How to use the Data Type Registry?

RDA Datafabric - IG

Ulrich Schwardmann

Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)

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

27 August 2020, video conference

Content

1. FAIR Digital Objects
2. Type Registration
   Life Demo
3. Profiles
4. Questions

How to Foster automation in the Data Domain?

- The amount of data needs **Automation** and
- the **Heterogeneity and Complexity of Data** are the mayor obstacles for automation

There are two classical approaches to ease automation and overcome Heterogeneity and Complexity

- **Abstraction**
  - this is the FDO concept
- **Standardization**
  - this should be covered by the Data Type Registry
  - a leightweight, domain specific and federated approach for standardization

Reusability by Abstraction

- needs knowledge about basic properties of data
  - **Metadata** is often unavailable, not connected to data or not interpretable
- Registration:
  bind metadata and data with PID to a digital object
- For reuse provide as much of this knowledge before access to the data
- **PID Information Types**
  - are additional metadata, stored in the PID database
  - similar to Mime Types, but much more flexible
  - Examples are *checksum* , *mime type* , *reference information, versioning (relative and absolute), embargo time, expiration date, add. metadata location, basic Dublin Core, access restrictions and methods, data and table column formats, collection description, ...*

Data Type Registry
Ulrich Schwardmann
FAIR Digital Objects
Type Registration
Life Demo
Profiles
Questions

Reusability by Abstraction

- needs knowledge about basic properties of data
  - **Metadata** is often unavailable, not connected to data or not interpretable
- Registration:
  bind metadata and data with PID to a digital object
- For reuse provide as much of this knowledge before access to the data
- **PID Information Types**
  - are additional metadata, stored in the PID database
  - similar to Mime Types, but much more flexible
  - Examples are *checksum , mime type , reference information, versioning (relative and absolute), embargo time, expiration date, add. metadata location, basic Dublin Core, access restrictions and methods, data and table column formats, collection description, ...*

Data Type Registry
Ulrich Schwardmann
FAIR Digital Objects
Type Registration
Life Demo
Profiles
Questions

Interoperability by Standardization

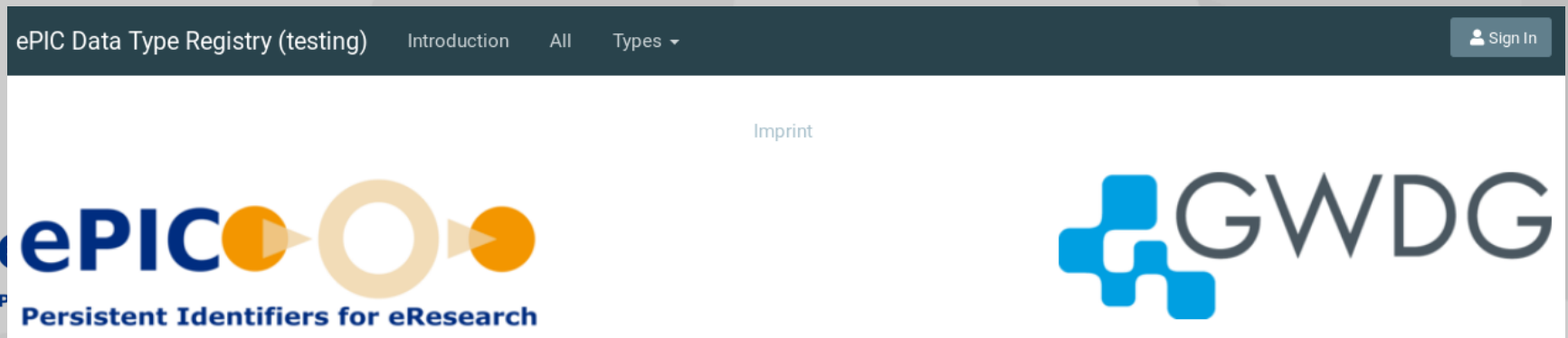
- needs standardization and policies first of all on the metadata (types), but also on references and operations
- usually standardization is made for technical interoperability, but a long and difficult process
- policies are often simpler, but technically intangible, because part of an agreement between humans
- one needs simple machine readable definitions, made for specific domains, but reusable by others
  - linked data goes into this direction, but the end points are in general not machine readable URLs
- here so called **Data Type Registries (DTRs)** come into play
  - to provide leight weight standardization for metadata
  - and attach a PID to each such definition for disambiguation

