Simon Rees :: Software Developer :: Paul Scherrer Institut

WICA-REST and WICA-JS

Two components for bringing PSI’s EPICS control system data to the web

EPICS Collaboration Meeting ITER June 2019

Project Motivations

Primary Goal – Status Display Replacement

“Replace the displays which show the status of PSI’s main facilities with something that scales better to the future.”

Minimum Requirements

- Provide a tool for PSI’s offsite technical staff -> should allow them to verify that the scientific facilities are working correctly.
- Provide a tool for PSI remote users -> should allow them to see how the program of work is evolving (so they can decide whether to come on-site)
- Improved User Experience -> should provide a “fancier”, more responsive user interface that work equally well on desktop, tablet and mobile devices. Buzzword: “PWA’s.

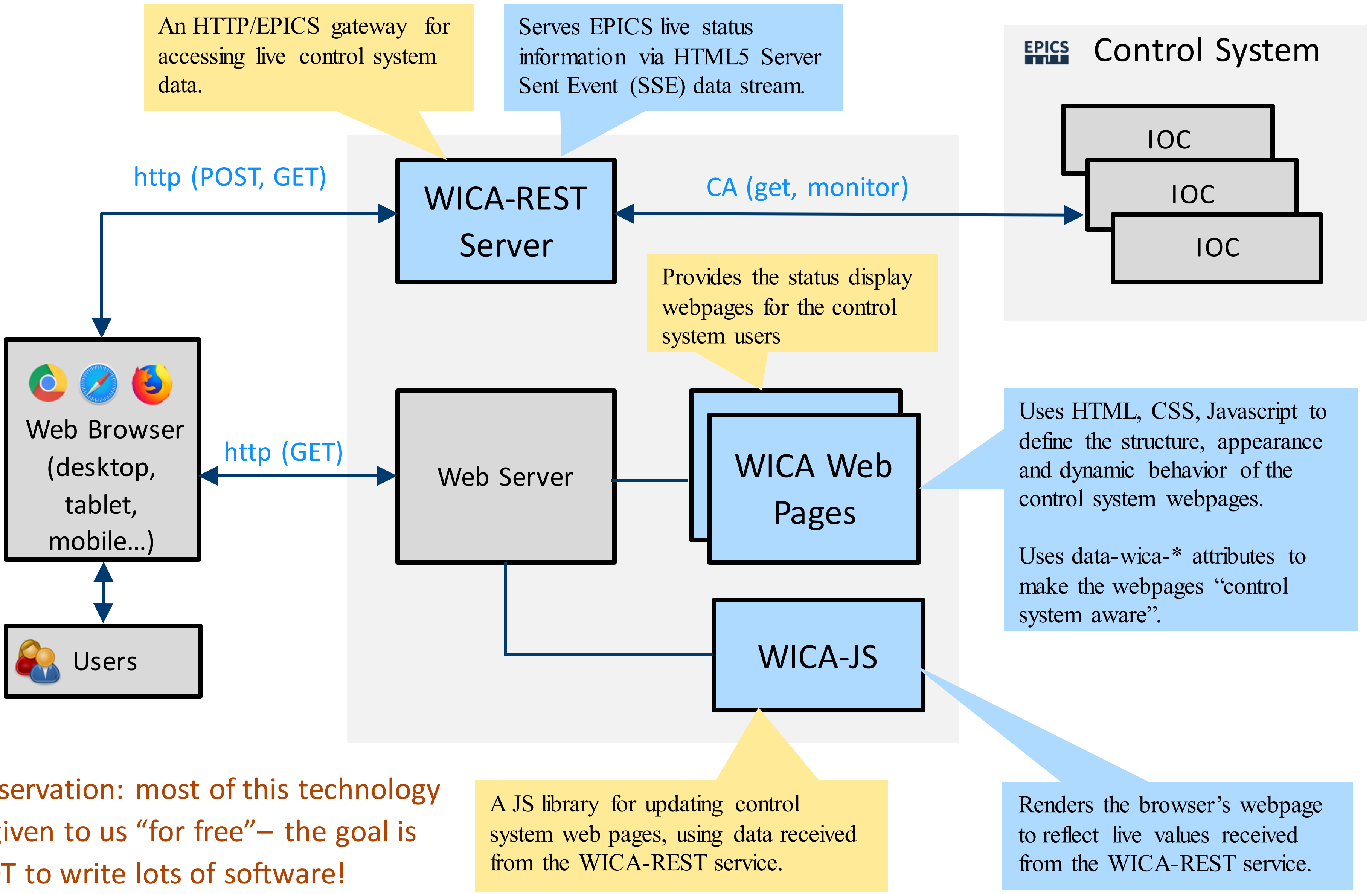
Secondary Goal – Learn Lessons

“Explore the difficulties of achieving the above by leveraging off powerful, modern, well-tested and widely-used web frameworks and libraries.”

