
MAX IV

High-speed data streaming at MAX IV

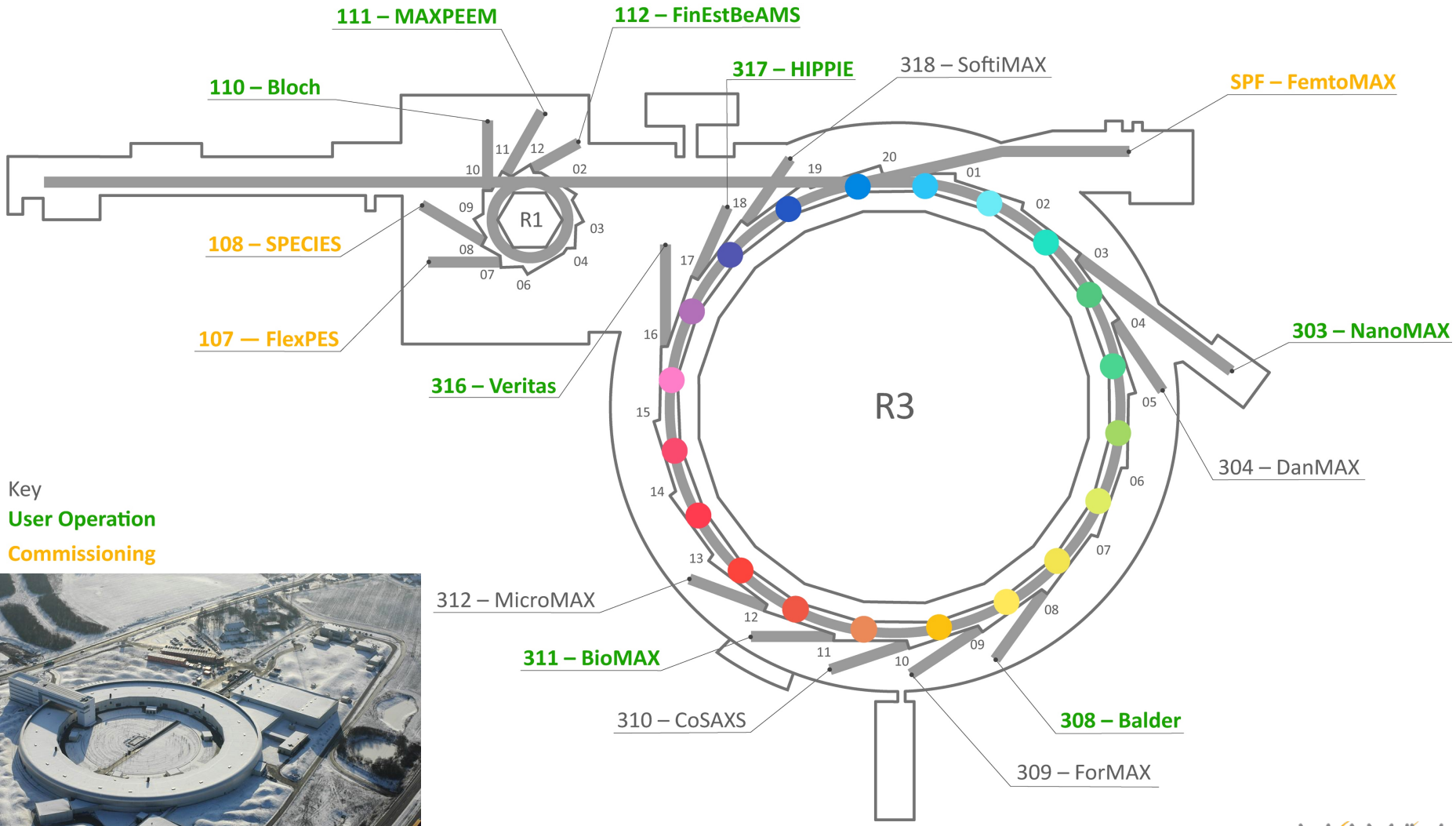
Emil Rosendahl, KITS

BACKGROUND

MAX IV

- Synchrotron facility
- 4:th generation x-ray light source
- Microscope
- Beamline ~= Experimental station

