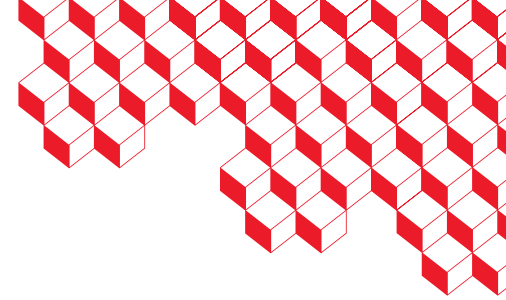




irfu



# High-Level Applications for the SARAF Accelerator

Between EPICS and Python

Antoine Choquet - CEA Paris Saclay



# Presentation outline

1. Definition and requirements
2. Graphical User Interfaces (GUIs) integration
3. Architecture and technical implementation
4. Status



# 1. Definition and requirements

# Why called: high-level?

These **applications** :

- do not interact directly with hardware
- rely on several other local control systems

Their **purposes** :

- acquire and compute data automatically
- help the operator configuring the accelerator
- reduce repetitive manual procedures

# Requirements

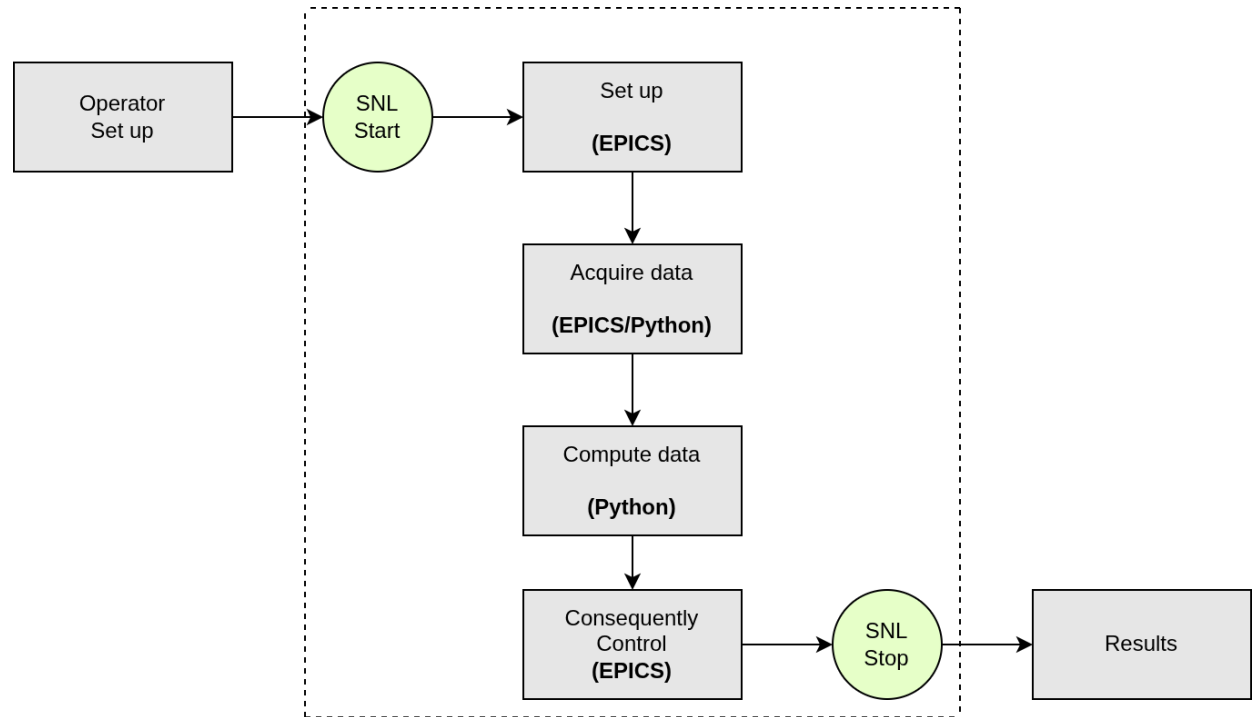
Accelerator experts from CEA :

- define requirements :
  - ▶ algorithms
  - ▶ general information and variables accessible from the **GUI**
  - ▶ expected results
  - ▶ involved equipments
  - ▶ ...
- write **Python analysis scripts**
- validate the applications with SNRC operators and EPICS developers

# Automated process : EPICS State Notation Language (SNL)

why SNL ?

- just a few steps
- simple code





# List & synthetic description

Having the same design:

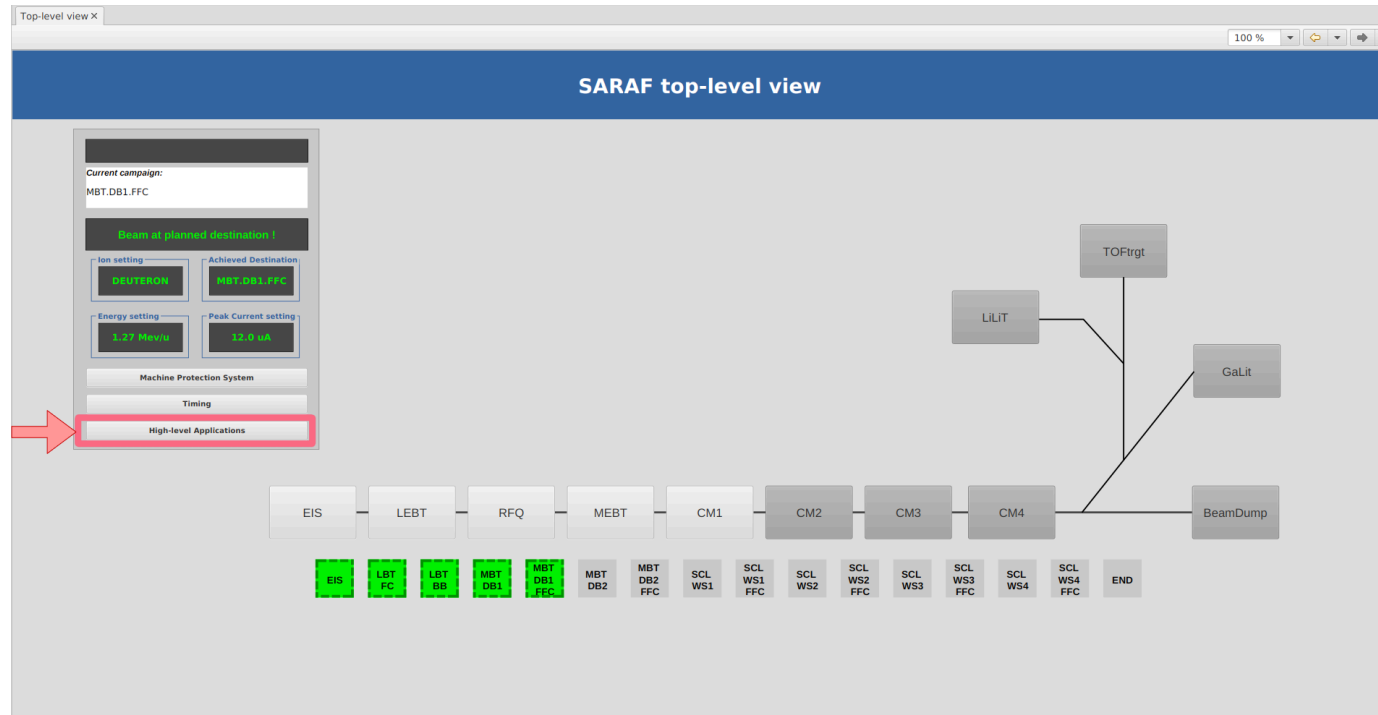
- **Time of Flight (ToF)** : measuring the beam energy,
- **CavityPhasing** : calibrating the cavities,
- **BpmCalib** : calibrating the Beam Position Monitor (BPM),
- **BeamAligner** : aligning the beam,
- **EmitScan** : calculating longitudinal and transversal emittance.