Solution Summary

- A new solution has been developed called “**WICA**”.
- WICA stands for **Web Interface for Controls Applications**. (The CA previously stood for “Channel Access”, but the intention is to make the library compatible with newer control system protocols when/if we actively use them).
- WICA consists of:
  - **WICA-PAGES**: these are standard HTML5 webpages that leverage off user-defined ‘**data-wica-*’** attributes to configure the control system channels of interest and to define other properties needed to render the element.
  - **WICA-REST**: this is an HTTP microservice which provides a means of getting, setting and/or streaming live data from an EPICS-based control system.
  - **WICA-JS**: this is a Javascript library which scans the WICA webpages, sets up a live data stream from the WICA-REST server and which updates the visual appearance of the elements in real time.
- WICA can render the textual content of html elements directly, or can work with other JS libraries (eg plot libraries like **Plotly**, **Highcharts** or web component libraries like **LitElement**) where more sophisticated functionality is required.

WICA Overview Picture



An HTTP/EPICS gateway for accessing live control system data.

Serves EPICS live status information via HTML5 Server Sent Event (SSE) data stream.

Provides the status display webpages for the control system users

Uses HTML, CSS, Javascript to define the structure, appearance and dynamic behavior of the control system webpages.

Uses data-wica-* attributes to make the webpages “control system aware”.

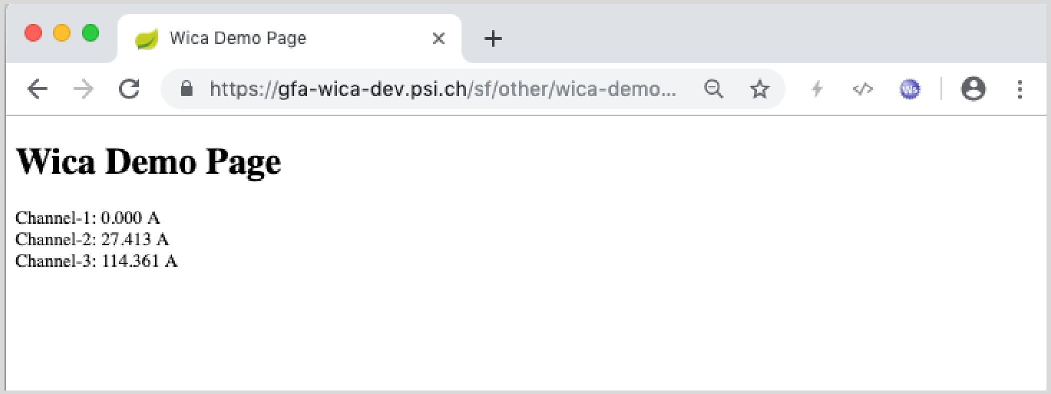
A JS library for updating control system web pages, using data received from the WICA-REST service.

Renders the browser’s webpage to reflect live values received from the WICA-REST service.

Observation: most of this technology is given to us “for free”– the goal is NOT to write lots of software!

