# Building and Maintaining a Registry for PID Info Types

## A DTR of the ePIC Persistent Identifier Consortium

Ulrich Schwardmann, GWDG

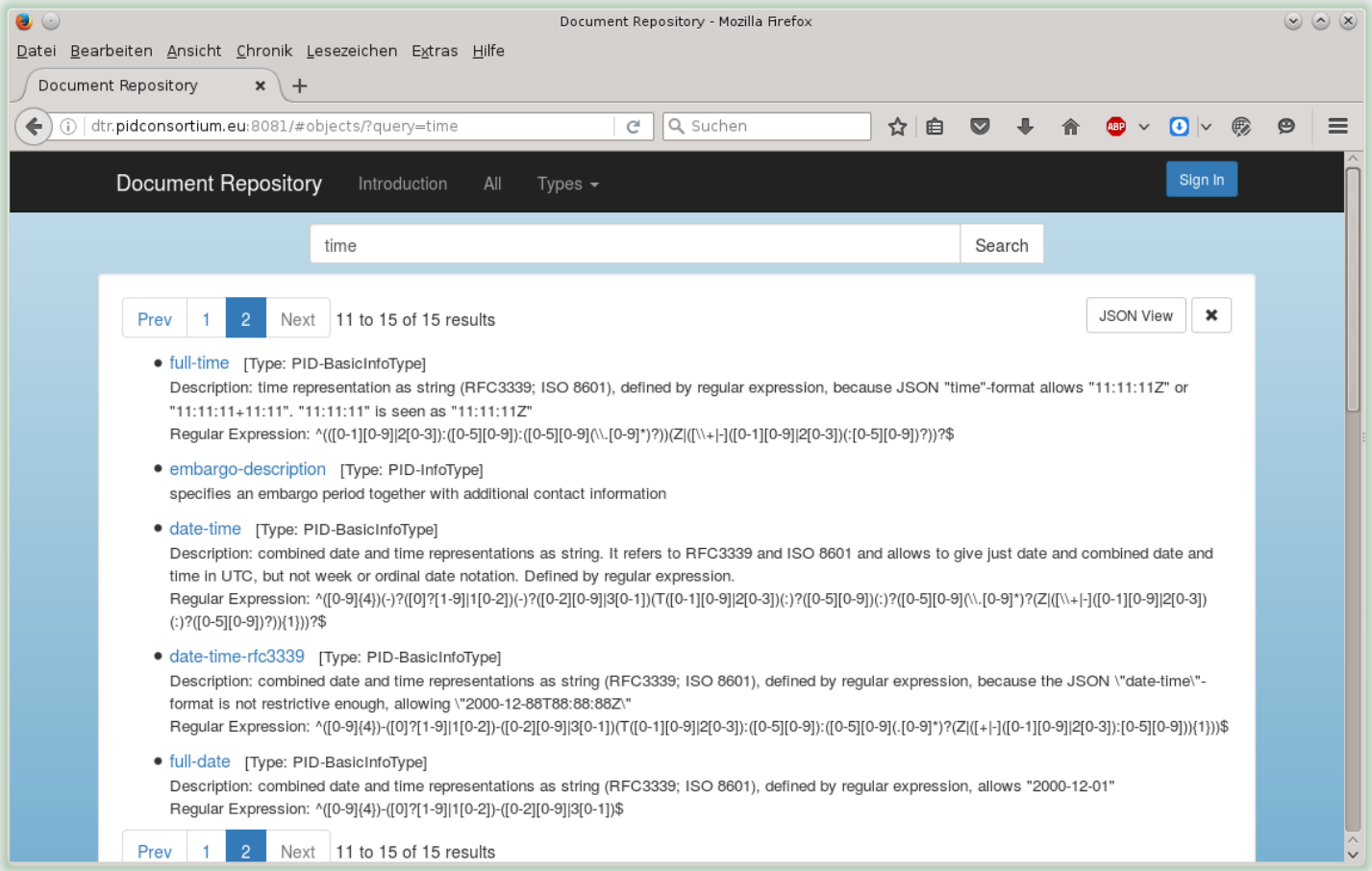
Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG)

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

17 September 2016, Denver

# ePIC PIT-DTR registration GUI

see: `http://dtr.pidconsortium.eu:8081`
currently 75 PID-BasicInfoTypes and 57 PID-InfoTypes defined





ePIC DTR4PIT

Ulrich Schwardmann, GWDG



DTRs and PID Info Types

Type Hierarchies

Schemas

Type Validation

Discussion

# Type conformity and validation

- InfoType information has to be particularly reliable, because
  - the functionality of the data services is dependent on a correct preprocessing.
  - this kind of metadata is interpreted by automated services
  - therefore it is necessary to avoid each precondition of human interpretation.
- **Schemas** need to be part of the information type description
  - and have to be defined in a clearly determined and reproducible way.
  - only an automatic process can guarantee this.

