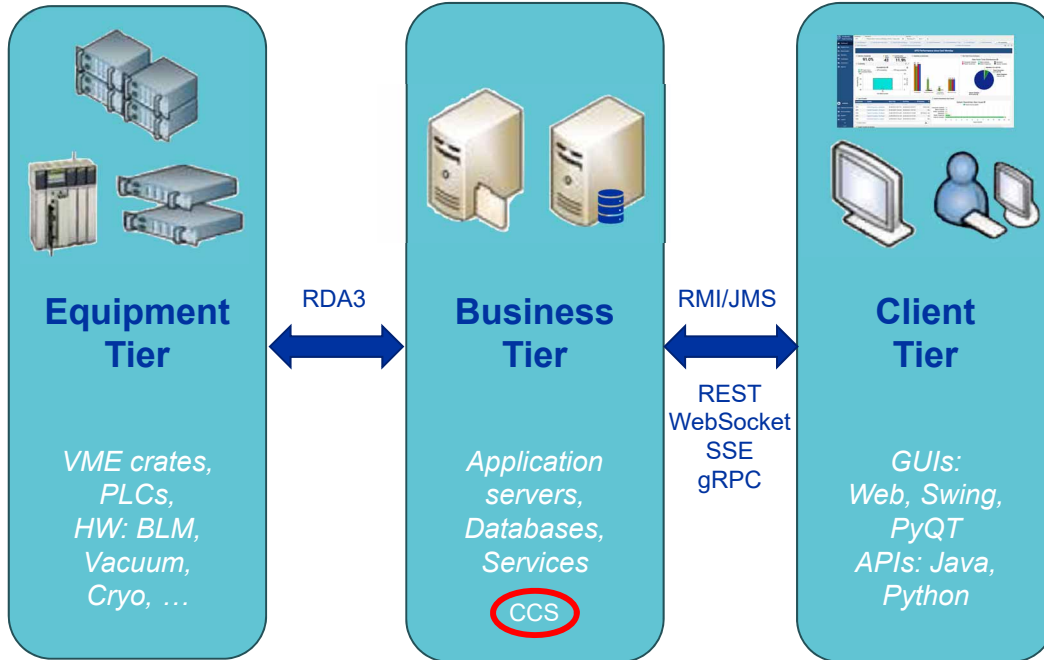
Developing Modern High-Level Controls APIs