# 2. GUIs

## integration

# Access from top-level view



The **top-level view** is always available from the Phoebus **HOME** button

# High-level applications menu - Time of Flight

Standard menu (navigation panel on the left)

Easy integration using the **Phoebus widget** : Navigation tabs

The screenshot displays the 'Time of flight' application interface. On the left is a 'High-level App Menu' with options: Beam Operation Mode, Cavity phasing, Time of flight (selected), BPM calibration, Emit. Scan Transverse, Shut beam debug, Menu (MPS menu, Timing menu), Home, Detach the window, and Beam (STOP, START). The main area is divided into three sections:

- 1. BPM selection:** Includes a 'Selection' box with 'Accelerator section' (MEBT), 'BPM' (BPM-1), and 'Number of BPM to be used' (3). Below are rows for 'Selected BPM', '1st Cavity', '2nd Cavity', '1st downstream BPM', and '2nd downstream BPM', each with status indicators and measurement values.
- 2. Context:** Contains 'Beam Energy Estimation' (Data source: BOM, User Input: 1.100 MeV), 'Particles kind' (Data source: BOM, Particles kind: w000b), 'BPM position' (Selected BPM: 0.599 m, 1st downstream BPM: 1.438 m, 2nd downstream BPM: 2.853 m), and 'BPM configuration' (Frequency selection: 176 MHz, 352 MHz, 176 MHz; Phase measurement type: fine, coarse, fine).
- 3. New acquisition and analysis:** Features a 'Start sequence' (OK, Start), 'Beam Energy Result' (1.100 MeV, Push), 'Result Context' (Particles kind: w000b, BPM Frequency: 176 MHz, Selected BPM, 1st downstream BPM, 2nd downstream BPM), and 'Analysis of previous acquisitions' (select a file, last file, Analysis).

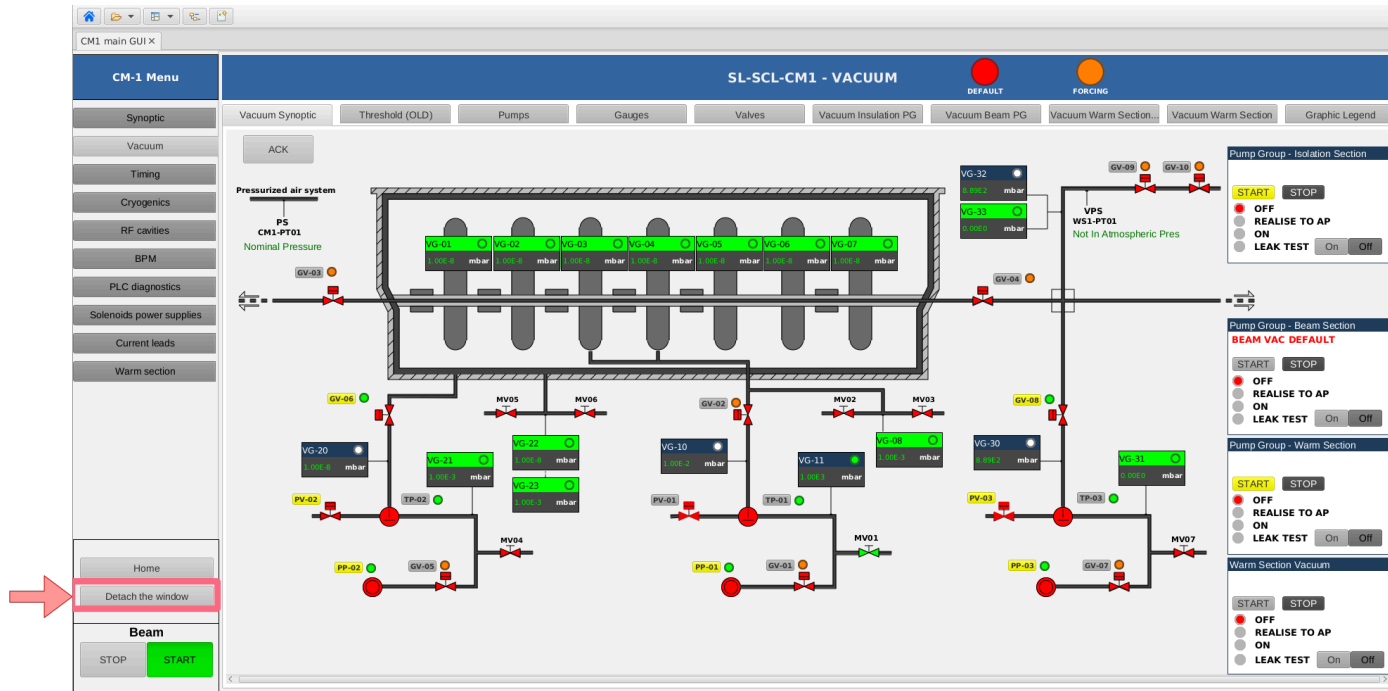
# Operate simultaneously on different systems

Meanwhile using the high-level applications, operators might need to check other systems → **Detach the window**

The screenshot displays the 'Time of flight' control interface. On the left is a 'High-level App Menu' with options like 'Beam Operation Mode', 'Cavity phasing', 'Time of flight', 'BPM calibration', 'Emit. Scan Transverse', 'Shut beam debug', 'Menu', 'MPS menu', 'Timing menu', 'Home', 'Detach the window', and 'Beam' (STOP/START). The main area is divided into three sections: '1. BPM selection', '2. Context', and '3. New acquisition and analysis'. '1. BPM selection' shows parameters for Accelerator section (MEBT), BPM (BPM-1), and Number of BPM to be used (3). It lists selected BPMs and cavities with their respective Meas phase, Uacc Meas, and Phiacc Meas. '2. Context' includes 'Beam Energy Estimation' (Data source: BOM+, User input: 1.100 MeV), 'Particles kind' (Data source: BOM+, User input: ppppp), and 'BPM position' (Selected BPM: 0.295 m, 1st downstream BPM: 1.258 m, 2nd downstream BPM: 2.265 m). '3. New acquisition and analysis' contains 'Start sequence' (ON), 'Beam Energy Result' (1.100 MeV), 'Result Context' (Particles kind: ppppp, BPM Frequency: 176 MHz), and 'Analysis of previous acquisitions' (select a file, last file).