Simple Wica Web Page Example

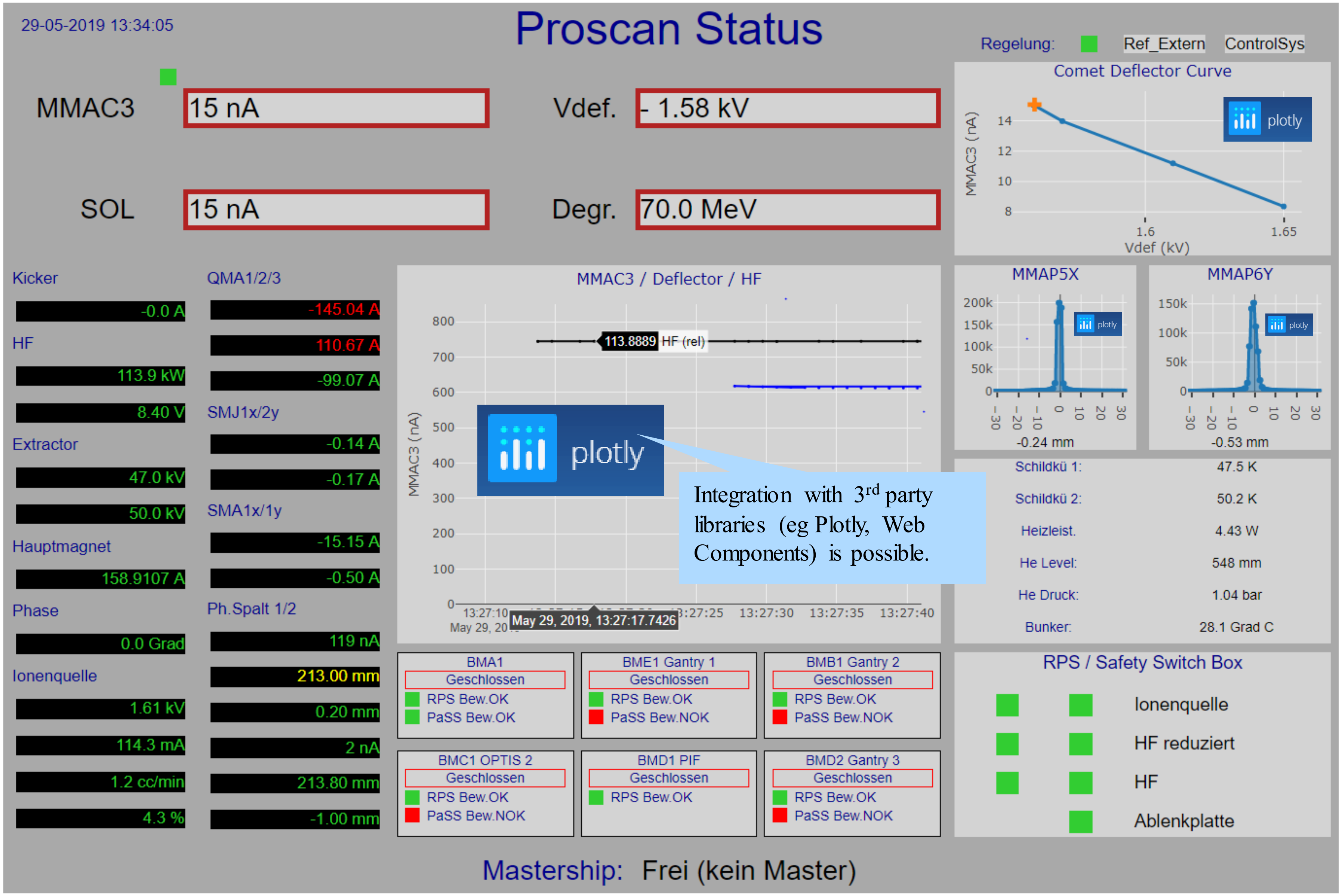
```
<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8"/>
    <title>Wica Demo Page</title>
    <script src="/wica/wica.js" type="module"></script>
</head>
<body>
    <h1>Wica Demo Page</h1>
    <label>Channel-1:</label> <span data-wica-channel-name="SINEG01-MBND300:I-READ"></span> <br>
    <label>Channel-2:</label> <span data-wica-channel-name="SINLH02-MBND100:I-READ"></span> <br>
    <label>Channel-3:</label> <span data-wica-channel-name="SINBC02-MBND100:I-READ"></span> <br>
</body>
</html>
```



Wica Demo Page

Channel-1: 0.000 A
Channel-2: 27.413 A
Channel-3: 114.361 A

More Complicated Example: Goal was to make it as ugly as the original ! ;-)



29-05-2019 13:34:05

Proscan Status

Regelung: 🟩 Ref_Extern ControlSys

MMAC3	15 nA	Vdef.	- 1.58 kV
SOL	15 nA	Degr.	70.0 MeV

Kicker	QMA1/2/3
-0.0 A	-145.04 A
HF	110.67 A
113.9 kW	-99.07 A
8.40 V	SMJ1x/2y
Extractor	-0.14 A
47.0 kV	-0.17 A
50.0 kV	SMA1x/1y
Hauptmagnet	-15.15 A
158.9107 A	-0.50 A
Phase	Ph.Spalt 1/2
0.0 Grad	119 nA
Ionenquelle	213.00 mm
1.61 kV	0.20 mm
114.3 mA	2 nA
1.2 cc/min	213.80 mm
4.3 %	-1.00 mm

Schildkü 1:	47.5 K
Schildkü 2:	50.2 K
Heizleist.	4.43 W
He Level:	548 mm
He Druck:	1.04 bar
Bunker:	28.1 Grad C

BMA1	BME1 Gantry 1	BMB1 Gantry 2
Geschlossen	Geschlossen	Geschlossen
🟩 RPS Bew.OK	🟩 RPS Bew.OK	🟩 RPS Bew.OK
🟩 PaSS Bew.OK	🟥 PaSS Bew.NOK	🟥 PaSS Bew.NOK