Bartek Urbaniec

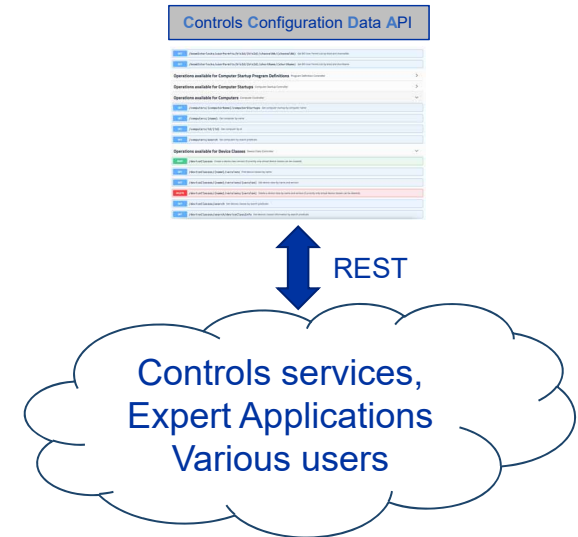
ICALEPCS 2023

APIs in the Accelerator Control System

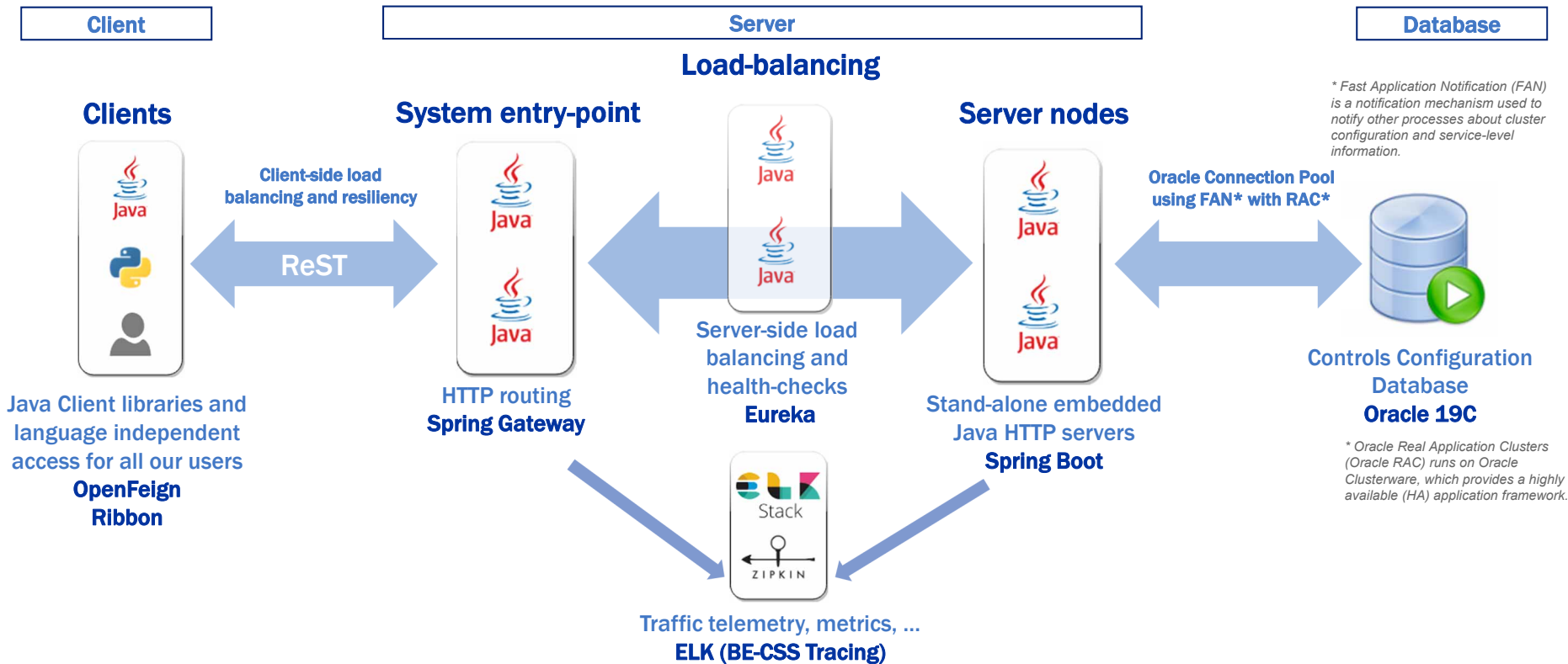
CERN Controls system services rely on a variety of APIs



The main purpose of the **Controls Configuration Service (CCS)** is to **unite and centralize** all the **information** relevant to the **Control systems (CS)** in such a way that integration between various Control sub-systems is consistent and efficient.



CCDA - the Controls Configuration API at glance

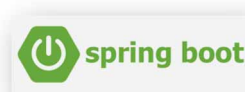


CCDA in numbers

- Up to **80 million** requests per day – coming from more than **400 hundred** different users, services and applications



- Availability this year 99.999% (1 hour of unplanned downtime over 9 months) – upgrades and releases are transparent due to multiple nodes and client/server balancing
- The API provides operations for **15 different controls domains** and more than **500 DB tables**
- More than **75k lines of code** – no boilerplate code due to usage of code generation libraries



Monitoring and alerting

- notifies us about all anomalies – helps to prevent system downtime
- allows to improve quality of service – problems are fixed before users spot them

monit alert -- Does not exist ACCSOFT-CCS-CCDA-1



Monit <[redacted]@cern.ch>
controls-configuration-notifications (Automatic notifications from Controls Configuration Service)
Thursday, 18 November 2021 at 10:12
[Show Details](#)

Does not exist Service ACCSOFT-CCS-CCDA-1
Date: Thu, 18 Nov 2021 10:12:15
Action: restart
Host: [redacted]
Description: process is not running

Your faithful employee,
Monit

```
Request URL: https://[redacted]/api/edge/configuration/hardware-types/VFC_HD_BSRA/Logical-hw-interfaces/vfc_hd_bsra/versions/0.0.1-dev
Request method: PUT
Referer header: null
Application name: ACCSOFT-CCS-CCDA-PRO
Username: belohrad

Root cause: NullPointerException:
Request body:
null

Stack trace:
java.lang.NullPointerException
    at cern.acsoft.ccs.ccda.domain.edge.mapper.BlockInstanceMapper.toBlockInstanceEntities(BlockInstanceMapper.java:18)
    at cern.acsoft.ccs.ccda.domain.edge.mapper.EdgeConfigurationMapper.lambda$buildBlockInstanceEntities$9(EdgeConfigurationMapper.java:143)
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1541)
    at cern.acsoft.ccs.ccda.domain.edge.mapper.EdgeConfigurationMapper.buildBlockInstanceEntities(EdgeConfigurationMapper.java:141)
    at cern.acsoft.ccs.ccda.domain.edge.mapper.EdgeConfigurationMapper.toRegisterMapDefinitionEntity(EdgeConfigurationMapper.java:38)
    at cern.acsoft.ccs.ccda.domain.edge.service.HardwareModuleRegisterMapDefinitionService.save(HardwareModuleRegisterMapDefinitionService.java:43)
    at cern.acsoft.ccs.ccda.domain.edge.service.HardwareModuleRegisterMapDefinitionService.update(HardwareModuleRegisterMapDefinitionService.java:60)
    at cern.acsoft.ccs.ccda.domain.edge.service.HardwareModuleRegisterMapDefinitionService$$astcClassBySpringCDLIBS$C0F8508.invoke(<generated>)
    at org.springframework.cglib.proxy.MethodProxy.invoke(MethodProxy.java:218)
```