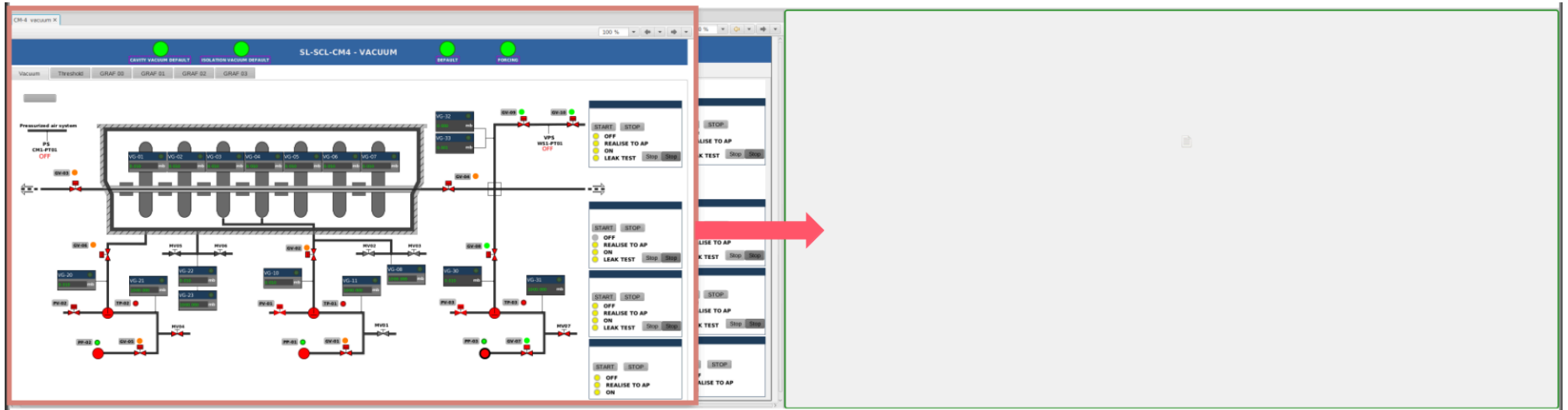
# Operate simultaneously on different systems

*example*



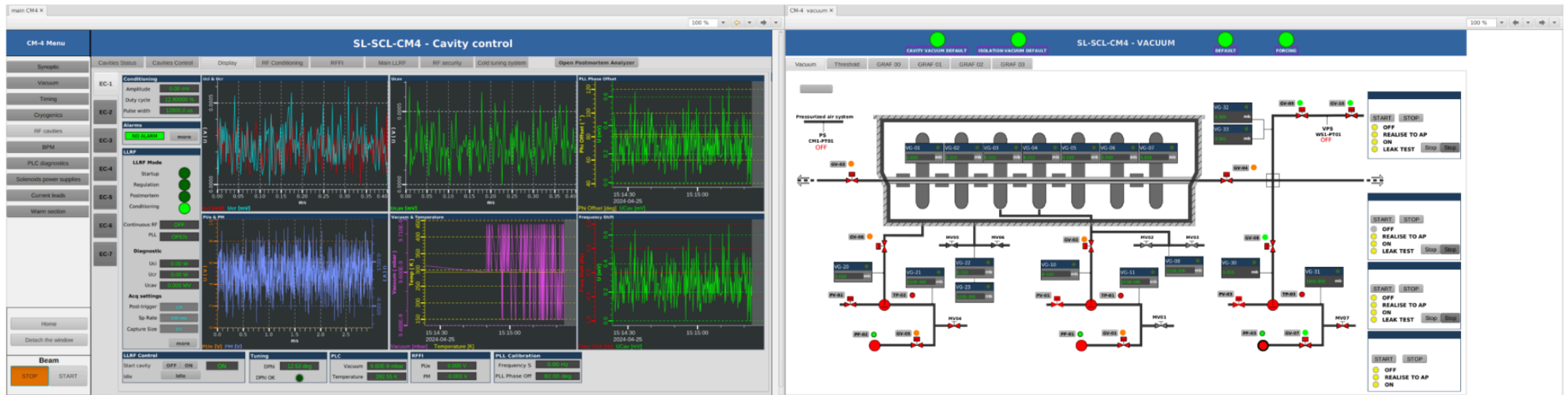
# Operate simultaneously on different systems