MAX IV



Key

User Operation

Commissioning

DATA ACQUISITION

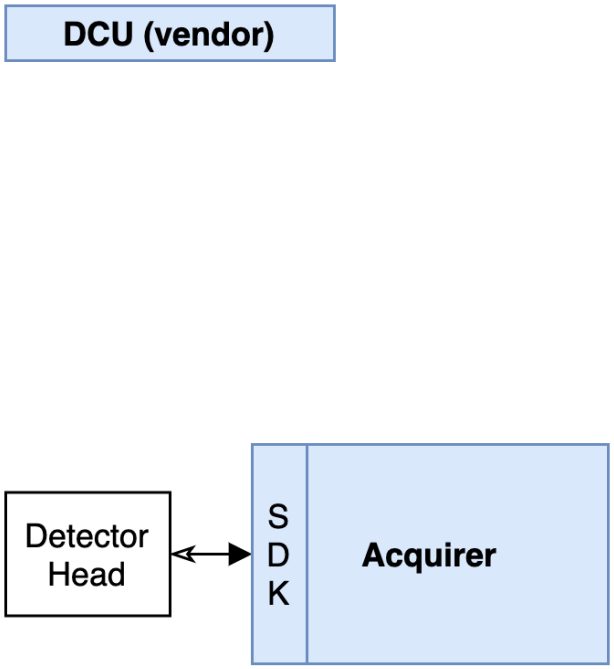
Priorities

- Prio 1: Data on disk
- Prio 2: Online analysis & Live view

High Speed - Streaming

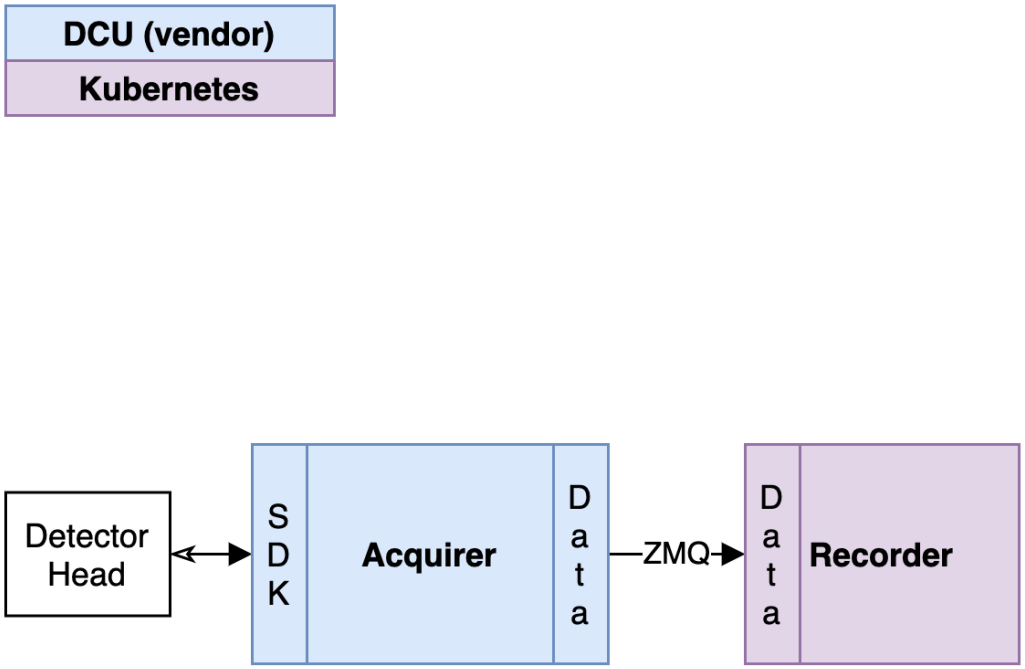
- Complex behaviours
- Faster feedback
- Custom HDF5 file
  - Hello NeXus!