BMC1 OPTIS 2	BMD1 PIF	BMD2 Gantry 3
Geschlossen	Geschlossen	Geschlossen
🟩 RPS Bew.OK	🟩 RPS Bew.OK	🟩 RPS Bew.OK
🟥 PaSS Bew.NOK		🟥 PaSS Bew.NOK

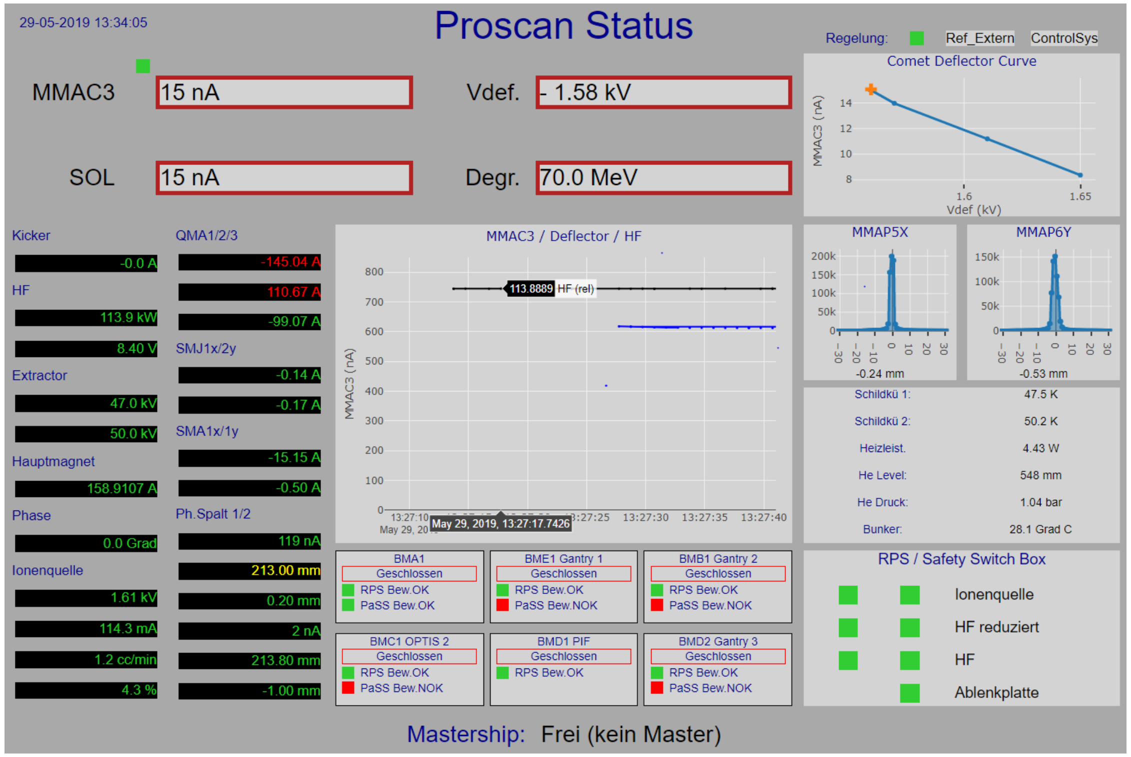
RPS / Safety Switch Box

🟩	🟩	Ionenquelle
🟩	🟩	HF reduziert
🟩	🟩	HF
	🟩	Ablenkplatte

Mastership: Frei (kein Master)

Integration with 3rd party libraries (eg Plotly, Web Components) is possible.

Because the pages just use normal web technology we can use CSS media queries to change the formats according to the features of the viewing device…


29-05-2019 13:34:05

Proscan Status

Regelung: 🟩 Ref_Extern ControlSys

MMAC3	15 nA	Vdef.	- 1.58 kV
SOL	15 nA	Degr.	70.0 MeV

Kicker	QMA1/2/3
-0.0 A	-145.04 A
HF	110.67 A
113.9 kW	-99.07 A
8.40 V	SMJ1x/2y
Extractor	-0.14 A
47.0 kV	-0.17 A
50.0 kV	SMA1x/1y
Hauptmagnet	-15.15 A
158.9107 A	-0.50 A
Phase	Ph.Spalt 1/2
0.0 Grad	119 nA
Ionenquelle	213.00 mm
1.61 kV	0.20 mm
114.3 mA	2 nA
1.2 cc/min	213.80 mm
4.3 %	-1.00 mm

Schildkü 1:	47.5 K
Schildkü 2:	50.2 K
Heizleist.	4.43 W
He Level:	548 mm
He Druck:	1.04 bar
Bunker:	28.1 Grad C