*example*



# Operate simultaneously on different systems

*example*

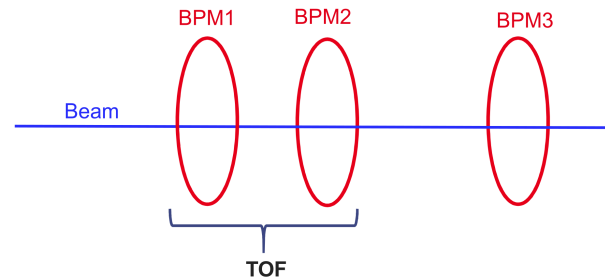




# 3. Architecture and technical implementation

# Time of Flight (ToF) - example

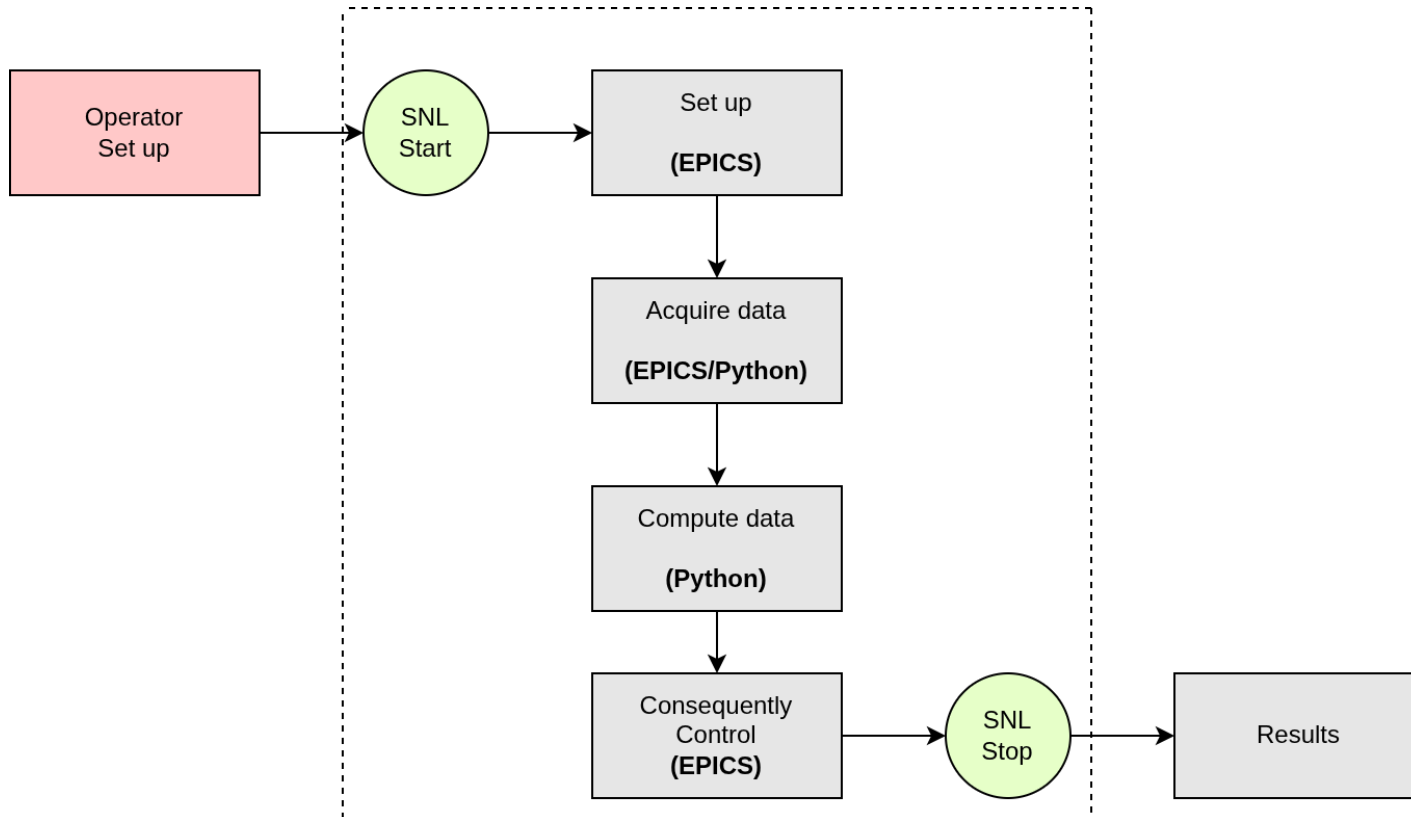
Application for measuring the beam energy anywhere in the SARAF LINAC



Synthetic requirements :

- recording beam phase in 2 or 3 BPMs – **Analysis** → calculates beam energy
- if 2 BPMs used, a first guess is required for discrimination
- the beam should not be accelerated between BPMs during ToF process

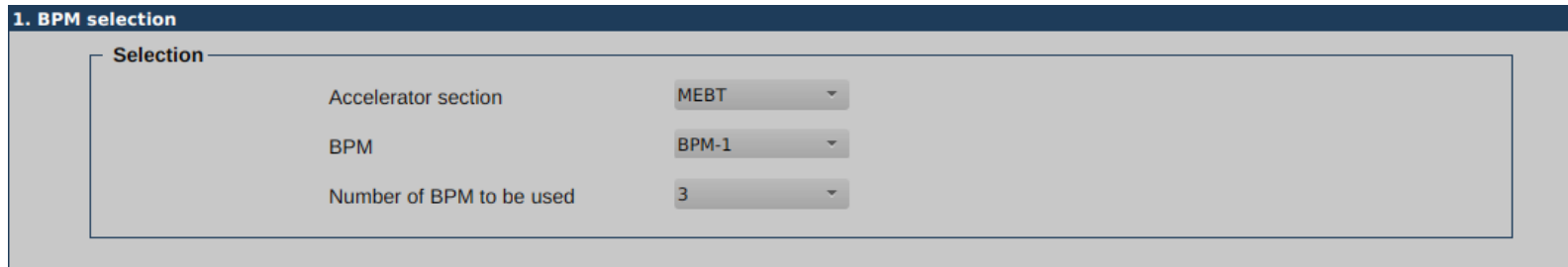
# Operator set up



# Operator set up - ToF - select BPMs

Operator selects :

- the first BPM
- the number of BPMs (2 or 3) to use in the process



1. BPM selection

Selection

Accelerator section	MEBT
BPM	BPM-1
Number of BPM to be used	3

24 BPMs along the SARAF accelerator

# Operator set up - ToF - select BPMs

Once first BPM selected, it is deduced the involved BPMs and cavities :

The screenshot displays the '1. BPM selection' interface. At the top, there is a 'Selection' box with three dropdown menus: 'Accelerator section' set to 'MEBT', 'BPM' set to 'BPM-1', and 'Number of BPM to be used' set to '3'. Below this, three rows of BPM configurations are shown. Each row includes a 'Selected BPM' button, a status indicator (green circle), 'Meas phase' with 'fine' and 'coarse' buttons, and 'Frequency' (176 MHz). The first row shows 'SL\_MBT\_MAG1\_DIA\_BPM\_1' with 'BPM Ready' status. The second row shows 'SL\_MBT\_MAG4\_DIA\_BPM\_2' with 'BPM Ready' status. The third row shows 'SL\_MBT\_MAG6\_DIA\_BPM\_3' with 'BPM Ready' status and a red label 'BPM after beam destination' below it. Each row also lists '1st Cavity' (SL\_MBT\_RBN1\_LRP\_CAV\_1, SL\_MBT\_RBN2\_LRP\_CAV\_1, SL\_MBT\_RBN2\_LRP\_CAV\_1) and '2nd Cavity' (None). 'Uacc Meas' (50.00) and 'Phiacc Meas' (2.00) are shown for the first two rows.

Selected BPM	Status	Meas phase	Frequency	1st Cavity	2nd Cavity	Uacc Meas	Phiacc Meas
SL_MBT_MAG1_DIA_BPM_1	BPM Ready	107.86 fine coarse	176 MHz	SL_MBT_RBN1_LRP_CAV_1	None	50.00	2.00
SL_MBT_MAG4_DIA_BPM_2	BPM Ready	109.48 fine coarse	176 MHz	SL_MBT_RBN2_LRP_CAV_1	None	50.00	2.00
SL_MBT_MAG6_DIA_BPM_3	BPM Ready	97.12 fine coarse	176 MHz				

The whole logic is based on [scalcout](#) records.