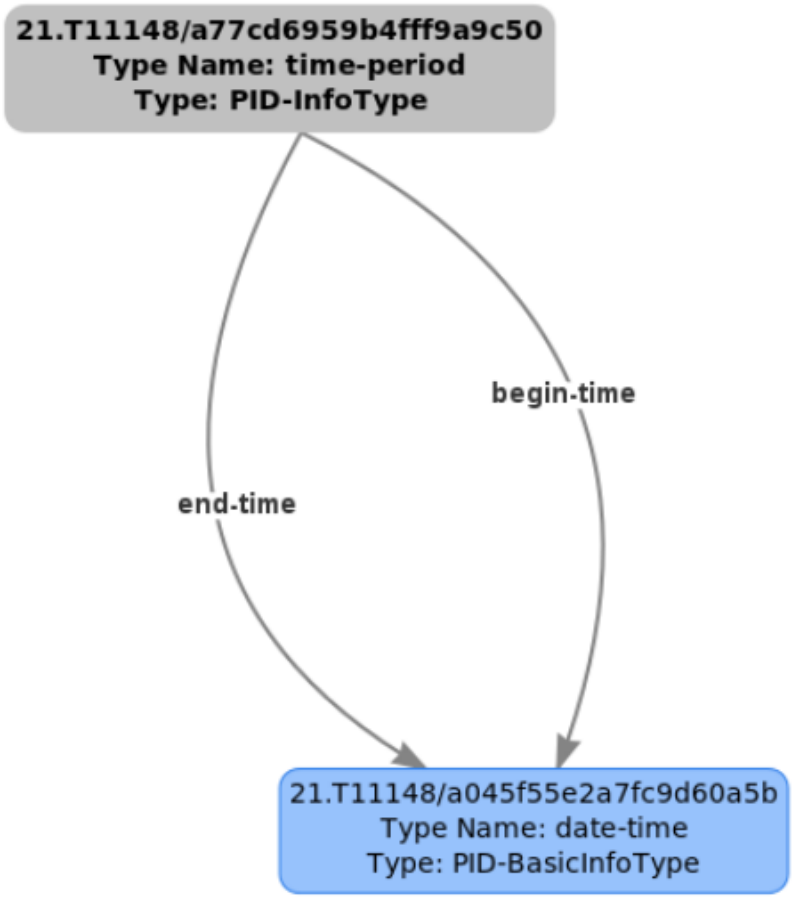
# Hierarchies of Types

- Information types are often refering to simpler types:
  - a geolocation contains longitude, given in sexagesimal or decimal form.
  - citation information contains an author, perhaps given by an ID in a certain ID system
- they are eventually based on very basic types
  - determined by regular expressions or other restrictions

**Suggestion:**

- Information types are recursively built out of a finite combination of
  - information types and
  - such basic information types
- possible advantage: reuse of schemas

# Hierarchies of Types



# Schemas, how to generate and maintain them?

- common practice: manually build a schema according to a given description
  - a lot of manual work for schema derivation and adaption
  - inconsistencies between description and the schema

automatic schema derivation (in JSON):

- exactly describe the information type dependencies in the type description in the DTR
  - enable as much flexibility in the JSON framework as possible
  - the description of dependencies is a derivation from the canonical type description set
- exploit the hierarchy in an automated process
- basic information types have a (simple) schema as leaves in the dependency graph
- store the schema in the type description of the DTR

# Hierarchies of Types

# Validation of instances



- all necessary information for type validation can be found in the PID and the DTR
  - and can be retrieved via the REST API of the Handle System or the DTR
- client function calls like `typeIsValid(PID,typeID)` are provided

# Validation of DataCite Mandatory Properties

- all mandatory metadata entries for the DataCite MDS are also defined in JSON in the DTR
- types for all metadata entries for the DataCite MDS will follow
- additionally there is a collective type of these mandatory properties defined in the DTR
- for both metadata instances in JSON and XML exist schema
- there exists a crosswalk from the ePIC JSON to the DataCite XML
  - validation of the JSON type instance guarantees a valid DataCite XML