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/`

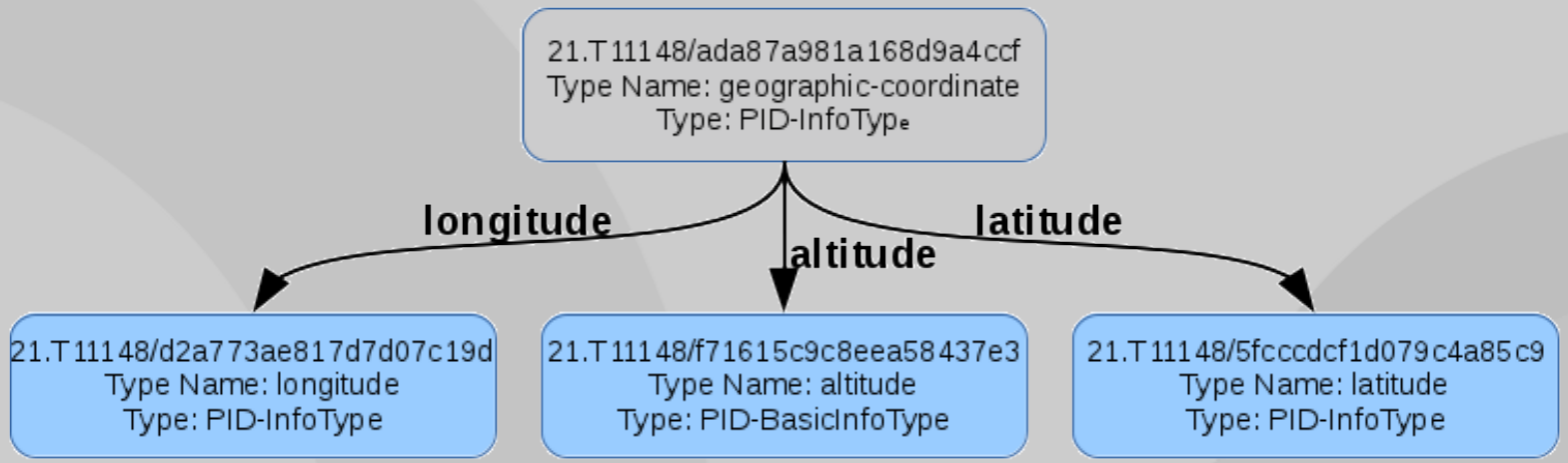
Policies for a PID InfoType life cycle:

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



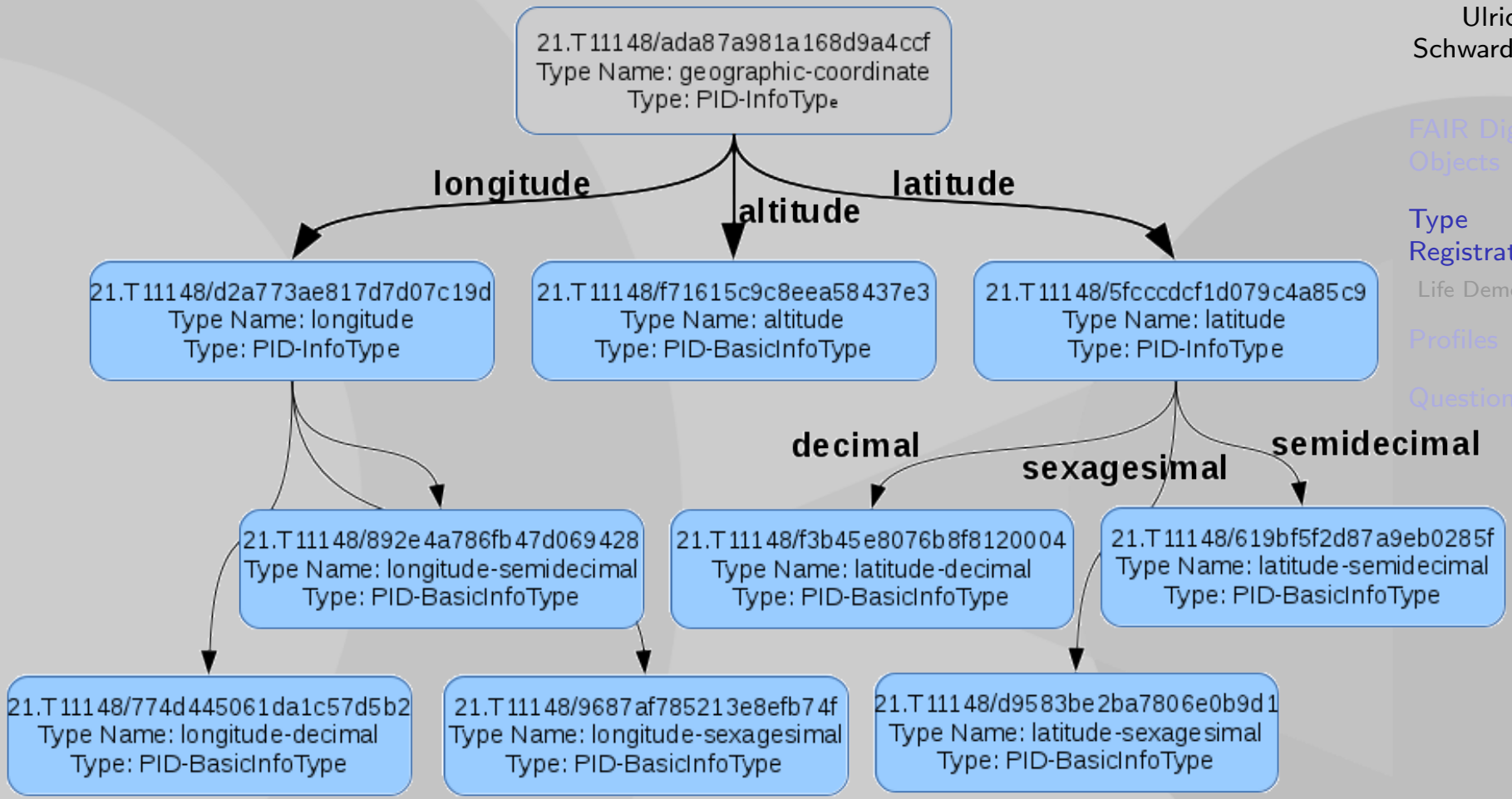
Hierarchies in Metadata

Example: geographic coordinate.



Hierarchies in Metadata

Example: geographic coordinate.



Hierarchical Type Definitions

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

Life Demo

Data Type Registry

Ulrich Schwardmann

FAIR Digital Objects

Type Registration

Life Demo

Profiles

Questions

Special Technical Registration Issues

- **Abbreviated Form**: the need for keys that are just numbers, strings ore arrays
  - since all types have a name usually the values of types would be objects with that name as parameter
  - *Abbreviated Form* equals True allows unnamed values
- **Omit Names of Subtypes**: avoid over-specified objects
  - all subtypes come with a name in their own definition and a name as subtype in the type they are used in
  - in some cases the values should specify both of these names in nested objects, in others this would be an over-specification
    - Example: many types reuse "unicode-string" for certain fields like name. but one would not like to have a name specification like { `"name" : { "unicode-string" : "Ulrich" } }`

Types vs. Linked Data

- An Example of a type: `isPreviousVersionOf`
  - Such a type is stored as key-value pair in the PID (*pid-do1*) of a digital object
  - as key-value pair consisting of the *type* and the PID of the previous version (*pid-do2*)

This gives a triple:

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

Thus one has a relation:

subject predicate object

with types as predicates.

- Types can be represented by PIDs again (DTR), so the SPO becomes a PID triple
- A converter of FAIR-DOs with registered types into JSON-LD representation exists.
- was shown in the contexts of PID for Instruments

Kernel Information Type Profile

	Property identifier	Content format	Cardinality	Explanation
1	PID	Handle	1..n	Global identifier for the object; external to the PID Kernel Information
2	KernelInformationProfile	Handle	1	Handle to the Kernel Information type profile; serves as pointer to profile in DTR. Address of DTR federation expected to be global (common) knowledge.
3	digitalObjectType	Handle	1	Handle points to type definition in DTR for this type of object. Distinguishing metadata from data objects is a client decision within a particular usage context, which may to some extent rely on the digitalObjectType value provided.
4	digitalObjectLocation	URL	1..n	Pointer to the content object location (pointer to the DO). This may be in addition to a pointer to a human-readable landing page for the object.
5	digitalObjectPolicy	Handle	1	Pointer to a policy object which specifies a model for managing changes to the object or its Kernel Information record, including particularly object access and modification policies. A caller should be able to determine the expected future changes to the object from the policy, which are based on managed processes the object owner maintains.
6	etag	Hex string	1	Checksum of object contents. Checksum format determined via attribute type referenced in a Kernel Information record.
7	dateModified	ISO 8601 Date	0..1	Last date/time of object modification. Mandatory if applicable.
8	dateCreated	ISO 8601 Date	1	Date/time of object creation
9	version	String	0..1	If tracked, a version for the object, which must follow a total order. Mandatory for all objects with at least one predecessor version.