# Operator set up - ToF - context data

Operator enters :

- first guess beam energy
  - whether providing a range or selecting “nearest” mode
- particle kind
- BPMs frequency
- phase measurement type (coarse or fine)

The screenshot displays the '2. Context' configuration panel with the following settings:

- Beam Energy Estimation:**
  - Data source: BOM+ (selected), User Input
  - User input: 1.100 MeV
  - Beam Energy Range Mode: Nearest (selected), Range
  - Beam Energy Estimation: 1.100 MeV
- Particles kind:**
  - Data source: BOM+ (selected), User Input
  - Particles kind: proton
- BPM position:**
  - Selected BPM: 0.295 m
  - 1st downstream BPM: 1.238 m
  - 2nd downstream BPM: 2.993 m
- BPM configuration:**
  - Frequency selection: 176 MHz (selected), 352 MHz, 176 MHz
  - Phase measurement type: fine (selected), coarse, fine

# Operator set up - ToF - context data

The BPMs position are defined in a yaml file :

```
...
elements:
  SL-MBT-MAG1:DIA-BPM-1:
    length: 0
    pos: 0.29460000000000003
  SL-MBT-MAG4:DIA-BPM-2:
    length: 0
    pos: 1.2384000000000002
  SL-MBT-MAG6:DIA-BPM-3:
    length: 0
    pos: 2.9933
...
```

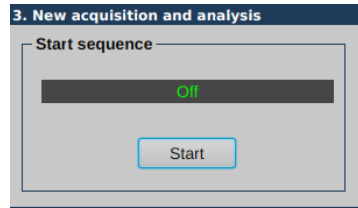
when operator selects the first BPM :

- a **Python** script reads the yaml file and writes values in PVs

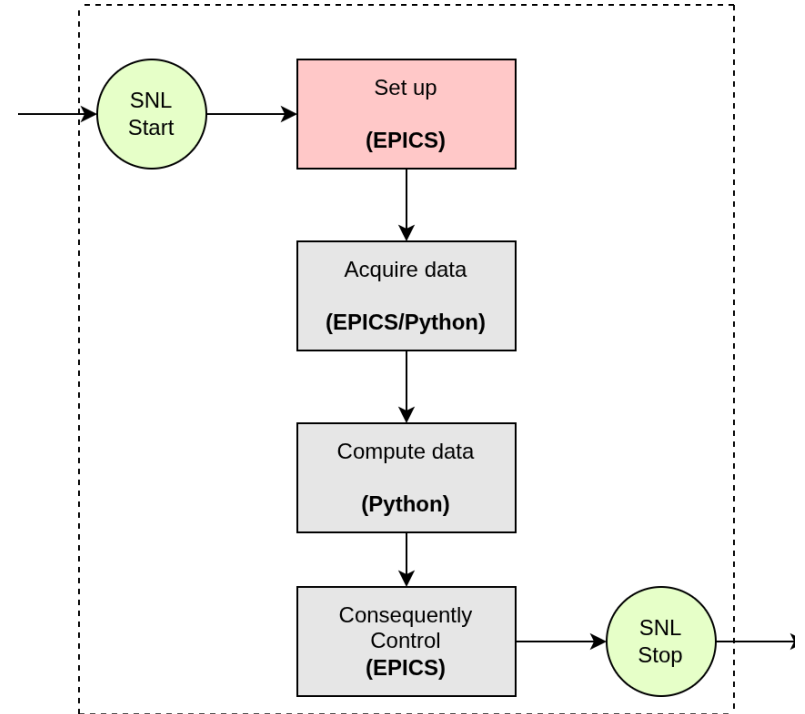
The screenshot shows a control panel titled "2. Context" with four main sections:

- Beam Energy Estimation:** Data source (BOM+ / User Input), User input (1.100 MeV), Beam Energy Range Mode (Nearest / Range), and Beam Energy Estimation (1.100 MeV).
- Particles kind:** Data source (BOM+ / User Input) and Particles kind (proton).
- BPM position:** Selected BPM (0.295 m), 1st downstream BPM (1.238 m), and 2nd downstream BPM (2.993 m). A red arrow points to this section.
- BPM configuration:** Frequency selection (176 MHz / 352 MHz / 176 MHz) and Phase measurement type (fine / coarse / fine).

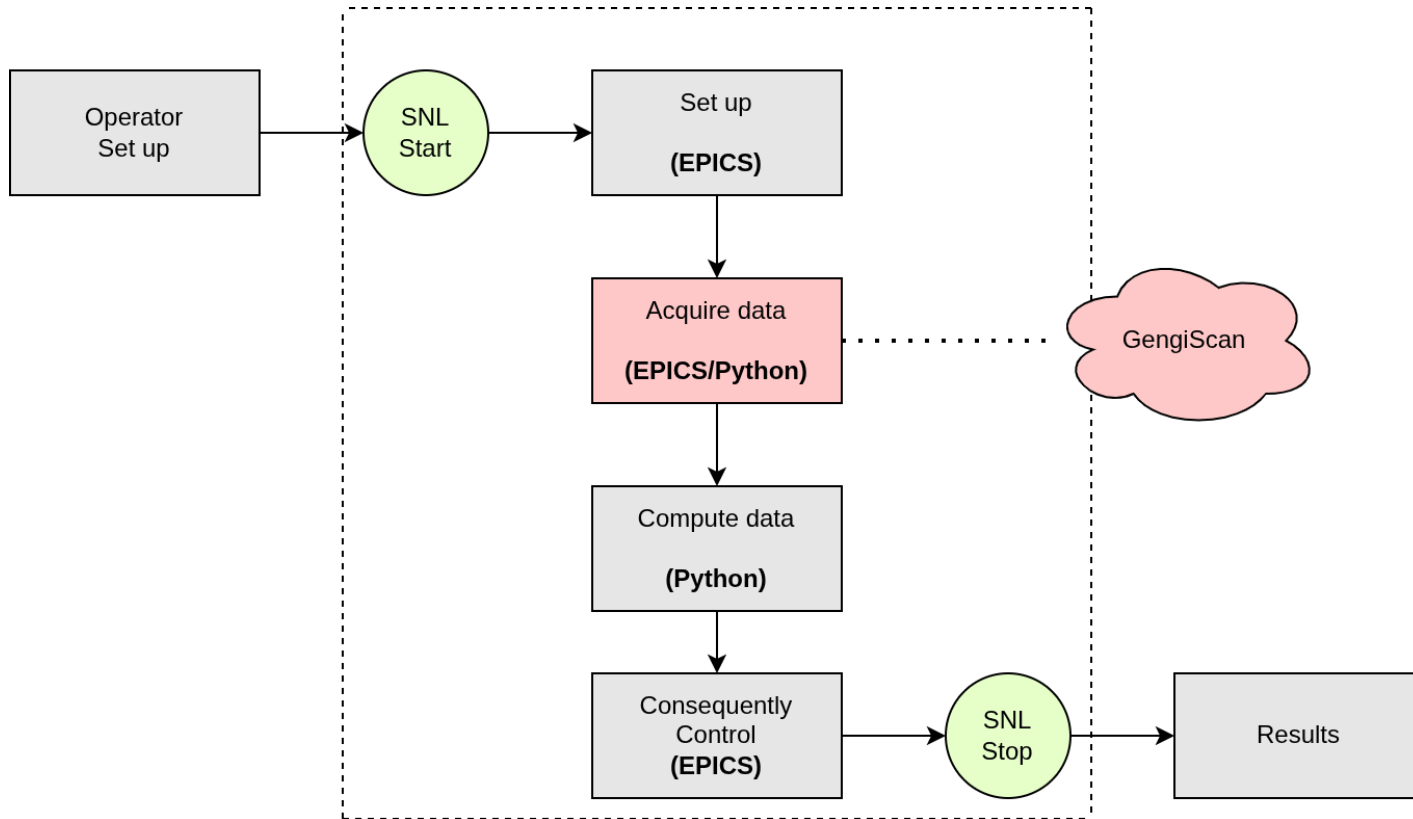
# SNL start: EPICS set up



- get initial values
- set GengiScan PVs
- regarding **ToF** for example:
  - detune the involved cavities