Conclusion: Types are a great medium to enable more interoperability between PID systems

# Validation of tabular data with the Frictionless Data Schema

ePIC
DTR4PIT

Ulrich

- Frictionless Data has a hierarchical schema description for necessary data to validate and process tabular data

Document Repository | Introduction | All | Types | Sign In

FLD | Search

JSON View

- primaryKey-FLD [Type: PID-InfoType]
  A primary key is a field or set of fields that uniquely identifies each row in the table given by a CSV file.
  Default Value: true
- reference-FLD [Type: PID-InfoType]
  a reference where entries in a given field (or fields) on this table ('resource' in data package terminology) is a reference to an entry in a field (or fields) on a separate resource.
  Default Value: true
- foreignKeys-FLD [Type: PID-InfoType]
  A foreign key is a reference where entries in a given field (or fields) on this table ('resource' in data package terminology) is a reference to an entry in a field (or fields) on a separate resource.
  Default Value: true
- fields-FLD [Type: PID-InfoType]
  fields is an ordered list of field descriptors one for each field (column) in the table
  Default Value: true
- CSV-table-description-FLD [Type: PID-InfoType]
  A CSV table description intends to describe the structure of tables consisting of a set of rows. Each row has a set of fields (columns). We usually expect that each row has the same set of fields and thus we can talk about the fields for the table as a whole. It decribes the delimiter in the CSV file between columns as well as the constrains of the columns and primarry keys and similar.
  Default Value: true
- field-FLD [Type: PID-InfoType]
  A field descriptor of Frictionless Data is a simple JSON hash that describes a single field. The descriptor provides additional human-readable documentation for a field, as well as additional information that may be used to validate the field or create a user interface for data entry.
- csvddf-FLD [Type: PID-InfoType]
  CSV Dialect Description Format defines a simple JSON format to describe the various dialects of CSV files; it aims to deal with a reasonably large subset of the features which differ between dialects (terminator strings, quoting rules, escape rules, etc), and roughly to describe the union of the capabilities of Python's csv module, Ruby's CSV module, and the MySQL and Postgres bulk load facilities at the time of writing (February 2013). The keys of the "dialect" member of the JSON dictionary are intended to be very close to the

# Validation of tabular data with the Frictionless Data Schema

- The FDL types together with corresponding resulting schemas are described in DTR


```
  "properties" : {
    "CSV-table-description-FLD" : {
      "additionalProperties" : false,
      "description" : "CSV-table-description-FLD@21.T11148_66ee7993765837104ce3",
      "properties" : {
        "CSVDDF" : {
          "$ref" : "#/definitions/21.T11148_bbbe2b8ca636e95eaffc"
        },
        "fields" : {
          "$ref" : "#/definitions/21.T11148_39c5c9e5b9a54b398562"
        },
        "foreignKeys" : {
          "$ref" : "#/definitions/21.T11148_071c82c85172bed4a8a5"
        },
        "primaryKey" : {
          "$ref" : "#/definitions/21.T11148_17ce40f9612c494cc3c3"
        }
      },
      "required" : [
        "fields",
        "CSVDDF"
      ],
      "type" : "object"
    }
  },
  "required" : [
    "CSV-table-description-FLD"
  ],
  "type" : "object"
}
```

# Validation of tabular data

ePIC
DTR4PIT

Ulrich Schwardmann, GWDG

DTRs and PID Info Types

Type Hierarchies

Schemas

Type Validation

Discussion

Example: Validate the CSV file at
`http://techslides.com/demos/country-capitals.csv`

- columns are country code, city name, latitude, longitude, region-code, continent-names
- for these columns exist (candidate) InfoTypes
- Validation by a simple python script
  - gets an array of the PITs of the column types
  - retrieves the CSV file above
  - gets the names and the schemas for the column types from the DTR
    - using the client library from above
  - validates the column entries with the schemas
  - and outputs each row with the offended column entry

