Configuration in the DUNE DAQ

Gordon Crone

University College London

DUNE UK meeting July 2023

Introduction

Requirements

The CCM is required to provide functionality to prepare, archive, validate and deliver configuration data, including the definition of required partitions.

- An interface for interaction with users, via command line and GUI, with equivalent functionality.
- A means by which DAQ application and detector component experts may specify the structure and constraints which describe the relevant configuration information.
- The configuration system shall provide a configuration generator/editor, capable of creating or loading DAQ system configuration data, modifying configuration data, and storing versioned configuration data.
- The configuration system shall provide a mechanism for checking the validity and consistency of configuration data with the DAQ and detector components.

Current configuration mechanisms

Configuration generation `daqconf`

- Individual packages provide moo schema (jsonnet) files
- `daqconf` suite of python scripts generate json files for every application in the DAQ using mixture of schema defaults, configuration time supplied vales and values coded in the Python of daqconf
- Tweaks made after configuration by editting json files lost next time a set is generated
- Scalability?
- Consistency?

Connectivity Service

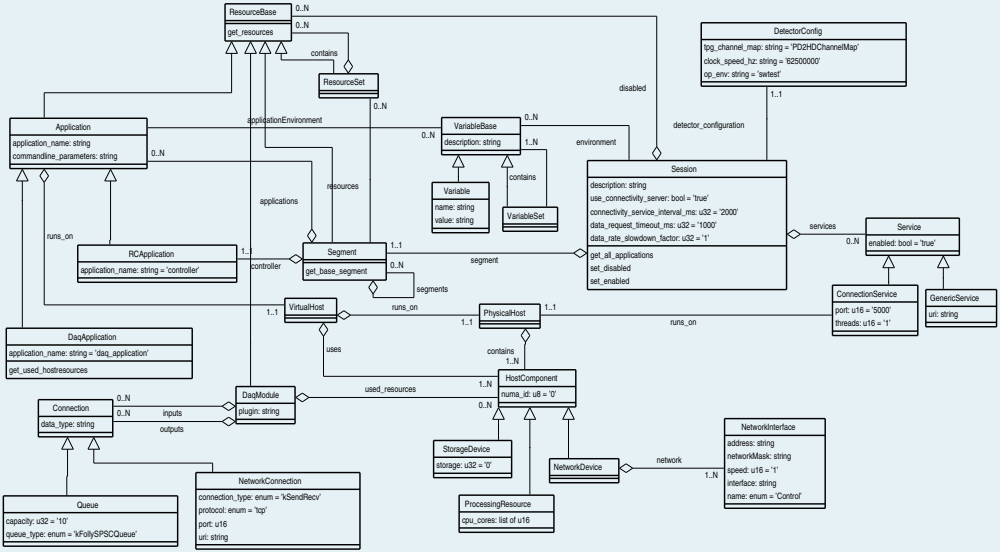
- Configuration complicated by static assignment of network ports
- Connectivity Service introduced this year
  - Listening apps let OS assign port and publish their connection details
  - Connecting apps look up connection info to find port to connect

The future of DUNE DAQ Configuration – OKS?

- OKS is the configuration database used in ATLAS
- Ported to the DUNE DAQ environment by John Freeman
  - See his excellent introduction from the CCM meeting of the 8th Feb

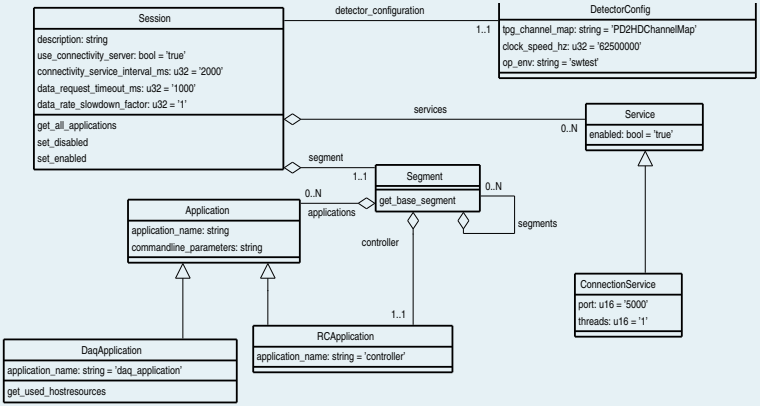
- Object oriented schema with inheritance between classes
- Relationships between objects allow common configuration objects to be shared among several objects rather than copied
- Schema and configuration (data) stored in a few XML files.
  - Can be generated but not intended as part of the normal work flow
- GUI editors provided for both schema and data files
- Application code accesses configuration through a 'data access library' `dal` which supplies getters for all attributes and relationships
- C++ code generated from the schema, Python bindings available
- dal `methods` allow manipulation of configuration at run time