Pipeline - Acquirer



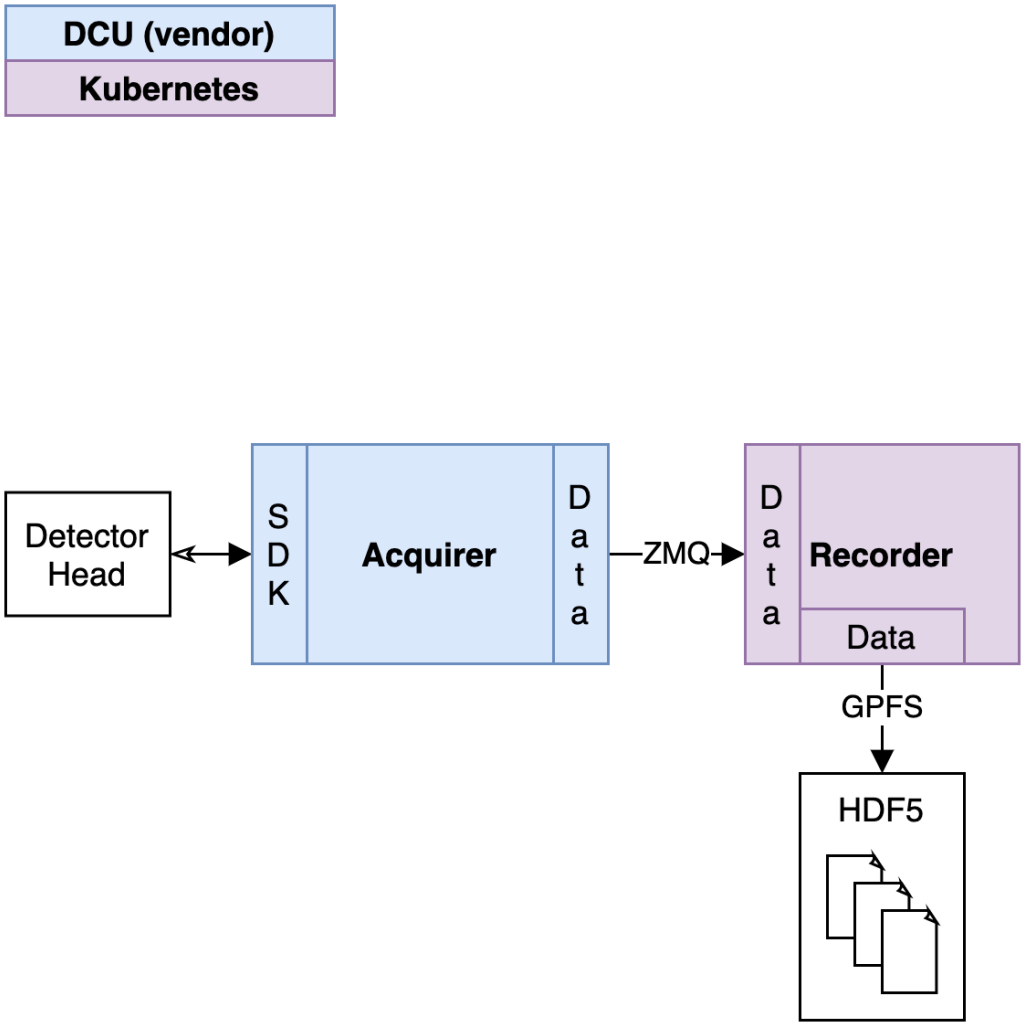
- Connects to hardware
- Vendor specific
  - Eiger & Pilatus from Dectris
  - Zyla & Balor from Andor
  - Dhyana from Tucsen
- Performance critical (ish)

