PSI Center for Scientific Computing, Theory and Data

BEC Core

Ophyd Devices

BEC Widgets

The BEC Scanning Approach – Devices Get Ready!

Christian Appel[1], Mirko Holler[3], Jan Wyzula[1], Sven Augustin[1], Matias Guijarro[1], Xiaoqiang Wang[3], Andreas Menzel[3], Klaus Wakonig[1]

[1]Center for Scientific Computing, Theory and Data, Paul Scherrer Institute, PSI, Forschungsstrasse 111, 5232 Villigen, Switzerland
[2]Center for Accelerator Science and Engineering, Paul Scherrer Institute, Switzerland
[3]Center for Photon Science, Paul Scherrer Institute, Switzerland
christian.appel@psi.ch

What is the challenge?

16 different beamline with mostly similar devices, yet different requirements and expectations during operation.

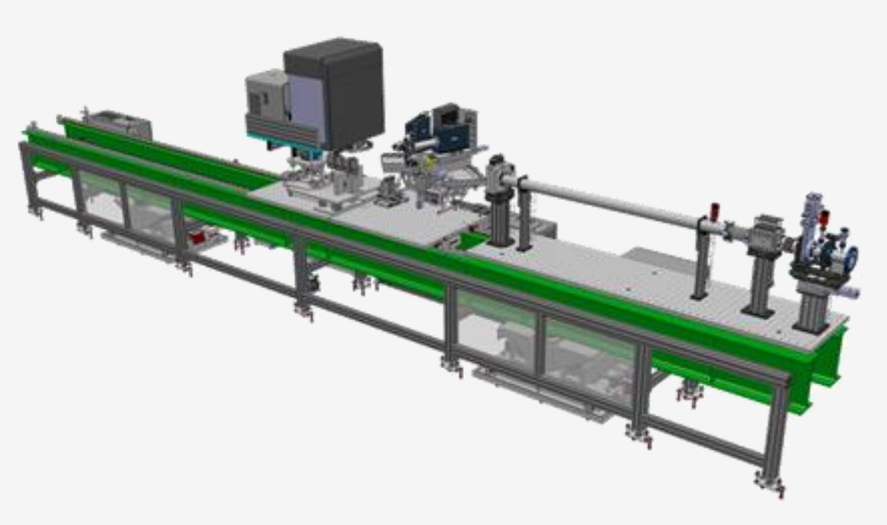
How do we avoid hard-coding beamline specific device logic in scans?

Ophyd Devices

Unified interface regardless of the underlying control layer and device type.

