ePIC Project Updates and Discussion

Digital Objects - from RDA Results towards Implementation

Ulrich Schwardmann (GWDG)

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen
(GWDG)

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

22 October 2019, Helsinki

ePIC
Persistent Identifiers for eResearch

Introduction

content before:

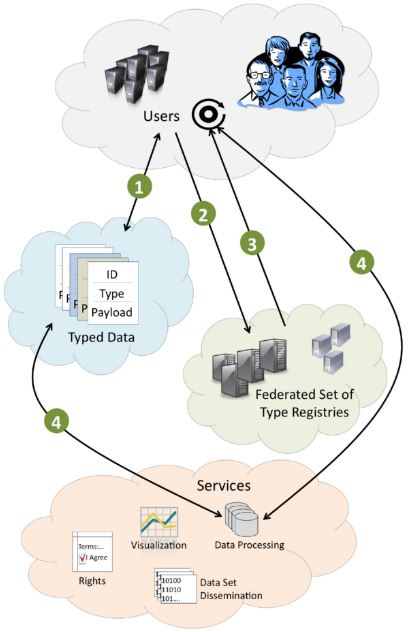
- Findability and PID
- Reusability and Metadata
- Interoperability and Registration of Types
    - Data Type Registries
    - Profiles and Policies
- Accessibility & the DO Cloud
    - Techniques
    - Collections
    - DO Browser
    - Concepts
- Searchability
    - Linked Data
- Questions

Introduction

content before:

- Findability and PID
- Reusability and Metadata
- **Interoperability and Registration of Types**
  - **Data Type Registries**
  - **Profiles and Policies**
- Accessibility & the DO Cloud
  - Techniques
  - Collections
  - DO Browser
  - Concepts
- **Searchability**
  - **Linked Data**
- Questions

Interoperability by Registration of Types



RDA working group on
Data Type Registries

- approach to provide *type definitions*
- a PID for each definition
- defines the type structure, its use and semantics
- CORDRA as DTR service
- typical use cases:
  - with given PID find a type and ask for its use at DTR (see left)
  - ask at DTR for types with given semantics and find via PIDs according data

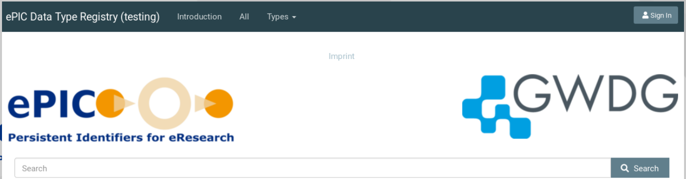
The ePIC Data Type Registry

- Features
  - Definition of PID Information Types
  - hierarchical types and automated schema extraction
  - Access via REST API, Browser
- based on CORDRA software
- GWDG is provider on behalf of ePIC
- Who can use the service?
  - public, authorization needed only for type definition

Overview: `http://dtr.pidconsortium.eu/`

PID InfoType states are:

- *in preparation* (21.T11148),
  - `http://dtr-test.pidconsortium.eu/`
- *candidate, approved, deprecated* (21.11104)
  - `http://dtr-pit.pidconsortium.eu/`


ePIC Data Type Registry (testing)
Introduction
All
Types
Sign In
Imprint
ePIC
Persistent Identifiers for eResearch
GWDG
Search
Search

Hierarchical Type Definitions at ePIC DTR

- types are often dependent from each other, how exactly?
- to exactly describe JSON objects by data types one needs:
    - a distinction between derived objects and basic objects
        - concept of *basic PID info types* and *PID info types*
    - a more exact description of the type dependencies
    - additionally a JSON schema inspired dependency model
- in consequence:
    - possibility to derive JSON schemas for the type values
        - automated server side schema derivation at ePIC DTR
    - one type defines in an exact way its whole dependencies
        - in objects of a certain type one can use the names of its parts (instead of type identifiers)
- see also Schwardmann, U.: Automated schema extraction for PID information types
    - PID: http://hdl.handle.net/21.11101/0000-0002-A987-7

RDA KernelInformationType Profile

- Output of the RDA KernelInformationType WG is a recommendation: *RDA_Recommendation_on_PID_Kernel_Information_0.pdf*
- specifying guiding principles (ch2)
- and a Draft Kernel Information profile (ch3)
  - consisting of special mandatory and recommended fields (types) that must/should be included as types of a PID
  - this Kernel Information profile is a first draft
  - it can be extended/modified wrt. the guiding principles
- these modifications need to be specified for single or groups of PIDs (profile policy)

RDA KernelInformationType Profile and Policies

Abstract Consequences:

- there exist different levels of abstraction for KIT **Profiles**:
  - schema level for describing different profiles
  - instance level of profiles
  - instance level of PIDs fulfilling profiles
- and there also levels of abstraction for KIT **Policies**
  - instance level of PIDs (or groups of PIDs):
    there needs to be a pointer to a profile (which is also given in the profile)
  - and there are other policies possible, regarding life cycles of DOs etc.