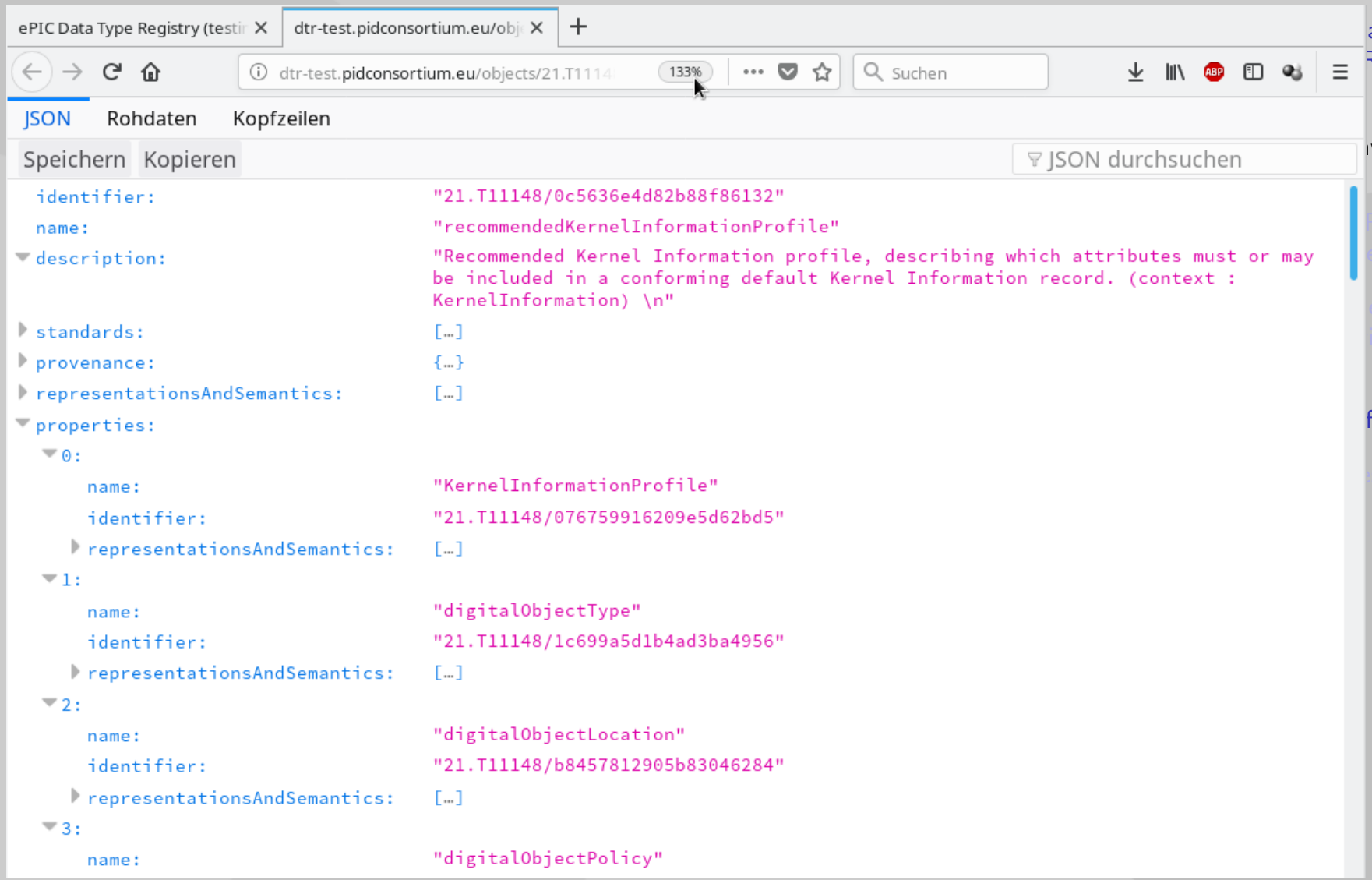
10	wasDerivedFrom	Handle	0..n	PROV-DM: an update o construction entity.
11	specializationOf	Handle	0..n	PROV-DM: shares all as presents mo the latter.
12	wasRevisionOf	Handle	0..n	PROV-DM: entity is a re
13	hadPrimarySource	Handle	0..n	PROV-DM: something p experience time of the t hindsight.
14	wasQuotedFrom	Handle	0..n	PROV-DM: an entity, su may or may
15	alternateOf	Handle	0..n	PROV-DM: thing. These and the alte time.

RDA KernelInformationType Profile at ePIC

ePIC Specific Consequences on the Typing:

- In order to express a Kernel Information Profile as an InfoType in the ePIC DTR we need a different **DTR schema**
  - **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 the properties, as described in this profile.

ePIC KernelInformationProfile



ata Type
Registry

Ulrich
wardmann

R Digital
ects

e
istration

Demo

files

estions

ePIC
Persistent Identifiers for eResearch

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

Further reading: Schwardmann, Ulrich, 2020, "The ePIC PID Information Type Registry", DOI: 10.25625/9DNRSJ

ePIC
Persistent Identifiers for eResearch