Prometheus

Instance is DOWN (0 active)

```
alert: Instance
      is DOWN
expr: up == 0
for: 1m
labels:
  severity: CRITICAL
annotations:
  description: '{{ $labels.instance }} of job {{ $labels.job }} has been down for
    more than 1 minute.'
  summary: Instance {{ $labels.instance }} is down
```

1 alert for job=ACCSOFT-CCS-CCDA-DEV

[View In AlertManager](#)

[1] Firing

Labels

alertname = Instance is DOWN
instance = ACCSOFT-CCS-CCDA-DEV-2
job = ACCSOFT-CCS-CCDA-DEV
severity = CRITICAL

Annotations

description = ACCSOFT-CCS-CCDA-DEV-2 of job ACCSOFT-CCS-CCDA-DEV has been down for more than 1 minute.

summary = Instance ACCSOFT-CCS-CCDA-DEV-2 is down

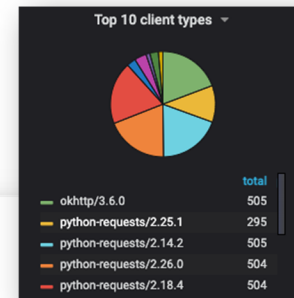
[Source](#)

Telemetry

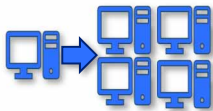
- informs us about usage of our API – who, how, when
- allows us to analyse and discover anomalies of a running system



```
t controller DeviceController
t domain CCS
t host [REDACTED] CERN.CH
t level INFO
t method findByNameOrAlias
t nameOrAlias FTN.BHZ459
t pid 10007
t process ACCSOFT-CCS-CCDA-1.jvm
t sourcename ACCSOFT-CCS-CCDA-PRO
t subdomain CCDA
t timestamp_nanos 918000000
# tracing_processingtime 121
@ tracing_timestamp Nov 17, 2021 @ 10:13:54.039
t xCcdaApplicationPath /opt/inca-server/ps-inca-server/lib/accsoft-ccs-ccda-client-core-1.5.1.jar
t xCcdaHostname [REDACTED] cern.ch
t xCcdaIPAddress [REDACTED]
t xCcdaOSUsername [REDACTED]
t xCcdaRequestURI /api/devices/nameOrAlias/FTN.BHZ459
t xCcdaTraceId deb1fccc4959814e
t xCcdaUserAgent okhttp/3.6.0
t xCcdaUsername unknown
t xCcdaVersion 1.5.1
```



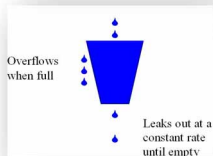
High-throughput enablers



- **Horizontal scaling**
adds resources to handle more client requests



- **Caching mechanism**
(more on the next slide)



- **Client throttling** (e.g. Leaky bucket algorithm)
limits misbehaving clients by rejecting requests



- **Circuit breaker**
stops requests in case of consecutive errors

To cache or not to cache

The premise of a cache - to store and provide already processed data

Benefits:

- limits unnecessary IOs → in many systems, physical IOs are the slowest operations
- load on the server and related services is reduced to minimum → output/data is processed only once

Drawbacks:

- increased complexity of the system (embedded cache vs standalone)
- eviction strategy* – especially difficult for complex system with mutable data
- challenge of consistency in distributed systems – every client should see the same state of cached data
→ there are solutions: e.g.: distributed cache like Apache Ignite, Redis

Cache eviction strategies:

- by time, e.g.: after n-seconds (TTL)
- by access frequency, e.g.: after 10k cache reads
- on-demand, e.g.: explicit cache purge
- by size, e.g.: max no. of elements in the cache