RDA KernelInformationType Profile at ePIC

ePIC Specific Consequences on the Typing level:

- a Kernel Information profile is expressible as an InfoType in the ePIC DTR
  - because it describes a kind of a schema assembling other InfoTypes and BasicInfoTypes.
  - However the derived schema is different, because we need to describe a Handle record at the end.
  - Therefore we need a different **DTR schema** here (**KernelInformationProfile**: `21.T11148/532ce6796e2828dd2be6`)
- A Kernel Information Profile is then an instance of the DTR schema KernelInformationProfile
  - An example instance of such a **DTR type** is **recommendedKernelInformationProfile**: `21.T11148/0c5636e4d82b88f86132`
- A PID, fulfilling a concrete Kernel Information profile, has to have all properties, as described in this profile.

RDA KernelInformationType Profile at ePIC

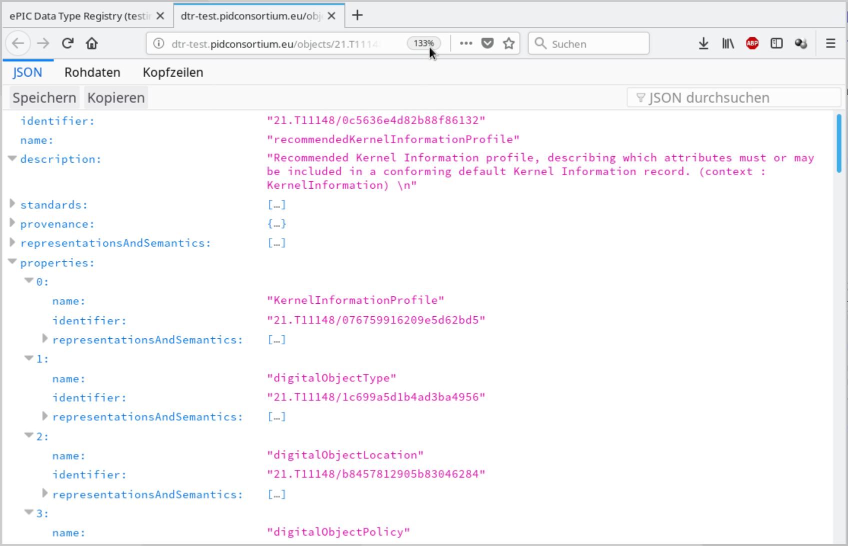
ePIC Specific Consequences on the Profile Verification level:

- from an ePIC Information type a JSON schema can be retrieved automatically
- a KIT Profile makes a statement about PID records
- PID records cannot be constrained by JSON schema to the needed level of detail as required by KernelInfoType
- consequence: the PID record needs to be provided in a different representation
- What to do:
  - use the existing JSON schema from ePIC DTR
  - provide **homomorph** PID records that can be constrained by this schema
- such a process is implemented for an instance PID of the recommended KIT profile

RDA KernelInformationType Policy at ePIC

- a policy describes general underlying rules for the use and the content of PIDs
  - some of such rules are digitally describable
  - some are also specified in the digitalObjectPolicy field of the RDA KIT recommendation
- created an instance **KernelInformationPolicy**: `21.T11148/f9aa655f3c6cb14bd7b0`
  - objectLifeCycleType (M), objectTombstoneInformation (O), objectLicense (O)
  - objectLifeCycleType: currently Unicode-String, could be a controlled vocabulary: static, dynamic_irregular, dynamic_regular, ...