Pipeline - Recorder

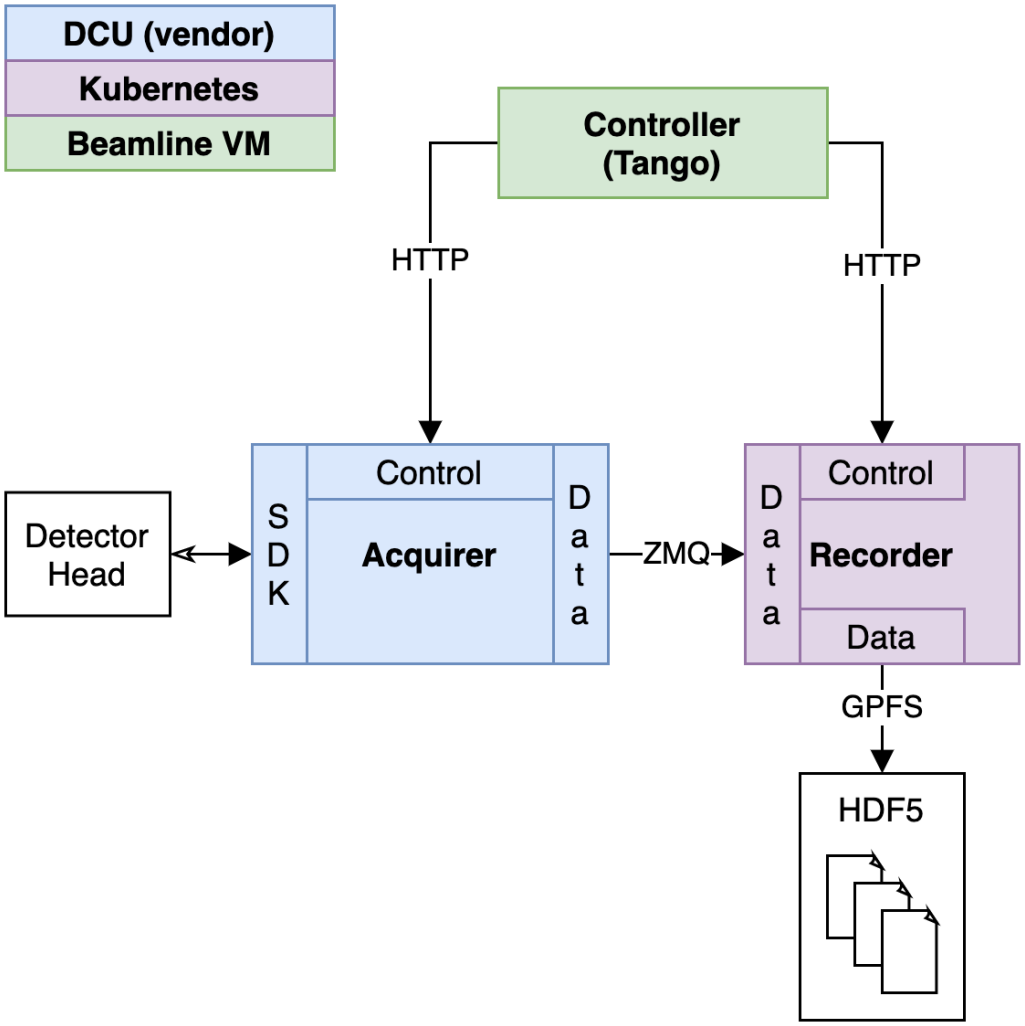


- Sophisticated HDF5 writer
- Stream in, file out

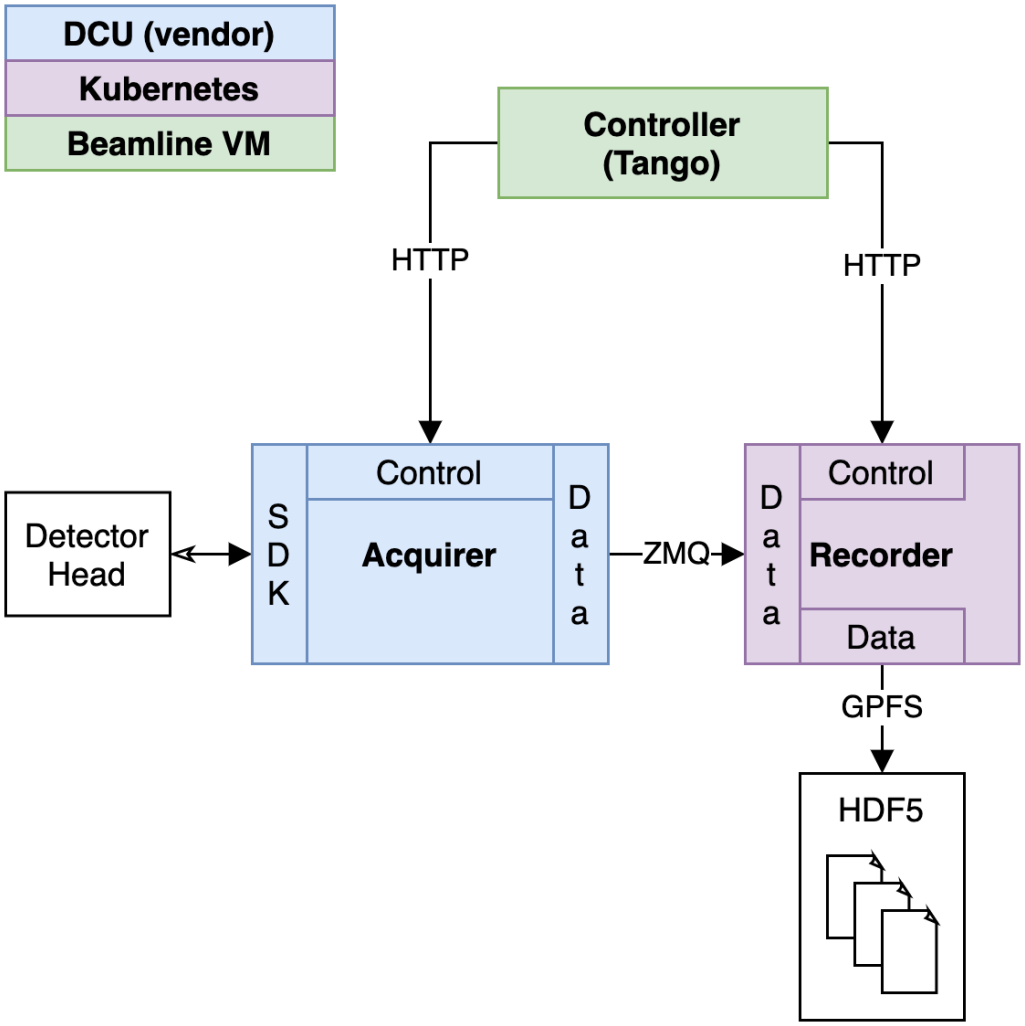
Pipeline - Recorder