CCDA responses reduced by factor of 10

query for an accelerator: 40ms -> 4ms

Caching is not a silver bullet for performance issues

How to verify our system - testing

- Unit, integration, regression tests – availability and reliability

	Declarative: Checkout SCM	1 - pre-Build	2 - Test: JUnit	3 - Test: Integration	4 - Build: Publish	5 - Test: Integration Backward compatibility	6 - Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~23min 45s)	2s	1min 23s	1min 26s	8min 20s	1min 26s	7min 43s	3min 20s	68ms
#699 Nov 18 11:58 1 commit	2s	1min 21s	1min 27s	8min 26s	1min 27s	7min 47s	3min 19s	76ms

- Stress testing – performance

7 - Gatling
Stress testing

2min 59s

2min 59s

```
=====
2021-10-14 10:43:10                               161s elapsed
----- Requests -----
> Global (OK=12577 KO=0 )
> findSelectors (OK=611 KO=0 )
> findFesaFieldValuesByName (OK=566 KO=0 )
> findDevicesByQuery (OK=1196 KO=0 )
> findAcceleratorByName (OK=1210 KO=0 )
> findDeviceClassesByQuery (OK=1220 KO=0 )
> findAllAccelerators (OK=1210 KO=0 )
> findDeviceByName (OK=1802 KO=0 )
> findDeviceClassVersionsByName (OK=1781 KO=0 )
> findDeviceClassVersionsByNameAndVersion (OK=1781 KO=0 )
> findComputerByName (OK=1200 KO=0 )
```



```
----- GLOBAL INFO -----
> request count
> min response time
> max response time 1215 (OK=1215 KO=- )
> mean response time 86 (OK=86 KO=- )
> std deviation 168 (OK=168 KO=- )
> response time 50th percentile 36 (OK=36 KO=- )
> response time 75th percentile 75 (OK=75 KO=- )
> response time 95th percentile 278 (OK=278 KO=- )
> response time 99th percentile 973 (OK=973 KO=- )
> mean requests/sec 78.118 (OK=78.118 KO=- )
----- Response Time Distribution -----
> t < 500 ms 12197 ( 97%)
> 500 ms < t < 2000 ms 380 ( 3%)
> t > 2000 ms 0 ( 0%)
> failed 0 ( 0%)
```

Usability and documentation

Swagger library is used to autogenerate user documentation in OpenAPI specification



Operations available for Device Classes

- POST** /deviceClasses Create a device class version (Currently only virtual device classes can be created).
- GET** /deviceClasses/{name}/versions Find device classes by name
- GET** /deviceClasses/{name}/versions/{version} Get device class by name and version
- DELETE** /deviceClasses/{name}/versions/{version} Delete a device class by name and version
- GET** /deviceClasses/search Get devices classes by search predicate
- GET** /deviceClasses/search/deviceClassInfo Get devices classes information by search predicate

```
@Operation(summary = "Get device class by name and version")
@GetMapping(value = "{name}/versions/{version}")
public DeviceClass findByNameAndVersion(@PathVariable String name, @PathVariable String version) {
    return deviceClassService.findByNameAndVersion(name, version);
}

@Operation(summary = "Find device classes by name")
@GetMapping(value = "{name}/versions")
public List<DeviceClassInfo> findByName(@PathVariable String name) { return deviceClassService.findByName(name); }
```

Client SDKs (Java + Python) with autocompletion

```
deviceClassService.|
var
findByName(String name)
deleteDeviceClass(String name, String version)
createDeviceClass(DeviceClass deviceClass)
searchDeviceClassInfo(String query, int page) CcdaPage<DeviceClass>
findByNameAndVersion(String name, String version) DeviceClass
search(String query) List<DeviceClass>
search(String query, int page) CcdaPage<DeviceClass>
search(String query, int page, int size) CcdaPage<DeviceClass>
search(String query, PageQuery pageQuery) CcdaPage<DeviceClass>
searchDeviceClassInfo(String query) List<DeviceClassInfo>
searchDeviceClassInfo(String query, int page, int size) CcdaPage<DeviceClassInfo>
searchDeviceClassInfo(String query, PageQuery pageQuery) CcdaPage<DeviceClassInfo>
```

```
cli.DeviceClass.|
search(cls, query, page_size)
find(cls, name, version)
name DeviceClass
device_class_info DeviceClass
device_class_properties DeviceClass
description DeviceClass
framework_version DeviceClass
```



Summary

- **After 5+ years of CCDA in production, we are satisfied with our choices:**
REST, Spring Boot, Spring Cloud, stateless nodes, client/server load balancers
- **It pays off to utilize industry standards where possible:**
focus on your domain and API usability; most technical problems have ready to use solutions off-the-shelf
- **Start with language specific SDKs and documentation as a code (Swagger):**
limits proliferation of custom solutions and API wrappers among the users
- **Design of a robust and scalable API is a long lasting and iterative story:**
next steps for CCDA is k8s and circuit breakers

