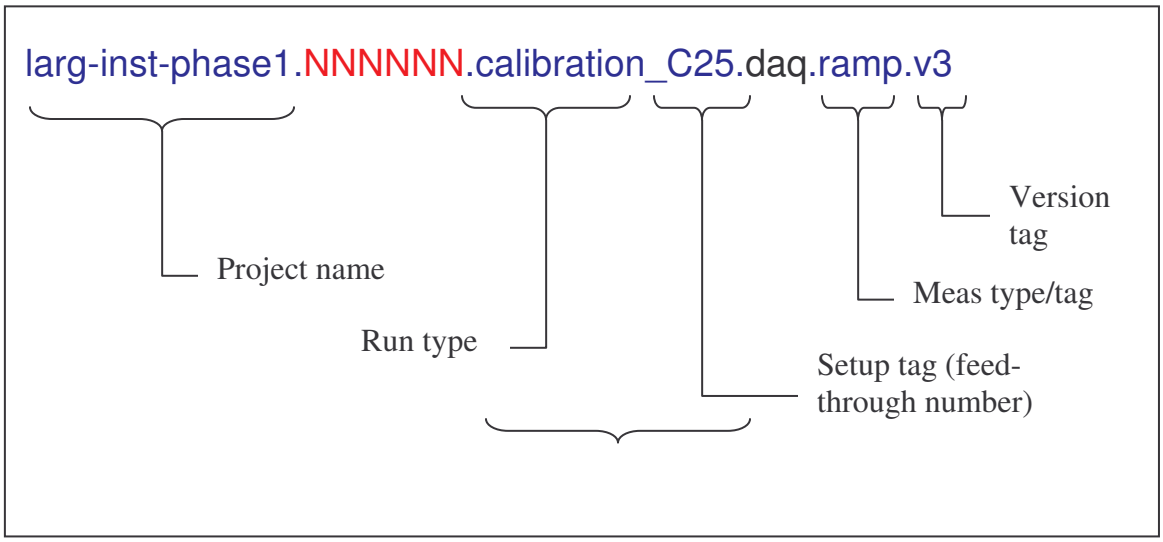
P.Gorbounov 17 January 2006
ATLAS SWING Internal Note

# Thoughts on meta tags for ATLAS commissioning data

Why did the s/w integration group initiate the discussion of meta tags in TDAQ? Answer: something was missing for the automatic basic registration of raw data set for DDM, the ATLAS Distributed Data Management system. The memo describes the "chain" which had lead us to TDAQ.

1. There is a wish to use DDM for future ATLAS data distribution at all levels: simulation, raw data, reconstruction data.
2. We want to start using DDM for commissioning data organization to exercise the existing tools and develop DDM-oriented infrastructure to manage the data description (metadata). We looked at what is needed for
   - (possibly, automatic) forming of DDM datasets (DS) – collections of files produced "under the same conditions";
   - DS cataloging and search.
3. DS formation includes composing the *DS name*and compiling the list of constituent files (runs). The DS name is composed of [1]:
   - "project name";
   - "the DS number", *NNNNNN*, unique within the project;
   - free short "physics reference" whose function is only to comment the DS number;
   - "transformation step", which in case of raw data is `daq`;
   - "format-type"
   - version

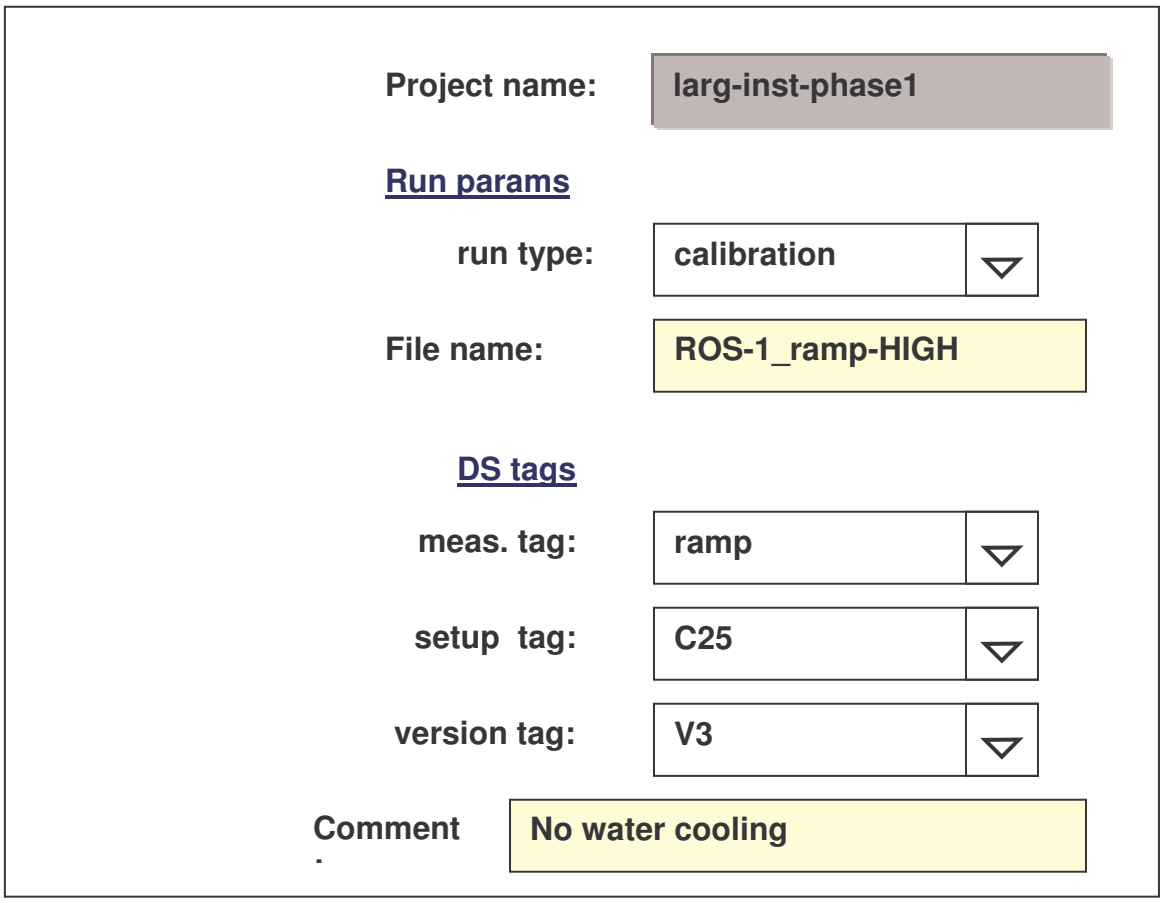
   Let us consider an example, inspired by LArg phase1 commissioning (front-crate installation):



