Behind the ESS public operations screen

Emanuele Laface
Henrik Carling

ESS/ICS
Date: 2019-06-05
Background
Background
Integrated Control Systems

- Use case 1: quick access to operational status overview for ESS staff from anywhere
- Use case 2: provide “on-line” operational status of the ESS facility to the public/internet
- Programmer idea 1: display EPICS data with the minimum possible quantity of code
- Programmer idea 2: provide extensible public http/json access to EPICS data
- Ambition: create a tool for simple EPICS data representation for everyone at ESS
Current status

- Current status
  - Three customized screens
    - Operation
    - Interlocks status
    - Beam instruments of the Ion source and LEBT
  - Graphic profile customization for ESS
  - Search engine for PVs
    - Requires some additional Python code
  - Python interface that exposes JSON files
    - Accessible from the internet
Current deployment

- **Docker**
  - Packed in a Docker container
- **PyEPICS**
- **EPICS Gateway**
  - Read Only EPICS Gateway
- **Freeboard**
  - Open Source Javascript Json parser and renderer
- **NGINX**
  - Webserver
- **Technical Network**
- **Ansible**
  - Deployed by Ansible
Programmers perspective

- The code is basically a translator from EPICS to Json. (Example: https://pos.esss.lu.se/data/getPublicData.json)

- The JavaScript/rendering part is done through the Freeboard editor without writing code (or very little for some customization)
  
  https://github.com/Freeboard/freeboard

- The graphic customization is done through a single CSS style sheet
Next steps

- Improvement potential
  - Polling/monitor
  - The Python polling system uses the monitor function of the pyepics class
    - Any other method times out when the PV is not available - can hang the polling thread
    - The timeout happens also in monitor mode but the variable is always accessible
    - The drawback of the monitor function is that data is received in Python at the speed of the machine: 14 Hz but I am interested in visualizing them at 1 Hz, overloading the network and the CPU of the Python server.
    - Can possibly be solved by IOC filtering?

```python
print('Connect to EPICS')
publicpvsv=[]
publicpvsv['ISrc-010:ISS-Magtr:PulsHLv1S']=epics.PV('ISrc-010:ISS-Magtr:PulsHLv1S', auto_monitor=True) # Magnetron Power
publicpvsv['ISrc-010:PwrC-RepPS-01:PwrR']=epics.PV('ISrc-010:PwrC-RepPS-01:PwrR', auto_monitor=True) # Repeller on/off
publicpvsv['ISrc-010:PwrC-RepPS-01:Vo1R']=epics.PV('ISrc-010:PwrC-RepPS-01:Vo1R', auto_monitor=True) # Repeller voltage
publicpvsv['LEBT-010:PwrC-RepPS-01:PwrR']=epics.PV('LEBT-010:PwrC-RepPS-01:PwrR', auto_monitor=True) # Repeller on/off
publicpvsv['LEBT-010:PwrC-RepPS-01:Vo1R']=epics.PV('LEBT-010:PwrC-RepPS-01:Vo1R', auto_monitor=True) # Repeller voltage
publicpvsv['ISrc-010:ISS-HVPS:PwrR']=epics.PV('ISrc-010:ISS-HVPS:PwrR', auto_monitor=True) # HV on/off
publicpvsv['ISrc-010:ISS-HVPS:Vo1R']=epics.PV('ISrc-010:ISS-HVPS:Vo1R', auto_monitor=True) # HV voltage
publicpvsv['ISrc-010:TS-EVG-01:Mxc1-Frequency-RB']=epics.PV('ISrc-010:TS-EVG-01:Mxc1-Frequency-RB', auto_monitor=True) # Frequency
publicpvsv['ISrc-010:ISS-EVR-Magtr:Pu0-Width-RB']=epics.PV('ISrc-010:ISS-EVR-Magtr:Pu0-Width-RB', auto_monitor=True) # Pulse length
publicpvsv['ISrc-010:Vac-VMC-01100:FlwR']=epics.PV('ISrc-010:Vac-VMC-01100:FlwR', auto_monitor=True) # H2 flow
publicpvsv['LEBT-010:Vac-VMC-01100:FlwR']=epics.PV('LEBT-010:Vac-VMC-01100:FlwR', auto_monitor=True) # N2 flow
publicpvsv['ISrc-010:Vac-VVA-01100:OpenR']=epics.PV('ISrc-010:Vac-VVA-01100:OpenR', auto_monitor=True) # H2 on/off
publicpvsv['LEBT-010:Vac-VVA-01100:OpenR']=epics.PV('LEBT-010:Vac-VVA-01100:OpenR', auto_monitor=True) # N2 on/off
```
Next steps

- Improvement potential
  - Flexibility
    - A user that wants a specific interface has to ask for the Python to JSON translator implementation for that set of PVs.
    - The ambition is to create an automatic system for the user to drop the list of PVs somewhere (git?) and have the JSON generated every second.
    - In this way the user can create the interface in Freeboard (or elsewhere) without writing any code.

```python
publicpvs_return_dict[pv]['units']=publicpvs[pv].units
if pv=='ISrc-010:PBI-BCM-001:AI4-Compressed':
    y_data_array=publicpvs[pv].value*bcm_correction
    y_data=list(y_data_array)
    x_data_array=linspace(0,12.8,len(y_data))
    x_data=list(x_data_array)
    publicpvs_return_dict[pv]['value']=list(zip(x_data,y_data))
    publicpvs_return_dict['pulse']={}
    publicpvs_return_dict['max_curr']={}
    publicpvs_return_dict['pulse']['units']='ms'
    publicpvs_return_dict['max_curr']['units']='mA'
if y_data_array.max()<1:
    publicpvs_return_dict['beam']=z
    publicpvs_return_dict['pulse']['value']=0
    publicpvs_return_dict['max_curr']['value']=0
```
Open project

- [https://gitlab.esss.lu.se/ics-software/ess-public-screens](https://gitlab.esss.lu.se/ics-software/ess-public-screens)
Live demo

• https://pos.esss.lu.se/