‘Complete’ view of core Schema



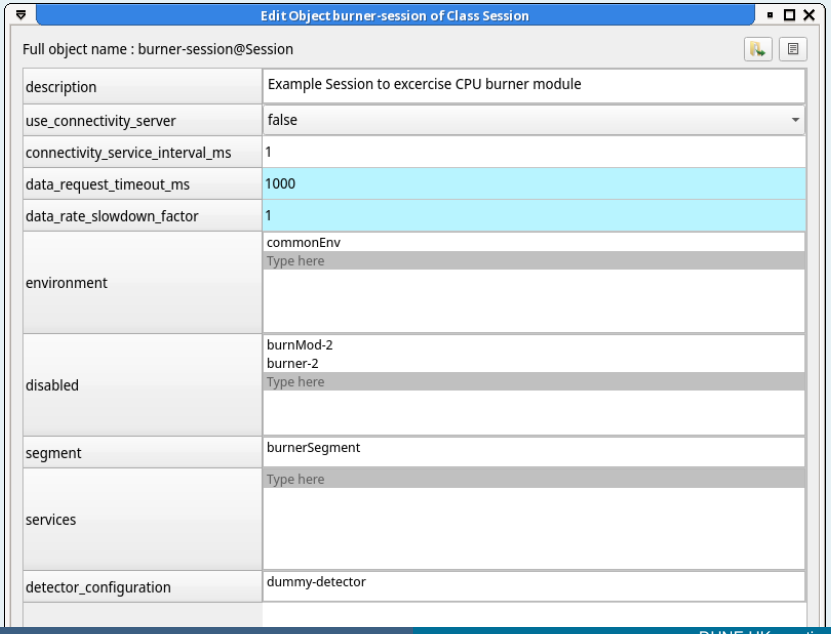
- Session is the top level. Applications are described in Segments

Session view of Core schema



- Session has some common attributes and a link to a single Segment
- Segment is a container for a logical grouping of Applications e.g. all the PDS readout or all the APA readout
- Segment has a `segments` relationship to itself so Segments can be nested

Editing Session



Status and plans

Current Status

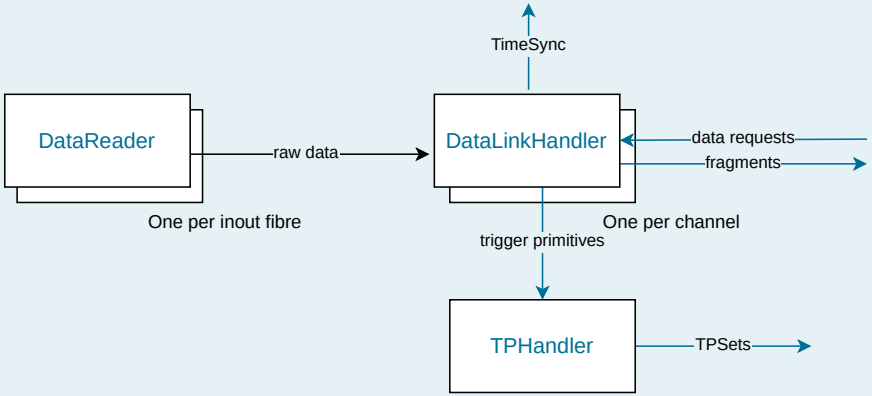
- OKS packages available in regular nightly build and will be in release 4.1.0
- Core schema down to the Application level in nightly and elease 4.1.0
- Should allow development of an OKS driven run control
- Now working on designing schema for readout applications

Future plans

- Extend core schema down to module level for Release 4.2.0
- Introduce DAL into `appfwk` and modules for Release 4.3.0

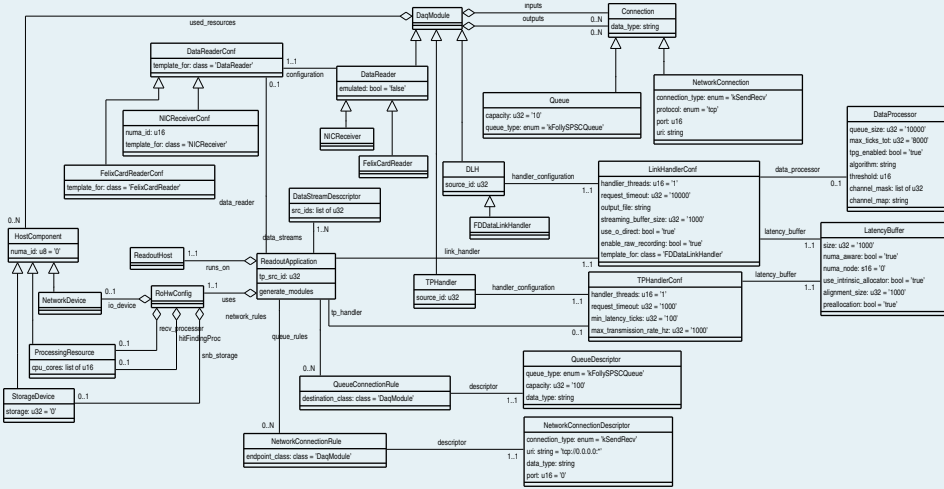
Backup Slides

Readout Application



- Multiple DataReaders and multiple DataLinkHandlers
- Multiplicities and connections between DataReaders and DataLinkHandlers can be inferred from readout map
- Don’t want to specify again in OKS configuration

View of readout Schema



- No relationship from ReadoutApplication to DaqModules
- All required DaqModules generated on the fly by the `generate_modules` method
  - Driven by config objects, code just knows how to connect DataReaders to DataLinkHandlers