RDA P10: A Generic Approach for PID Kernel Information Profiles

PID Kernel Information

Ulrich Schwardmann

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)

Am Fassberg, 37077 Göttingen
ulrich.schwardmann [at] gwdg.de

21 September 2017, Montreal
Abstract Features of PID Kernel Information Types (PID-KIT)

PID-KIT should:

- **(A) enable fast decisions** in workflows without data retrieval
  - which implies some simplicity
- **(B) should describe unchanging features** of the digital object
  - because they can only be changed by the PID admin
- **(C) should be given by grounded definitions**
  - for instance described by (a set of) profiles
  - that rely on well defined types
- **(D) should cover the use cases** in mind
Discussion about PID Kernel Information Types

we had three directions:

- naming allowed PID InfoTypes (expert naming)
  - was often the implicit common sense in earlier discussions
  - but: does geolocation belong to the PID-KIT?
  - is probably too restrictive,

- describing PID InfoTypes by semantical requirements
  - Beth: PID-KIT should be structural metadata
    - describes the intellectual or physical elements of a digital object
  - FAIR:
    - PID-KIT should help to find, access and reuse data in an interoperable way

- describing PID InfoTypes by syntactical requirements
  - my approach: see below
  - semantics becomes community dependent
The definition by **structural metadata**

- is this really *well defined*?
  - is `nextVersion` or `previousVersion` structural metadata
  - is DC structural metadata
- restricts to much
  - we always found exceptions with interesting workflows
- structural metadata may also describe *changing features*
  - `nextVersion` changes but `previousVersion` in general not
- structural metadata can be really complex like DC

PIT-KIT should support the **FAIR principle**

- is relatively unspecified on the tools level
  - which partly explains it success, but
  - does it really help in this discussion?
Description by Syntactical Requirements

Suggestion:

- allow only PID-KITs of the following value type:
  - pid
  - boolean
  - integer
  - number
  - strings

- strings additionally can be restricted by
  - RegExp
  - Controlled Vocabulary
  - Length less than 10KB?

- ??? allow dictionaries and lists with max. 100 entries and 3 levels ???
Advantages:

- **simple** and **grounded** (C)
  - still **easy to prove** as criteria for a profiles
- driven by **process velocity** (A)
  - all allow very fast and easy decision processes
- allow to impose a huge variety of **semantics by naming** (D)
  - numbers can be temperature in one context and height in another
  - allows community specific semantics definition
  - are all use cases covered this way?
(un)changing features (B)

- unchanging features are not a syntactical but a semantical requirement
  - changing PID-KIT **needs special authorization**
  - they have from the users point of view a **WORM** (Write Once, Read Multiple) property
  - this gives a simple practical restriction on the scope of types

- Consequence: this can be an additional requirement
My Suggestion:
How should a PID-KIT Profile look like

- A PID-KIT Profile contains a **small set** (10-15?) of types
  - with values of the **syntactical requirement above**
  - and with an open naming and therefore **open semantics**
  - can contain **mandatory** and **optional** elements
  - can be explicitly **determined for a community**
  - and the **values cannot be changed** after minting the PID

- there is **not one profile** or a few them
  - but there is a huge number of possible profiles, that fulfil all the
    - requirements A-C and
    - the **FAIR principles** on the semantics at the community level

- the **implementation** of the profile should be given on the prefix level
Many Thanks

Questions ???

Contact@ePIC:
  - support@pidconsortium.eu

Contact@GWDG:
  - Ulrich Schwardmann
    T: 0551 201-1542, E: ulrich.schwardmann@gwdg.de