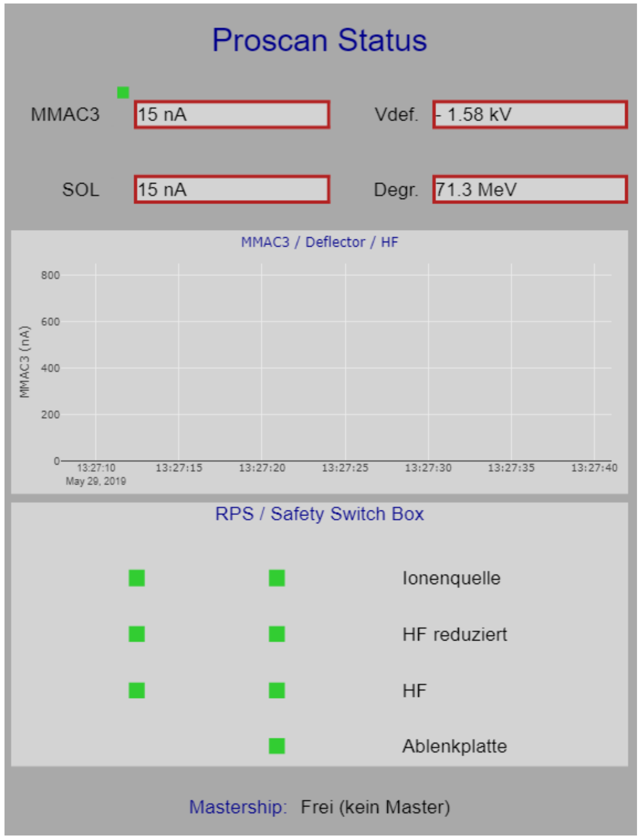
BMA1	BME1 Gantry 1	BMB1 Gantry 2
Geschlossen	Geschlossen	Geschlossen
🟩 RPS Bew.OK	🟩 RPS Bew.OK	🟩 RPS Bew.OK
🟩 PaSS Bew.OK	🟥 PaSS Bew.NOK	🟥 PaSS Bew.NOK

BMC1 OPTIS 2	BMD1 PIF	BMD2 Gantry 3
Geschlossen	Geschlossen	Geschlossen
🟩 RPS Bew.OK	🟩 RPS Bew.OK	🟩 RPS Bew.OK
🟥 PaSS Bew.NOK		🟥 PaSS Bew.NOK

RPS / Safety Switch Box

🟩	🟩	Ionenquelle
🟩	🟩	HF reduziert
🟩	🟩	HF
	🟩	Ablenkplatte

Mastership: Frei (kein Master)


Proscan Status

MMAC3	15 nA	Vdef.	- 1.58 kV
SOL	15 nA	Degr.	71.3 MeV

RPS / Safety Switch Box

🟩	🟩	Ionenquelle
🟩	🟩	HF reduziert
🟩	🟩	HF
	🟩	Ablenkplatte

Mastership: Frei (kein Master)

WICA-REST Service: an HTTP/EPICS Gateway

The WICA REST Service supports HTTP operations (POST, PUT, GET, DELETE) on two information resources that map onto the underlying control system: ‘*Channel’* and ‘*Stream* ‘.

‘Channel’ means EPICS channel. ‘*Stream*’ is a collection of channels which can be aggregated together and sent down the wire as a single HTML5 Server-Sent-Event (SSE) message.

The implementation is based on **Java Open JDK** (11) and **Spring Boot** (2.1). The use of Spring Boot, particularly has a big impact on reducing the amount of code that needs to be written. Communication with the EPICS Control System is achieved using the PSI-funded **Java CA** client library (written by Matej Sekoranja, CosyLab).

At PSI the server is deployed in a **Docker Container**. We hope this will provide an easier migration path for scaling the solution upwards should the load become excessive.

WICA-REST Service - Create Stream

Command: 'Stream Create'

This command takes an array of channel names and returns a ***<stream_id>*** which can be used later for subscription purposes. Optional properties provide finer-grained control over the data when it is streamed.

```
POST /ca/streams
Content-Type: application/json
{ "channels" :[{ "name": "abc:def", "props": {"prec": 3 }, { "name": "ghi:jkl" }] }
Returns a new <streamId>, a unique string which can be used subsequently for subscribing.
```

What the backend server does:

- creates EPICS channels to obtain data from the IOCs on the backend control system
- obtains the EPICS channel metadata (type, alarm and control limits, etc.)
- establishes an EPICS ca monitor on each channel and begins to cache the received values.

Additional Options:

- control over the ***precision*** of the streamed data.
- control over the ***rate*** at which information is sent down the stream.
- control over whether the stream contains ***polled*** or ***monitored*** data.
- control over various types of ***filtering***.