ePIC KernelInformationProfile


ePIC Data Type Registry (testi
dtr-test.pidconsortium.eu/obj
dtr-test.pidconsortium.eu/objects/21.T1114
133%
Suchen
JSON
Rohdaten
Kopfzeilen
Speichern
Kopieren
JSON durchsuchen

```
identifier:                     "21.T11148/0c5636e4d82b88f86132"
name:                           "recommendedKernelInformationProfile"
description:                    "Recommended Kernel Information profile, describing which attributes must or may
                                be included in a conforming default Kernel Information record. (context :
                                KernelInformation) \n"
standards:                      […]
provenance:                     {…}
representationsAndSemantics:    […]
properties:
  0:
      name:                     "KernelInformationProfile"
      identifier:               "21.T11148/076759916209e5d62bd5"
      representationsAndSemantics: […]
  1:
      name:                     "digitalObjectType"
      identifier:               "21.T11148/1c699a5d1b4ad3ba4956"
      representationsAndSemantics: […]
  2:
      name:                     "digitalObjectLocation"
      identifier:               "21.T11148/b8457812905b83046284"
      representationsAndSemantics: […]
  3:
      name:                     "digitalObjectPolicy"
```

How could a Policy look like

- Examples
  - suffix generator (counter, hash)
  - deletion allowed/forbidden
  - use of profile for information types
  - inheritage of profile elements from prefix to suffix
  - inheritage of policy elements from prefix to suffix
- all those can be described by boolean values or controlled vocabulary

Searchability ???

Hasn't Google solved the searchability question?

- Searchability actually is a kind of *reverse lookup*
  - findability was answered by: get the data for the reference
  - searchability means: get the reference for some criteria
- this raises a lot of questions
  - technical implementation
    - centralized vs. distributed
    - scalability
    - access control
    - data base
    - query languages
  - legal, social
    - privacy
    - GDPR
  - governance and trust
  - ...

Search in Handle Mirrors

- the Handle system allows to setup mirror servers for higher resolution availability and as Handle database backup in hidden mode
- these hidden mirror services can be exploited in different ways
  - for complex SQL queries without load on the productive resolution system
  - for other kind of searches, if the database backend maps to different systems
  - GWDG developed backends[1] for
    - elasticSearch with facetted search on distinguished types (service: `https://pid.gwdg.de/search/`)
    - Neo4J as graph database on types (service: `http://141.5.105.252:7474/browser/`)
    - Documentation can be provided on request

[1] Triet Ho Anh Doan, A Graph Database for Persistent Identifiers, https://doi.org/10.25625/0LJ60A, Masterthesis, 2019

Types vs. Linked Data

- An Example of a type: `isNextVersionOf`

This gives a triple:

- *pid-do1 type pid-do2*
- Digital-Object-1 `isNextVersionOf` Digital-Object-2

Thus one has a relation:

subject predicate object

with types as predicates.

- Types can be represented by PIDs again (DTR)

A feasability study at GWDG:

- mapping of type triples into a Neo4J graph database
- enables SPARQL queries
- realized as a Handle mirror with Neo4J database adapter

ePIC
Persistent Identifiers for eResearch

Types vs. Linked Data

Examples of a types for metrology:

```
@prefix ePICdtr: <http://dtr.pidconsortium.eu/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xml: <http://www.w3.org/XML/1998/namespace> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://dtr.pidconsortium.eu/21.T11148/0a0fa93c89ac30e19d74>
ePICdtr:identifier <hdl:21.T11148/0a0fa93c89ac30e19d74> ;
ePICdtr:name "qty:time";
ePICdtr:properties "{’dimensions’:’T’,’name’:’unit:s’,
’issuer’:’BIPM’},’symbols’:{’alphabet’:’Latn’,’symbol’:’t’},
... }]}
ePICdtr:type ePICdtr:PID-BasicInfoType-Metrology .

<http://dtr.pidconsortium.eu/21.T11148/2f9571fa836af29bce01>
ePICdtr:identifier <hdl:21.T11148/2f9571fa836af29bce01> ;
ePICdtr:name "cnst:constant_Planck"; ...
```

Types vs. Linked Data

- currently only prototypical level
- required by customers to justify the choice of types
- Hierarchical Type Defintions lead to recursion in operation
    - which can be exploited automatically
- algorithm: Python with RDF plugin
- level of granularity still has to be determined

Many Thanks

Questions ???

Contact at ePIC:

- support [at] pidconsortium.eu

Contact at GWDG:

- **Ulrich Schwardmann**
  T: 0551 201-1542, E: ulrich.schwardmann [at] gwdg.de