# GengiScan



# GengiScan

## Generic application to perform scan (half Python, half EPICS)

- Positioner: any actuator that affect the beam measurement
- Acquisitioner: system that acquire data for beam diagnostic
- Scan: a multiple-steps procedure used to save acquisitioner data depending on positioners position

see presentation “*GengiScan*” by *Stephane TZVETKOV*

GengiScan generates **formatted YAML** data file

- accelerator experts adapt their Python analysis scripts to this format

# GengiScan - ToF

Simple use case :

- get BPM phases
- store data (with operator input) into **formatted YAML file** →

```
context:
...
beam info:
...
  particles: proton
  constant PVs:
  floats:
    - name: SL-APP:DIA-TOF-1:CstFloat1Mes
      val: 1.3
      desc: estimated kinetic energy
      prec: 3
      egu: MeV
      inp: SL-APP:DIA-TOF-1:BeamEnergyContext NPP
    - name: SL-APP:DIA-TOF-1:CstFloat2Mes
      val: 0.2
      desc: kinetic energy range
      prec: 3
      egu: MeV
      inp: SL-APP:DIA-TOF-1:BeamEnergyRange NPP N
    - name: SL-APP:DIA-TOF-1:CstFloat3Mes
      val: 176.0
      desc: BPM frequency
      prec: 0
      egu: MHz
      inp: SL-APP:DIA-TOF-1:BpmFreqMes NPP NMS
    - name: SL-APP:DIA-TOF-1:CstFloat4Mes
      val: 2.0
      desc: number_of_Bpm
      prec: 0
      egu:
      inp: SL-APP:DIA-TOF-1:BpmNbRb NPP NMS
    - name: SL-APP:DIA-TOF-1:CstFloat5Mes
      val: -51.099998474121094
      desc: measured BPM 1 phase
      prec: 2
      egu: deg
      inp: SL-APP:DIA-TOF-1:BpmOffPhiMes NPP NMS
...
    - name: SL-APP:DIA-TOF-1:CstFloat8Mes
      val: 1.2384000000000002
      desc: position BPM 1 phase
      prec: 3
      egu: m
      inp: SL-APP:DIA-TOF-1:BpmPosMes NPP NMS
...
  strings:
    - name: SL-APP:DIA-TOF-1:CstString5Mes
      val: SL-MBT-MAG4:DIA-BPM-2
      desc: BPM 1 name
      inp: SL-APP:DIA-TOF-1:BpmNameTextIn NPP NMS
...

```

# GengiScan - EPICS Side

**GengiScanApp** is integrated into the **topMcs** which includes all the high-level applications.

Several templates needs to be instantiated, for example to set the **positioners** and **acquisitioners** :

```
...
## gengiscan core
file gengiscan_core.template {
  pattern { P }
    { "${P}_GSCAN" }
}
...
## gengiscan context constants
file gengiscan_context_constants.template {
  pattern { P
    IDX  DESC_FLOAT
    { "${P}_GSCAN" } 1 ""
    { "${P}_GSCAN" } 2 ""
    { "${P}_GSCAN" } 3 "BPM_frequency"
    { "${P}_GSCAN" } 4 "number_of_BPM"
    { "${P}_GSCAN" } 5 ""
    { "${P}_GSCAN" } 6 "first_downstream_BPM_Phi_Mes"
    { "${P}_GSCAN" } 7 "second_downstream_BPM_Phi_Mes"
    { "${P}_GSCAN" } 8 "upstream_BPM_Pos_Mes"
    { "${P}_GSCAN" } 9 "first_downstream_BPM_Pos_Mes"
    { "${P}_GSCAN" } 10 "second_downstream_BPM_Pos_Mes"
  }
  INP_FLOAT
  "${P}_GSCAN}BeamEnergyContext"
  "${P}_GSCAN}BeamEnergyRange"
  "${P}_GSCAN}BpmFreqMes"
  "${P}_GSCAN}BpmNbrB"
  "${P}_GSCAN}BpmOffPhiMes"
  "${P}_GSCAN}DownBpm1OffPhiMes"
  "${P}_GSCAN}DownBpm2OffPhiMes"
  "${P}_GSCAN}BpmPosMes"
  "${P}_GSCAN}DownBpm1PosMes"
  "${P}_GSCAN}DownBpm2PosMes"
  DESC_STRING
  ""
  ""
  ""
  ""
  "${P}_GSCAN}BpmNameTextIn"
  "${P}_GSCAN}DownBpm1NameTextIn"
  "${P}_GSCAN}DownBpm2NameTextIn"
  ""
  ""
  ""
  ""
  INP_STRING
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  ""
  TYPE_WAVE
  }
  }
  }
  }
  }
  "STRING"
  "STRING"
  "STRING"
  }
  }
  }
  }
  }
  }
  }
  }
}

...
## gengiscan positioners
file gengiscan_positioner.template {
  pattern { P
    IDX
    { "${P}_GSCAN" } 1
    { "${P}_GSCAN" } 2
  }
}
...
## gengiscan acquisitioners
file gengiscan_acquisitioner.template {
  pattern { P
    IDX  NELM  FTVL
    { "${P}_GSCAN" } 1 32768
  }
  "FLOAT" } { "${P}_GSCAN" } 2 32768
  "FLOAT" }
  ...
}
}
```