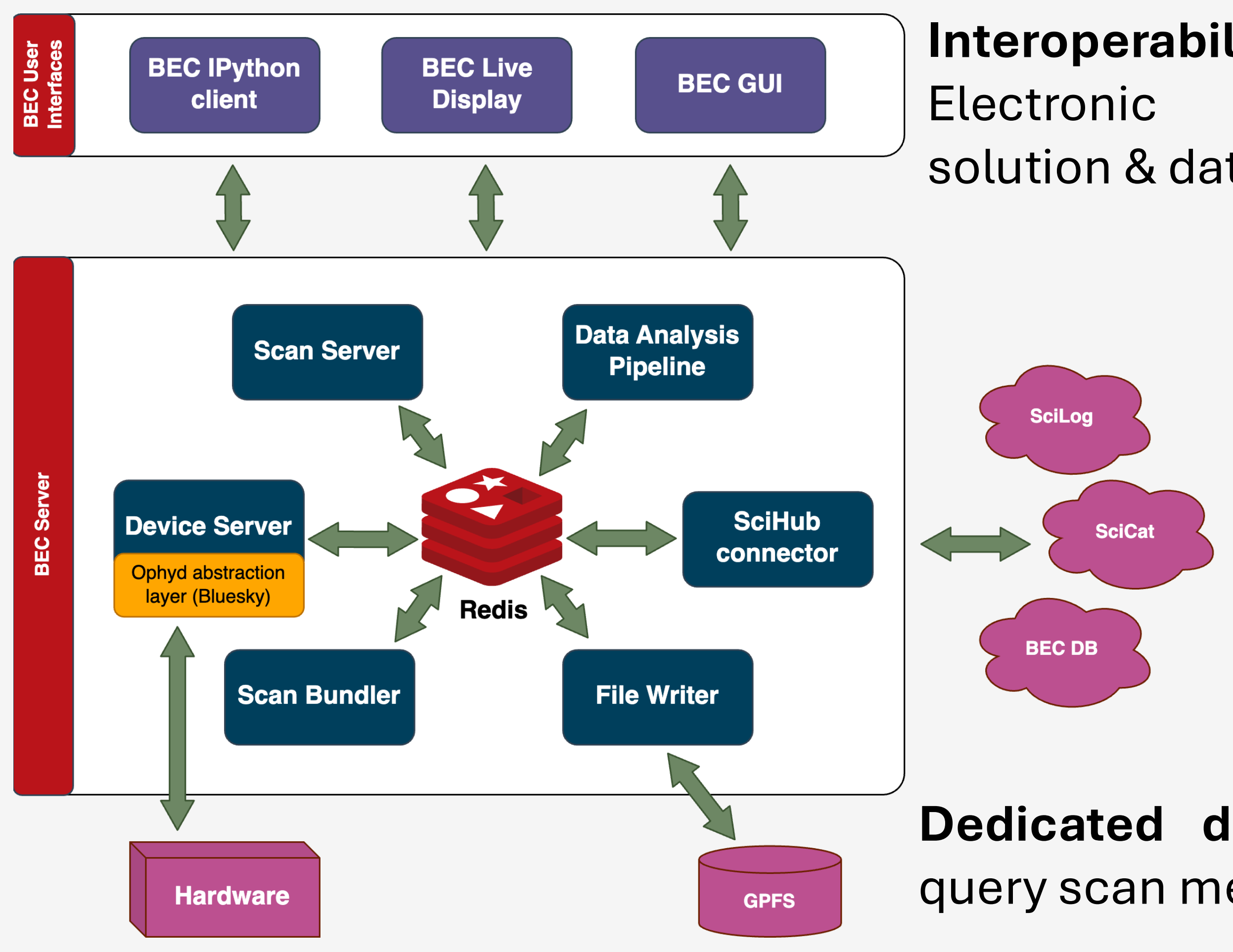
Extensive support of **EPICS** devices through **ophyd.**

Non-EPICS devices (i.e. REST, ZMQ or TCP communication) are integrated with the **same abstract interface**.



What is BEC?

BEC is a **B**eamline **E**xperiment **C**ontrol system with a **service-oriented architecture** for orchestrating and steering the experiment at research facilities.



Interoperability as key requirement: Electronic logbook, archiving solution & data processing pipelines.

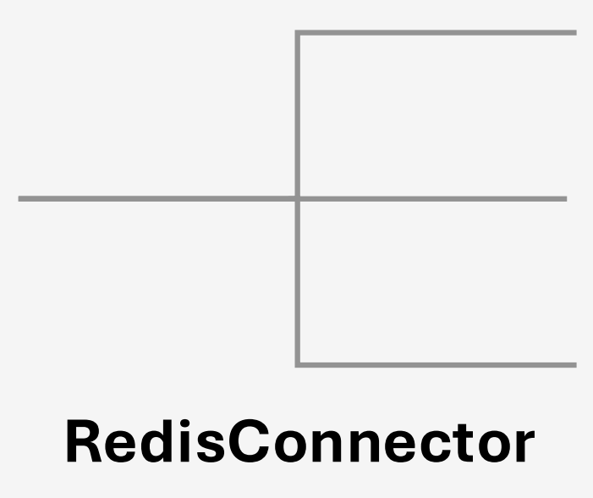
Dedicated database to store and query scan metadata for future usage.

BEC events

I. RedisConnector:
wrapper around redis-py

II. MessageEndpoints:
validate operations

III. BECMessage:
pydantic models



MessageEndpoints		BECMessages
scan_status(scan_id=...)	⟷	ScanStatusMessage(content, metadata)
file_event(name=...)	⟷	FileMessage (content, metadata)
device_read(device=...)	⟷	DeviceMessage(content, metadata)

Scan Hierarchy in BEC

open	Emit scaninfo viaScanStatusMessage
stage	Inform devices to prepare for the scan
pre_scan	Execution of time critical actions
"core"	Scan loop
complete	Report on data acquisition, success or failure
unstage	If necessary, revert/remove scan-specific logic
close	Close the scan, emit event that scan is done