Pipeline - Controller

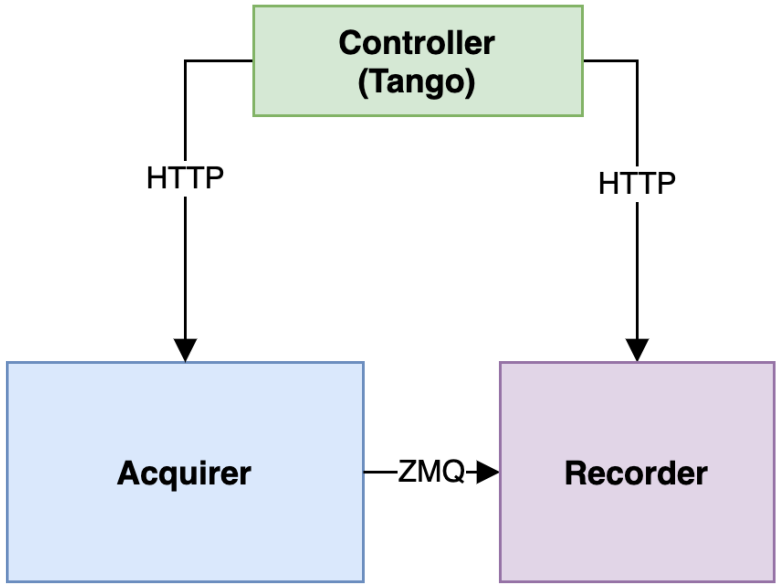


Pipeline - Controller



- A Tango device
- Controls two things
  - Acquirer
  - Recorder
- Combined state
- Monitors the acquisition