WICA-REST Service – Subscribe to Scream

Command: 'Stream Subscribe'

This command takes a <*stream_id*> and returns the corresponding live data stream.

```
GET /ca/streams/<streamId>
Returns the HTML5 Server-Sent-Event Stream (SSE) for the specified <streamId>.
```

What the backend server does:

Returns a continuous stream of server-sent event messages with the following message types:

- **channel-metadata:** for all channels in the stream – **sent once**.
- **channel-initial-values**: for all channels in the stream – **sent once**.
- **channel-updated-values:** includes information on monitored channels which have changed – **sent periodically** (at a configurable rate).
- **channel-polled-values:** and/or polled channels whose polling interval has expired **- sent periodically** (at a configurable rate).
- **stream-heartbeat**: a message which the WICA-JS library uses to detect loss of the connection – sent periodically.

WICA-REST Service – Stream Messages

Example Stream Messages:

```
id:123
event:ev-wica-channel-metadata
data:{"AMAKI1:IST:2":{"type":"REAL","egu":"A","prec":3,"hopr":72.000000,"lopr":-
72.000000,"drvh":72.000000,"drvl":-72.000000,"hihi":NaN,"lolo":NaN,"high":NaN,"low":NaN}, ...etc }
:2019-03-06 09:39:39.407 - initial channel metadata

id:123
event:ev-wica-channel-value
data:{"MMAC3:STR:2":[{"val":15.069581,"sevr":0QMA1:IST:2":[{"val":-
91.472626,"sevr":0}],"QMA3:IST:2":[{"val":-97.093582,"sevr":0}]}
:2019-03-06 09:39:54.526 - initial channel values

id:123
event:ev-wica-channel-value
data:{"MMAC3:STR:2":[{"val":15.069581,"sevr":0QMA1:IST:2":[{"val":-
91.472626,"sevr":0}],"QMA3:IST:2":[{"val":-97.093582,"sevr":0}]}
:2019-03-06 09:39:54.526 - channel value changes

id:123
event:ev-wica-channel-value
data:{"MMAC3:STR:2##2":[{"val":15.069581,"ts":"2019-03-
06T09:39:54.527468"}],"CMJSEV:PWRF:2##2":[{"val":113.888885,"ts":"2019-03-
06T09:39:54.527522"}],"EMJCYV:IST:2##2":[{"val":0.922709,"ts":"2019-03-06T09:39:54.527459"}]}
:2019-03-06 09:39:54.528 - polled channel values

id:123
event:ev-wica-server-heartbeat
data:2019-03-06T09:39:54.348562
:2019-03-06 09:39:54.348 - server heartbeat
```

WICA-REST Service Commands – Channel Get / Put

Commands: 'Channel Get' and 'Channel Put'

These commands offer a very simple channel get/put capability.

```
GET /ca/channels/<channelName>[?timeout=XXX]

Returns a JSON string representation of the value of the channel. For a channel whose underlying
data source is EPICS the returned information looks like this:

{"type":"STRING","conn":true,"val":"15.101","sevr":0,"stat":0,"ts":"2019-03-
06T09:37:22.103198","wsts":"2019-03-06T09:37:22.103211","wsts-alt":1551865042103,"dsts-
alt":1551865042103}
```

```
PUT /ca/channels/<channelName>
Content-Type: text/plain the new value

Somevalue

Returns a string "OK" when the put was successful.
```

What the backend server does:

- creates an EPICS CA channel to the process variable on the EPICS control system.
- performs a synchronous, (= "confirmed") GET/PUT operation, returning/using the obtained/supplied value.

Additional Options:

- Control over the timeouts for the getting or putting the data.

PAUL SCHERRER INSTITUT PSI

WICA-JS Library

The WICA-JS library is loaded after the rest of the webpage to scan the source document for elements that are ''wica-aware" (= elements whose **'data-wica-channel-name'** attribute is set).

The library then collaborates with the WICA-REST server to stream back the channel metadata (eg alarm and display limits) and changing channel values, setting the following attributes to match the received data:

Attribute	**Description**
data-wica-stream-state	Contains status of connection to Wica Server.
data-wica-channel-connection-state	Contains status of connection to data source.
data-wica-channel-alarm-state	Contains alarm status of data source.
data-wica-channel-metadata	Contains last received metadata from data source.
data-wica-channel-value-latest	Contains last received value from data source.
data-wica-channel-value-array	Contains array of latest received values from data source.

