

# KaraboGUI: The Multi-Purpose Graphical Front-End for the Karabo Framework

Burkhard Heisen, Martin Teichmann, Kerstin Weger, John Wiggins  
European XFEL GmbH, Hamburg, Germany

Karabo is the new integrated control, data acquisition and processing framework developed for the photon beamlines at the European XFEL.

There is one graphical user interface for everything one can do with Karabo. This includes running and configuring devices, designing graphical interfaces, writing and executing macros.

The GUI allows visualization of slow control data and is capable to online monitor images provided by large 2D detectors during acquisition and processing steps, such as calibration and analysis.

The live navigation shows all running devices, their classes and the device server they are running on. Users can instantiate and shutdown devices.

Users can design scenes, graphical representations of the running system. Properties are dragged from the device's configuration to be shown in the scene.

With the device configurator the user can configure any running device, as well as pre-configure devices to be run. The configuration panel is autogenerated from the device parameters already known before instantiation.

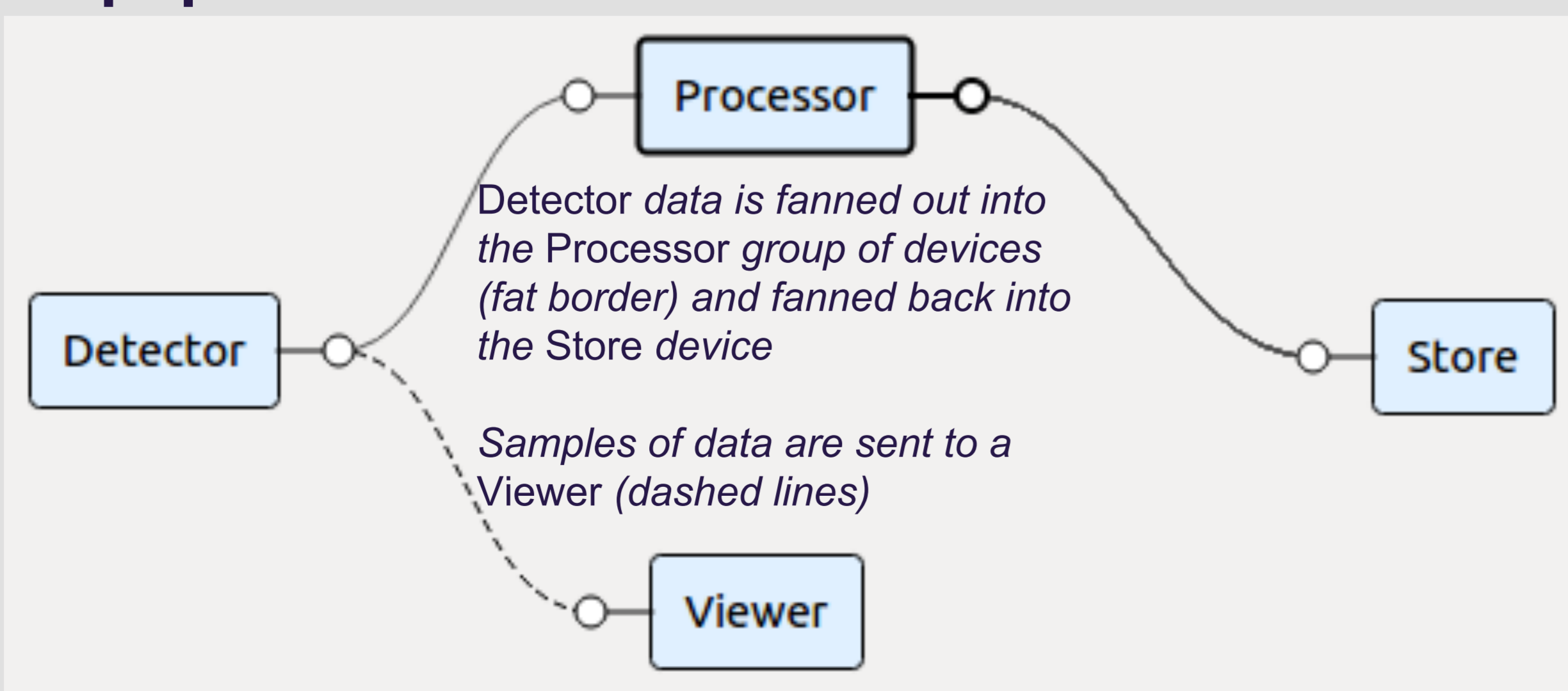
The screenshot displays the KaraboGUI interface for 'PHOEBE Vacuum Control'. It features a central scene with various components like pumps, valves, and detectors, each with control buttons. On the left, a 'Navigation' pane shows a hierarchical tree of devices and projects. On the right, a 'Configuration' panel lists parameters for a selected device, such as 'DeviceID', 'Progress', and 'State'. Below the scene, a 'Log' widget shows a table of messages with columns for ID, Date and time, Message type, Instance ID, and Description. An orange callout box points to the configuration panel with the text 'drag properties and slots to scene'. Another orange callout box points to the scene with the text 'Different widgets can be selected for a property'. A third orange callout box points to the project list with the text 'drag devices to scene for workflows'. A fourth orange callout box points to the data visualization scene with the text 'A data visualization scene'. The data visualization scene includes a 2D heatmap and a 1D line plot, with a callout box stating 'Data can be retrieved from storage by dragging to the past.'

Projects are containers persisting everything needed for a specific task:

- A list of all devices that need to run, and their desired configuration
- The scenes graphically representing the task
- Specific configurations to be applied to devices
- Macros to program repetitive or sequential tasks
- A list of device parameters to be monitored for the task

- The logging widget logs all messages broadcast in Karabo.
- An IPython based console allows to use the macro language interactively.

## Data pipelines



Karabo devices can communicate high-bandwidth data via direct links. These pipelines can be designed in the GUI as well.

## Macros

```
from karabo import *

class Scan(Macro):
    start = Float(description="Start position")
    stop = Float(description="End position")
    steps = Int()

    @Slot()
    def run(self):
        """Start the scan"""
        with getDevice("motor") as m:
            m.targetPosition = self.start
            m.move()
```

Macros can be edited directly in the GUI, but then run on a dedicated macro server. They are class-based and are a special form of devices.

They are to be used for specific tasks only, generic tasks should be implemented as devices.