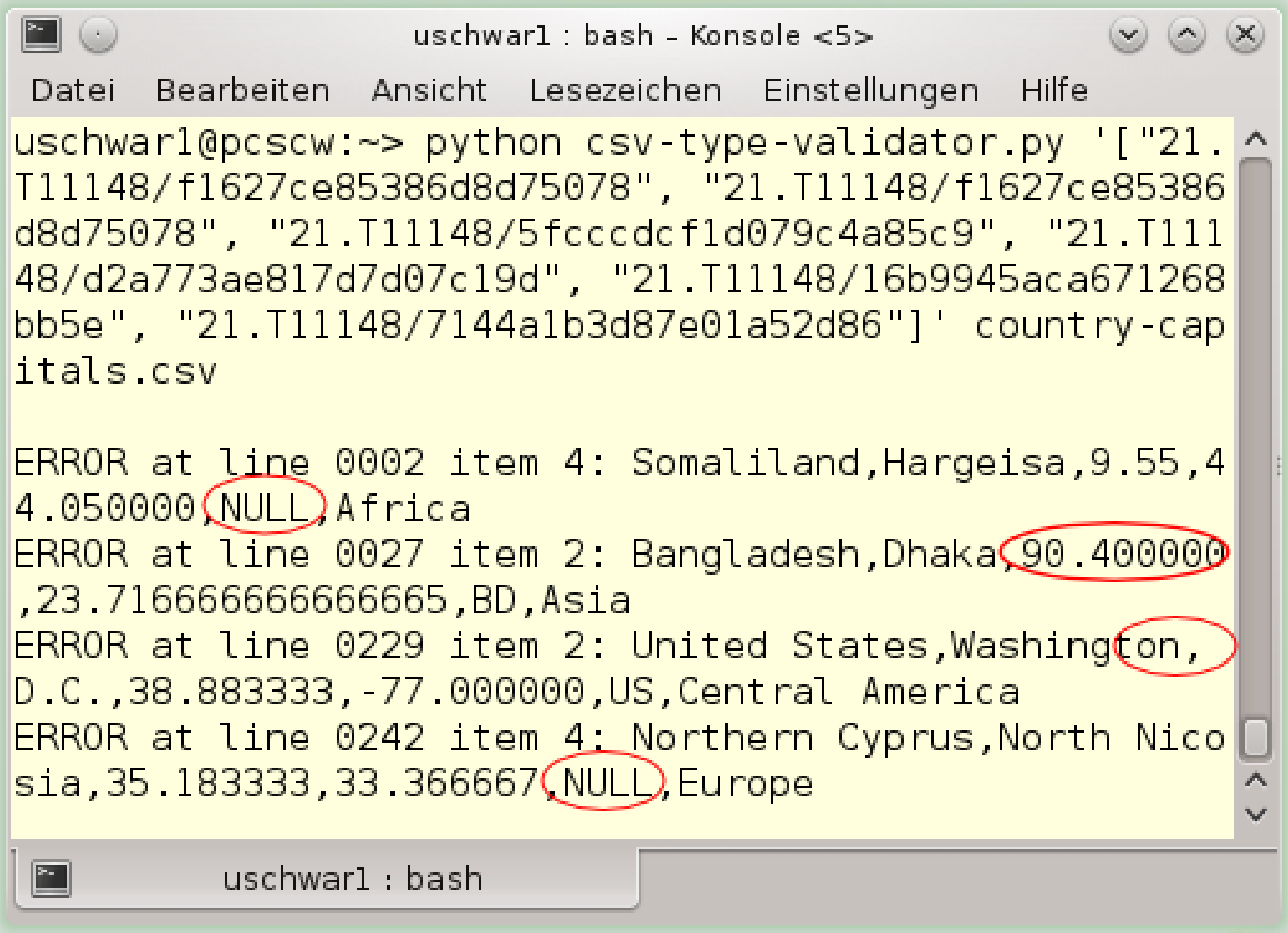
# Validation of tabular data



Example: Validate the CSV file at
`http://techslides.com/demos/country-capitals.csv`

- columns are country code, city name, latitude, longitude, region-code, continent-names
- for these columns exist (candidate) InfoTypes
- Validation by a simple python script
  - gets an array of the PITs of the column types
  - retrieves the CSV file above
  - gets the names and the schemas for the column types from the DTR
    - using the client library from above
  - validates the column entries with the schemas
  - and outputs each row with the offended column entry

# Validation of tabular data



# Endorsement and Deprecation of Types

- registration of PID InfoTypes is an endorsement process
  - currently all registered PIT are **in preparation**
  - after their description is sufficiently settled they become **candidates**
  - after there is no change request anymore they become **approved** and will be frozen
  - new version is necessary for the changement of appoved types
    - in this case the former type becomes **deprecated**
- the endorsement of PID InfoTypes needs a **reviewing board**

# **Questions** and **Discussion**



**Thanks** for your attention !

View also the ePIC Webpage:

`http://pidconsortium.eu`