Lessons Learned so far…

- Creating Flexible, Modern, Responsive, Beautiful webpages for today's web that integrate with our control system data is now possible. ☺
- But… leveraging off the possibilities of the today's modern web doesn't come cheap… there is a lot of extra information to think about when trying to build responsive, scalable layouts.
- Whatever format we choose **(.medm, .ui**, **.html**) there will always be a lot of extra information to capture that before our lab did not really care about…
- Tooling would be useful, but web designers seem to prefer hacking html and Javascript than using the builder tools that have been common in the EPICS community. To make a solution with wide applicability to our users we would need them.
- Fortunately "incompatible browser syndrome" is less of a problem these days… however ***Internet Explorer*** (and their newer browser ***Edge***) are still outside the web community and likely to remain so. At our Lab we mainly ignore them. (JS Libraries such as ***Modernizr*** can assist in autodetecting incompatibilities and providing a hint to the user if the browser does not adequately support the required features).

Project Next Steps

Consolidation

- Complete **WICA-RELAY** -> access webpages from internet whilst satisfying PSI’s internet security guidelines.
- Complete rollout of displays for all PSI’s scientific facilities.
- Publish **WICA-REST** and **WICA-JS** to web (if there is interest).

Improve the User Experience

“Make further improvements that take advantage of the capabilities offered by our new enabling technologies (SpringBoot, Lit-Element, modern JS)”

- Possibility to include Epics status information in any web page.
- Create webpages that comply with the requirements of PSI’s communication department
- Provide status displays usable from other devices (tablets, phones, wearables…)

Future

- Consider whether a GUI Build Tool would be viable.
- Consider whether an import tool would be viable (PSI’s PEP tool at least should be straightforward)

Thanks for your attention ! ☺

Thanks go to:

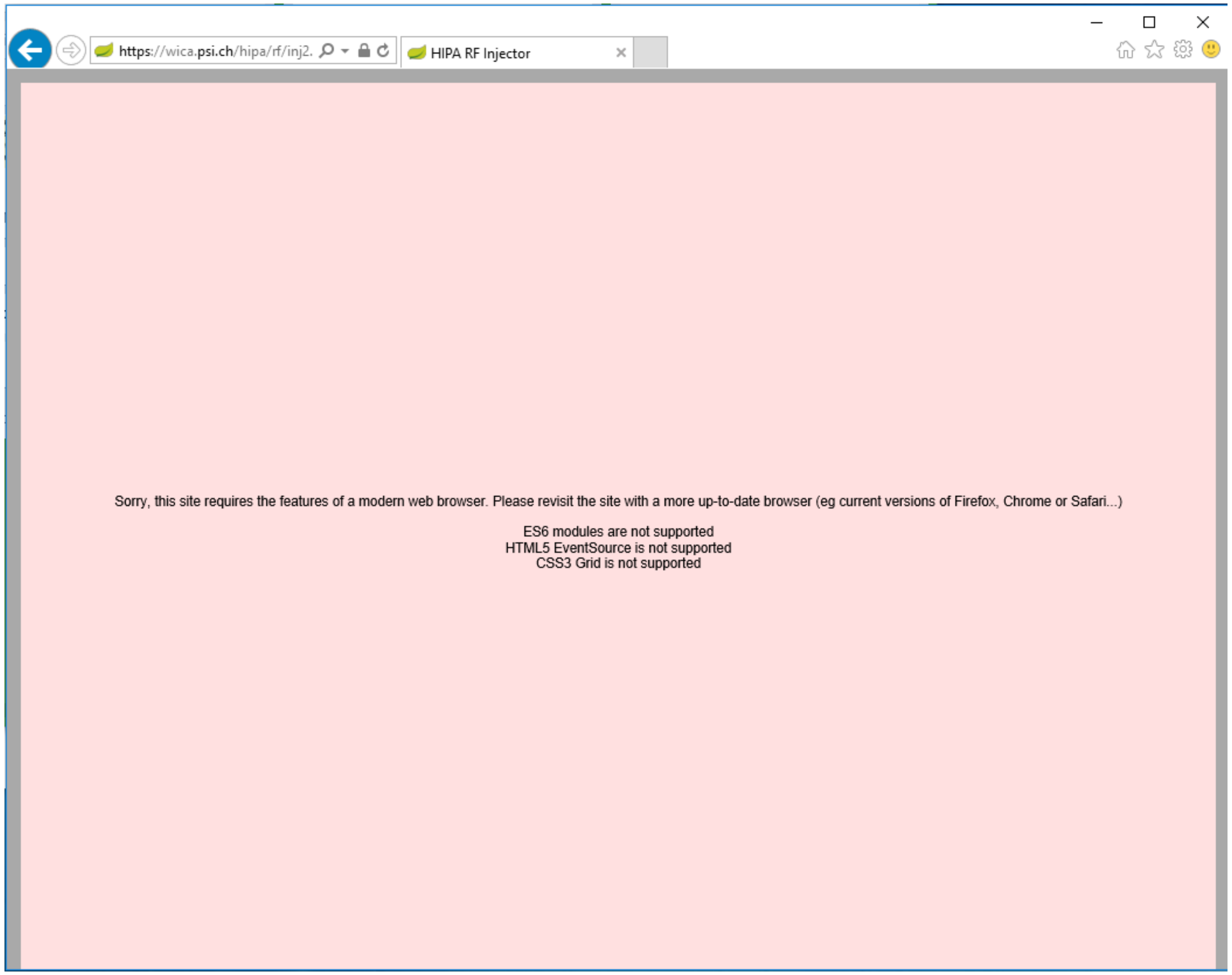
- Simon Ebner
- Daniel Lauk

And especially:

- Dirk Zimoch (for giving this talk)

For questions on this work please contact:
- Simon Rees - simon.rees@psi.ch or
- Simon Ebner - simon.ebner@psi.ch

Use of *Modernizr* JS library to autodetect any browser compatibility issues.



Sorry, this site requires the features of a modern web browser. Please revisit the site with a more up-to-date browser (eg current versions of Firefox, Chrome or Safari...)

ES6 modules are not supported
HTML5 EventSource is not supported
CSS3 Grid is not supported

Not looking at any companies anywhere in particular (Seattle)...

Goals and Non-Goals

- Provide an HTTP REST (micro)service for accessing PSI's EPICS-based control system from the web.
- Make it easy for users to create HTML pages that show the evolving live status of the control system.
- Explore some of the issues involved in bringing the control system's user-interface to the web.
- Currently Not a Goal: provide a general-purpose replacement for PSI's caQtDM display-building technology. (two essential components are still missing: design tool, conversion tool).

Concrete Use Case - 2

Provide EPICS Microservice for Snapshot Tool

Implementation -1

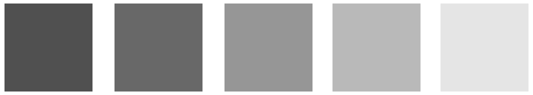
- General principle: consolidate our future solutions based on technologies that are the unquestionable standard in any serious IT companies. Everything has moved to the web… “no-one develops applications for the desktop any more”.

- Leverage off the general availability of powerful, highly scalable, open source frameworks to write the minimum code possible.

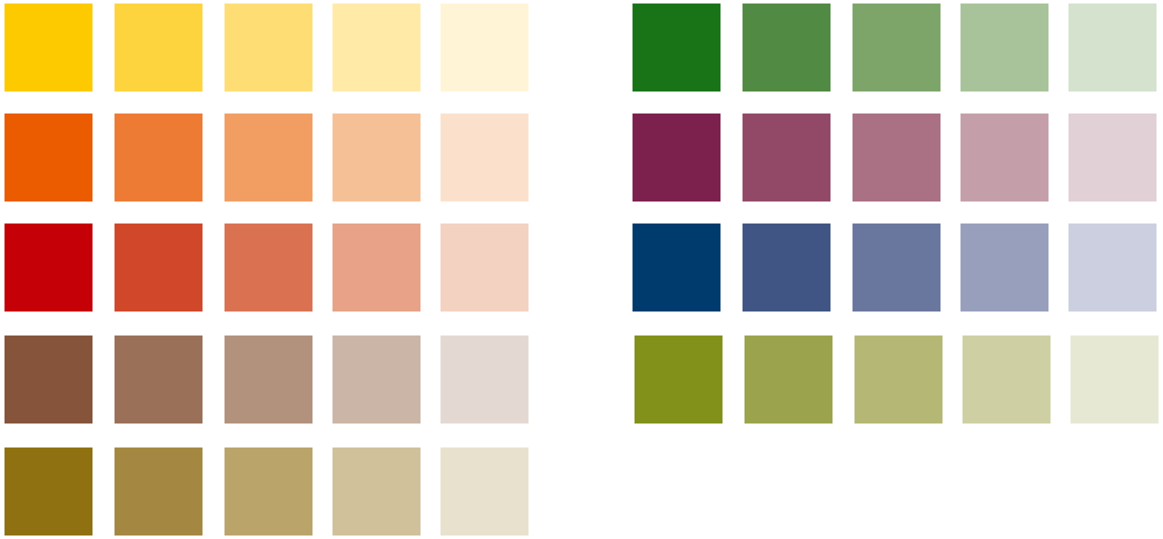
PAUL SCHERRER INSTITUT

PSI Colour Scheme

PSI‘s basic colours



Colour options for graphs:
1st choice



Colour options for graphs:
2nd choice