Pipeline - CAR



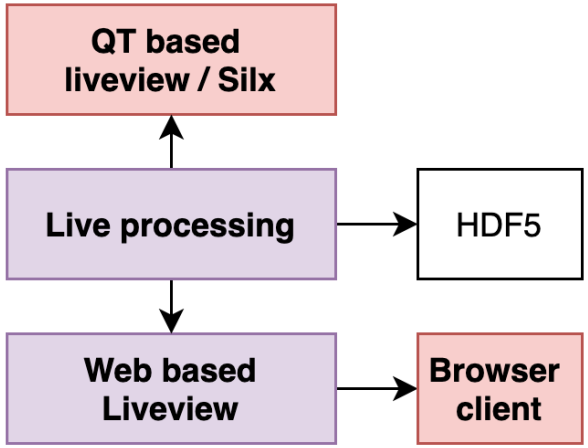
Priorities

- ~~Prio 1: Data on disk~~
- Prio 2: Online analysis & Live view

Pipeline - Live view and Processing

Kubernetes

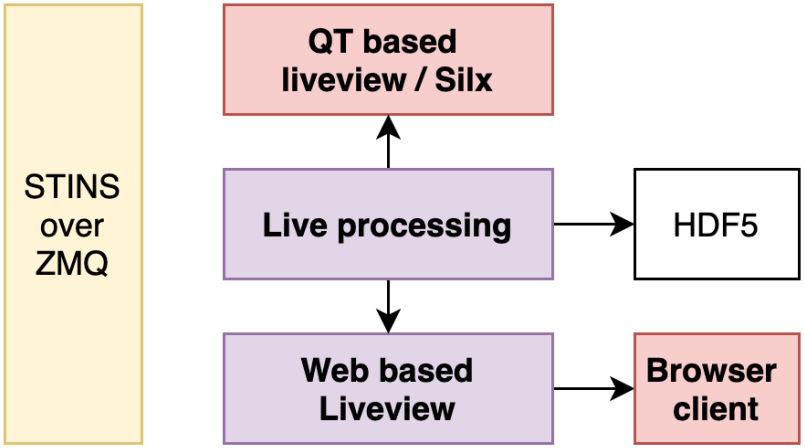
Beamline Computer



Pipeline - STINS

Kubernetes

Beamline Computer



- Protocol
- Interoperability

Pipeline - STINS

- Protocol
- Interoperability

Pipeline - STINS

STINS - Stream meta data

A protocol for a stream meta data structure.

Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 (see "Key words for use in RFCs to Indicate Requirement Levels").

Goal

Meta data from detectors come in a variety of flavours and structures, with varying amount of details. The goal of this document is to have a standard structure of meta data in order to provide the possibility of interoperable software.

The goal is NOT to dictate how the meta data is carried over a wire or how it's written to disc (this leaves ZMQ, HTTP, TCP, HDF5 etc out of the discussion). This topic will be on follow-up proposals (for example 'STINS over ZeroMQ').

Basic structure

The meta data MUST be structured as a JSON object. All top-level keys (or fields) are reserved for this specification. Custom keys MUST NOT be used in the top-level or in structures defined by the specification. This ensures that the standard will be able to be expanded without conflicts with implementations.

The meta data MUST come in the context of a message. This message is part of the stream, and the job of the meta data is to identify and describe the message.

Stream Messages

There are three stream messages. They all share the same basic structure, mandatory and optional keys.