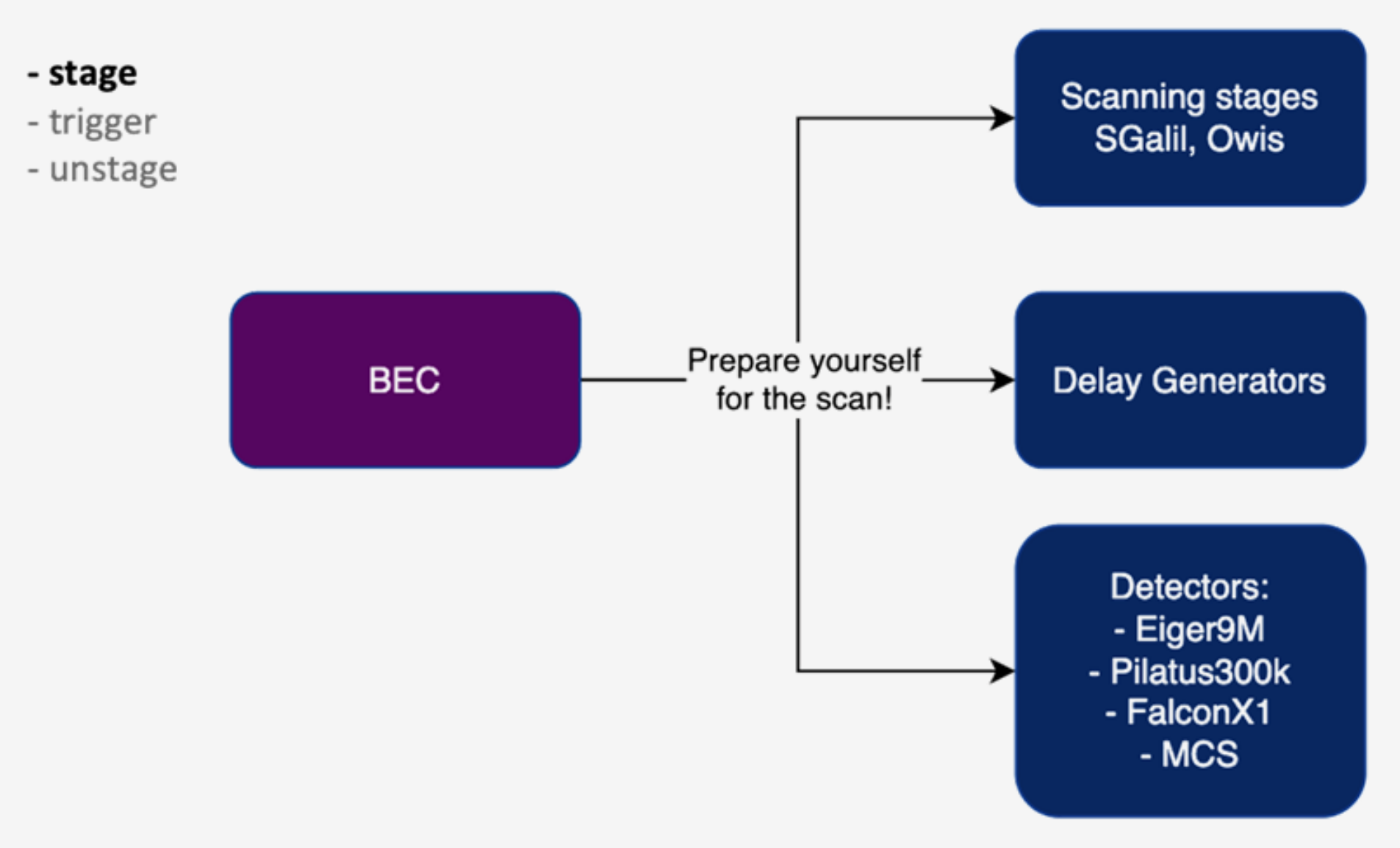
Conclusion

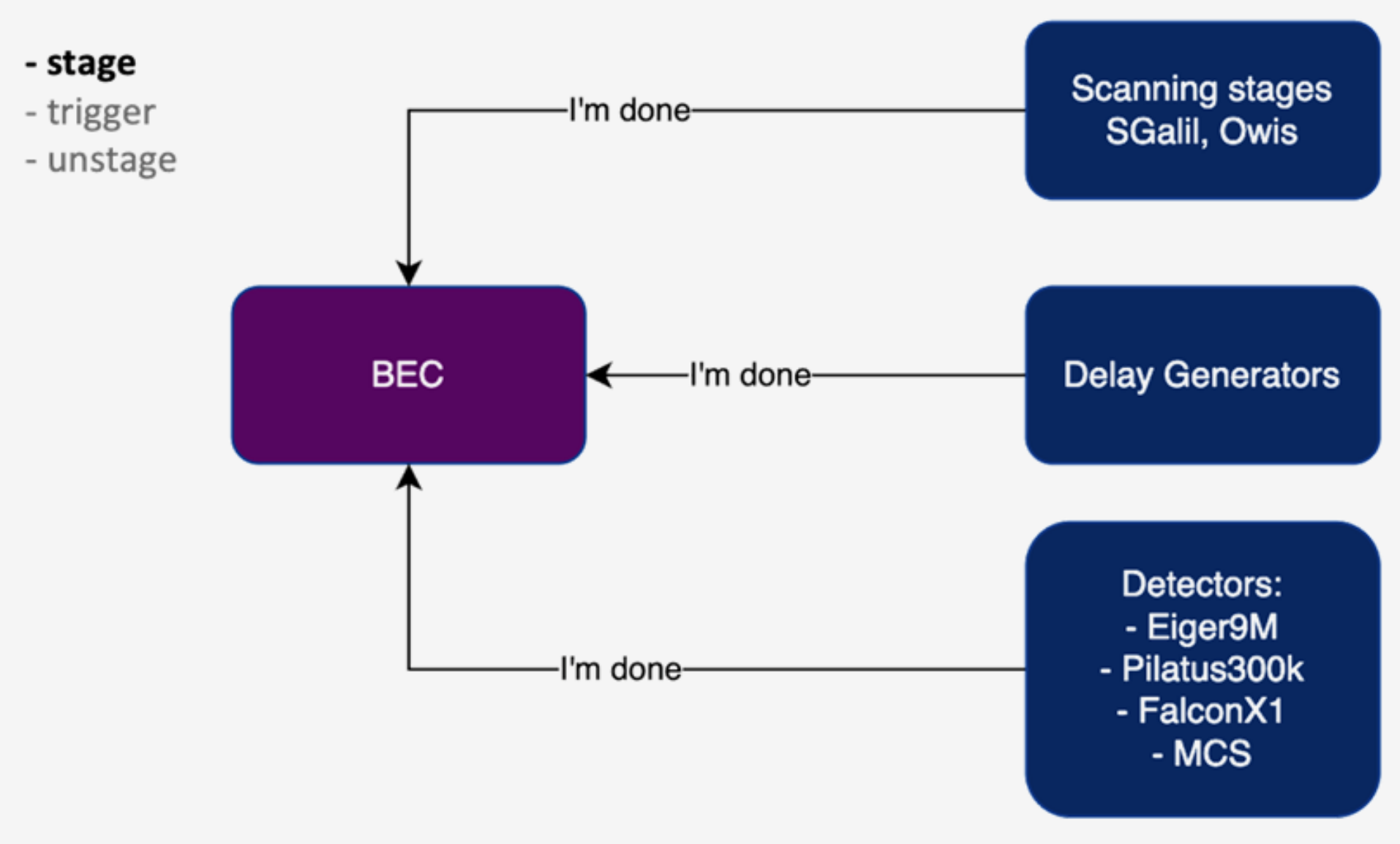
Unified interface for all devices

Disentangle device logic from scan logic

Flexible thanks to BEC's event system

Bootstrap Approach


- stage
- trigger
- unstage
BEC
Prepare yourself for the scan!
Scanning stages SGalil, Owis
Delay Generators
Detectors: - Eiger9M - Pilatus300k - FalconX1 - MCS


- stage
- trigger
- unstage
BEC
I'm done
I'm done
I'm done
Scanning stages SGalil, Owis
Delay Generators
Detectors: - Eiger9M - Pilatus300k - FalconX1 - MCS

Devices

- Detector control unit
- Temperature controller
- Trigger device
- Motor controller

+

Methods

- stage
- trigger
- complete
- ...

+

Custom Prepare Actions

Two type of scans: **step** and **fly**. Upon *stage, trigger, complete, etc.* **beamline-specific actions** are executed on the devices.

```python
def on_stage(self):
    self.exp_time.set(self.scaninfo.exp_time)

def on_trigger(self):
    if self.scaninfo.scan_type == 'step':
        self.trigger_pv.set(1)
    elif self.scaninfo.scan_type == 'fly':
        pass
```

All beamlines can share the scans despite different hardware or triggering schemes.