# GengiScan - Python Side

## Run as a daemon using **procServ**

```
# envfile
PROCSERV_CHDIR=/iee/tops/epics-user/topSaraf/topMcs/gengiScanApp/gengiscan/dist/

PROCSERV_PORT=2007

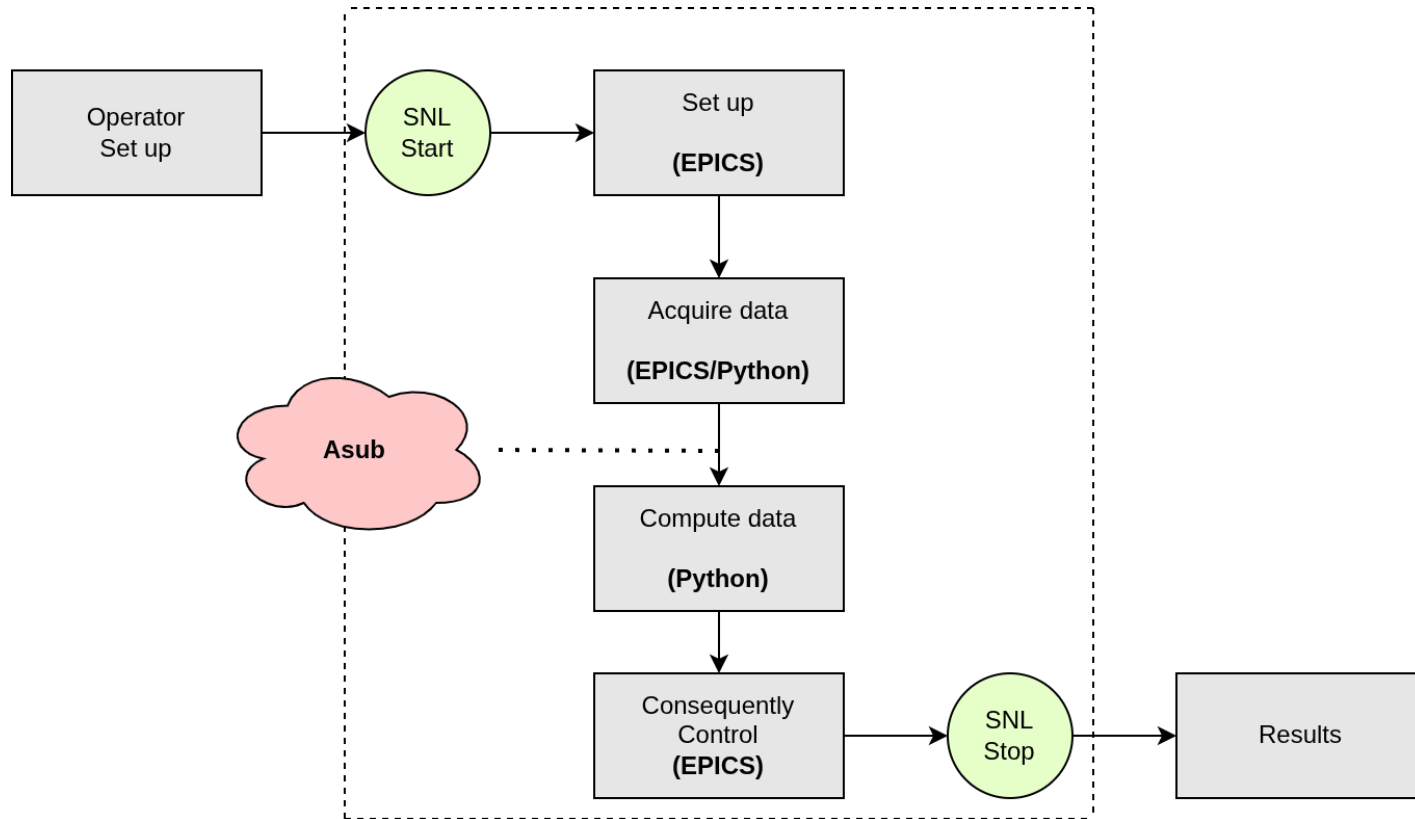
PROCSERV_COMMAND="./gengiscan -p=SL-APP:DIA-TOF-1: --camonitor-timeout=6000 --output-folder=/data/gengiscan/tof/outputs --output-backup-folder=/data/gengiscan/tof/outputs --log-folder=/data/gengiscan/tof/logs --log-backup-folder=/data/gengiscan/tof/logs"
```

## ProcServ wrapper using **envfile** and **systemd services**

Same **GengiScan** executable for all high-level applications :

- arguments are adapted, such as the PVs prefix

# Interface EPICS/Python



# Interface EPICS/Python - ToF - using Asub

SNL writes in PV → triggers **Asub** → executes Python executable

```
executeAnalysisScript(aSubRecord *precord)
{
    int tofexecute = *(epicsUInt32 *)precord->a;    // execute tof
    l4e_info(MyApp,"%s(): Start; \n", __FUNCTION__);
    if (tofexecute == 1) {
        l4e_info(MyApp,"Analysis the data");

        const char * command = "..../ToFApp/ToFAppPy/dist/tof ";
        system( command );

    }
    l4e_info(MyApp,"%s(): Stop; \n", __FUNCTION__);
    return 0;
}

epicsRegisterFunction(executeAnalysisScript);
```

**System command & relative path** → not ideal but reliable because:

- Irfu Epics Environment (IEE) running on standard SARAF machines
- executables managed by Continuous Integration / Continuous Development

# Main SARAF CI/CD

One job to get all **executables**, and **distribute them** at the right path:

```
...
get-generated-executables:
  image: ldisc-container-registry.extra.cea.fr/iee-allinone:latest
  stage: build
  needs: [get-submodules]
  script:
    - *job-common-whoami
    - ./CICD/get_generated_executables.sh --output_folder ./CICD/builds/
  artifacts:
    paths:
      - "CICD/builds/"
      - "**/dist/"
...
run-saraf-delivery:
  stage: deploy
  needs:
    - job: get-generated-executables
    artifacts: true # Python executable
...
```

For more details about the main SARAF CI/CD :

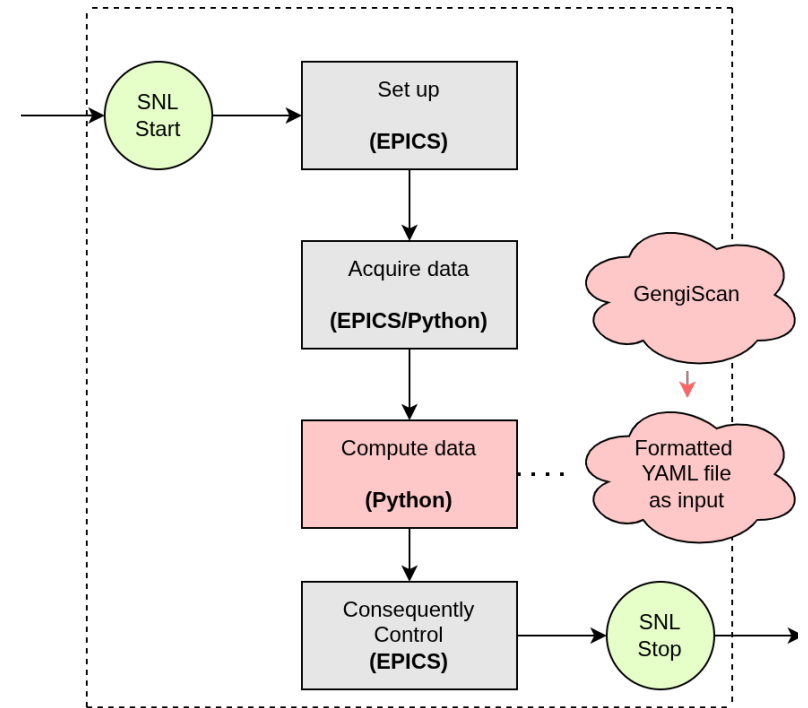
- see slide 8 of the presentation *“Remote integration and commissioning of an EPICS-controlled accelerator”* by Gabriel DESMARCHELIER