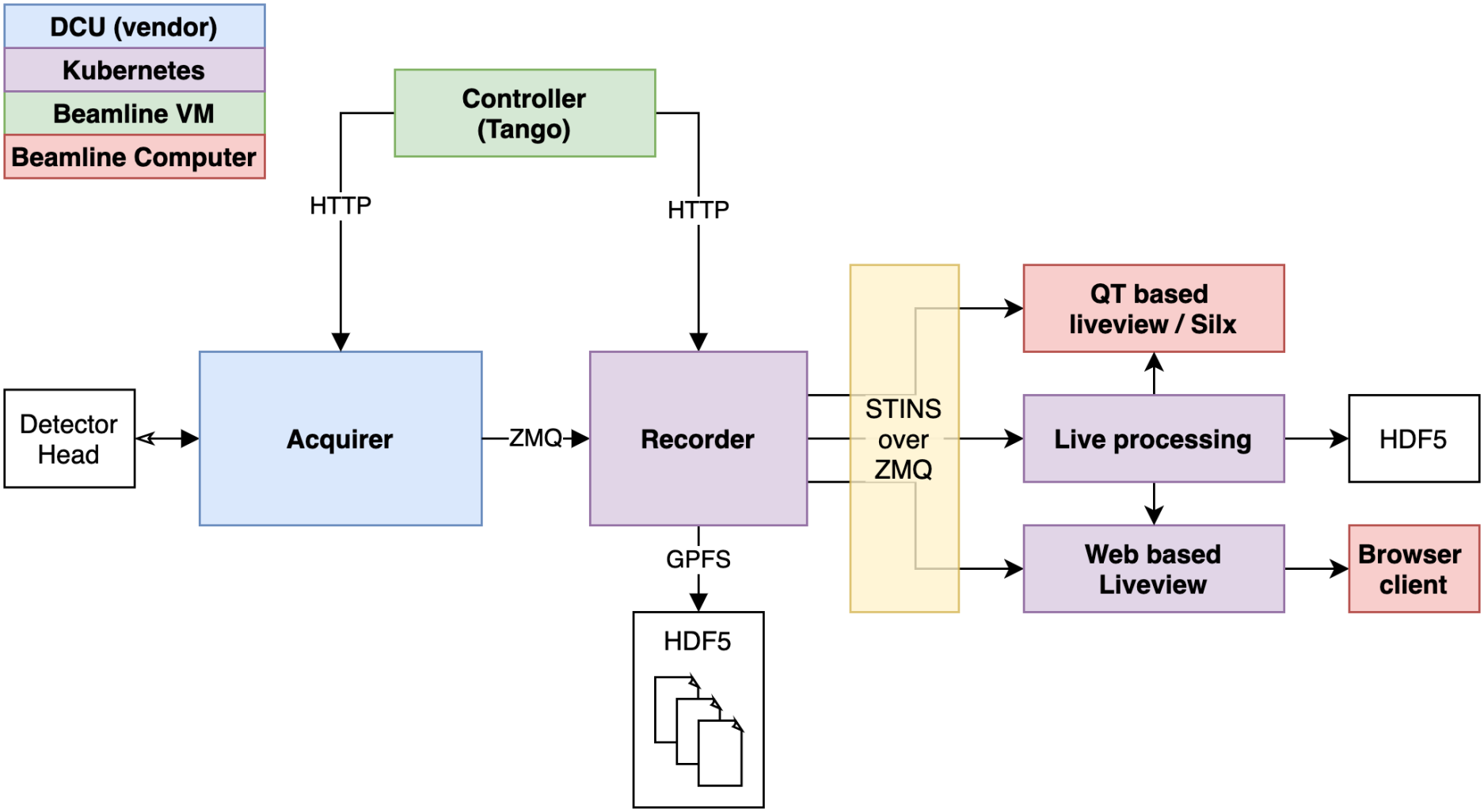
- Series Start
- Data
- Series End

Mandatory keys

Every meta data structure MUST include the following key(s).

- `version` : A number unique for the version of this standard. The value MUST be 1
- `message_type` : A specific key to define the type of message it is. The potential values are a strict set of strings (see each message type).
- `message_id` : A number that MUST be incremented with *every* stream message. It MAY be reset on software restart.

Pipeline - STINS



Infrastructure

- New network
  - Exclusive for Data Acquisition
  - Reliable performance
- Kubernetes
  - Resource configuration and distribution
  - User rights management

Questions?

emil.rosendahl@maxiv.lu.se