This DS will consist of 3 runs, each consisting of 3 files – 9 files in total. In order to form the DS, I need:

- to consult the paper or electronic logbook (both are not always garanteed to be filled correctly and in time);
- enter manually the DS name;
- enter manually the list of runs belonging to this DS.

This all would not be needed, if in the RC panel, at a run start, I had something like that:




This differs from the panel in the current TDAQ by the following:

- no irrelevant attributes “Beam type” and “Beam Energy (GeV)”
- a “permanent” (easily *configurable*) “Project name” field
- three new combo-boxes to select three “DS tags”: “measurement”, “setup”, “version”. They replace “Beam Type” and have to be *configurable*: as a minimum, the lists of possible project-specific values will be defined by a run planner (project manager).
- A very desirable free “Comment” field, not needed for the basic DS formation/registration, but useful for cataloging and automatic bookkeeping.

What shall I do with these tags? Assuming that they will be passed to the headers of the raw files recorded by EventStorage and to the CondDB, I will retrieve them from either of these two sources (useful redundancy for the case if no CondDB is written during the data taking!), and

- automatically add to the same DS all consecutive runs having the same run type and the set of DS tags, for example:

```
calibration   ramp    C25   phase1
… …
calibration   delay   C25   phase1
… …
calibration   pedest  A31   phase1
… …
calibration   ramp    A31   phase1
… …
```

- automatically enter these tags into metadata DB (eg AMI);
- automatically generate the DS name, for example:

  "project_name".NNNNNN. "run_type+setup".daq."meas"."version"

4. One can argue that all needed tags could be "encoded" in the existing "file_name" meta-field. True, but this field is intended for another purpose. In addition, it will be difficult to ensure that all commissioning teams follow the same rules on how to generate it. The goal of ATLAS integration is to have a common platform for many historically different "cultures" matured in different detector teams. To try to unify these "cultures", e.g. by introducing a common standard for electronic logbooks, is hopeless. On the other hand, TDAQ *is* the framework containing *common* features. The transition from the TB to Commissioning stage, represents a good occasion to re-consider what we should have in the RC panel and what – in the file headers. This had been done in the past, in anticipation of TB and CTB [2,3].

   Why *three* additional tags? It is, probably, the bare minimum for the potential diversity of commissioning DSs . The "setup" field complements the rigid pre-defined "Run type" field, the other two are just needed to generate a correct DS name. The fixed number of tags will be straighforward to implement in the IS (by extending the RunParams object) and in EventStorage.

   There is no guarantee, of course, that in future *more* tags won't be requested. Therefore, it could be wise to anticipate a *fully configurable set of tags*, with not only lists of values, but the tag names and the very number of tags (if more than 3) to configure by the project manager. This will be a bit more tricky and longer to implement (see Appendix A), but all pre-requisites seem to be available.

5. New meta tags and the offline software. Probably, the change in EventStorage format can be made transparent for object converters, simply by retaining the old fields and appending the new ones. In my opinion, the use of meta-information for the analysis should be prohibited – it must use CondDB, full stop. On the other hand, small stand-alone applications (for example, a cataloging agent) will be very easy to adapt to any EventStorage format variations.

**References**

[1] ATLAS Data Set Definition, ATLAS s/w integration group Internal Note 2005-10-29, https://uimon.cern.ch/twiki/pub/Atlas/SoftwareIntegration/datasets.pdf

[2] M. Caprini et al., Run Parameters for the ATLAS DAQ/EF Prototype-1, Dec 1999 http://atddoc.cern.ch/Atlas/Notes/146/Note146-1.html

[3] S. Gadomsky Meta-information in the data files written by DataFlow s/w in 2004 beam test, Jan 2004, http://cern.ch/szymon/DataFlow/FileFormatProposal.pdf

**Appendix A:**

**Exuberant features: fully configurable meta-tag sets, run commentary field , keywords**

To implement variable tag sets in IS and EventStorage is more tricky than the fixed sets, but seems to be doable. For example, these tags could be made independent objects in the IS (not part of a common static structure), with the names having a common prefix, e.g. `runtags_`. EventStorage will retrieve these attributes by using the regexp search by name and put them all into one long and easy-to-parse string of the form:

tag=value, tag=value, …

The mechanism for storing variable-length strings in file header is available in EventStarage (cf. file name tag).

(unfinished)