# Python analysis scripts

Written by **accelerator experts** :

- use the **formatted YAML** data file generated by GengiScan as input

High-level applications CI/CDs generate the executables of the Python analysis scripts



# Python analysis scripts - generating executable

Scripts are built as executables using **PyInstaller** and **Poetry**

```
# build.py

import PyInstaller.__main__
from pathlib import Path

HERE = Path(__file__).parent.absolute()
path_to_main = str(HERE / "../tofapppy/tof.py")

def install():
    PyInstaller.__main__.run([
        "--clean",
        "--onefile",
        "--hidden-import",
        "epics.clibs",
        path_to_main,
    ])
```

# Python analysis scripts - ToF - CI/CD

Scripts are built **automatically**

```
generate-new-executable:  
  stage: build  
  allow_failure: false  
  before_script:  
    - poetry install -v  
    - poetry run Python tofapppy/tof.py -h  
  
  script:  
    - poetry run build  
    - ./dist/tof -h
```

Then push on Gitlab package registry.

# Python analysis scripts - ToF - CI/CD: docker image

SARAF – IEE – *CentOs 7 (deprecated)* = specific environment



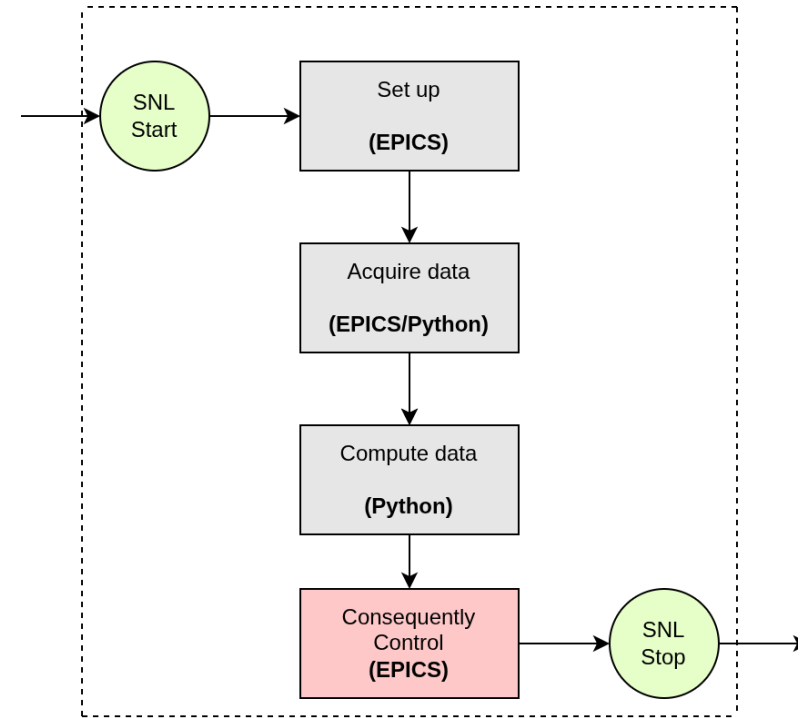
CI use specific docker images from a self-hosted docker registry:

default:

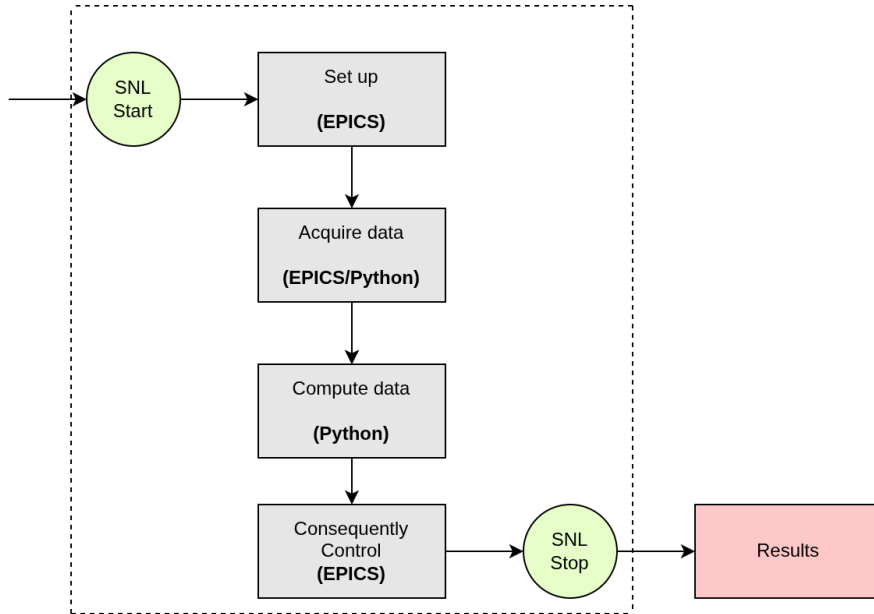
```
image: ldisc-container-registry.extra.cea.fr/iee-Python
```

# Before SNL stop

- Restore the original settings
- Use the result to configure equipments



# Results - ToF



The interface shows a 'Beam Energy Result' section with a green box containing '1.025 MeV' and a 'Push' button. To the right, the 'Result Context' section displays: Particles kind: proton; Selected BPM: SL-MBT-MAG1-DIA-BPM-1; BPM Frequency: 176 MHz; 1st downstream BPM: SL-MBT-MAG4-DIA-BPM-2; 2nd downstream BPM: SL-MBT-MAG6-DIA-BPM-3.

## AutOlog:

The AutOlog interface shows a log viewer with a list of events on the left and a detailed view on the right. The detailed view for a 'High-level application' event includes the following text:

A result has been processed by an high-level application  
The log creation has been triggered by the pv: SL-APP:DIA-TOF-1:BeamEnergyFinalResult, with value: 1.3336472466761486  
[Context]  
Time of Flight result (in MeV):  
• 1.3336472466761486  
At the BPM:  
• SL-MBT-MAG1:DIA-BPM-1  
Log created automatically by the application AutOlog



# 4. Status

# Status

Latest ToF updates :

- removed automatic detuning of the cavities via SNL
- fixed an error in Python analysis script

ToF has been fully **tested and validated with beam** remotely at the beginning of 2026.

# Status

Waiting for beam sessions for testing :

- BpmCalib
- EmitScan
- CavityPhasing

First version developed, requires simulated testing :

- BeamAligner

# Conclusion

- Accelerator experts appreciate Python scripts.
- The choice of this architecture facilitates collaboration between accelerator experts and EPICS developers/integrators.



irfu



**Thank you**

