18 months, RDA P1 to P4
co-chairs: Tim DiLauro (JHU); Tobias Weigel (DKRZ)

Goal: Harmonization of basic information types associated with PIDs across disciplines and infrastructures

Approach: Design an API and type examples to target practical usage

Strong interaction with Data Type Registries WG
Example scenario

Data?
Metadata?
Types for information directly associated with PIDs

- IDENTIFIER
  - properties
    - size
    - checksum
    - timestamps
    - format
    - aggregation
    - version
    - version
    - flags
    - ...

Resolver

Storage

black box
A Persistent Identifier is a long-lasting ID represented by a string that uniquely points to a DO and that is intended to be persistently resolvable to access meaningful, current state information about the identified DO. (from DFT wiki)
Something needs to reach across PID systems

The PID Information Types API serves two purposes: Facilitating **typing** and enabling **interoperability** across PID Systems.

**Higher level services**

**PID Info Types API**

**PID system**  **PID system**  **PID system**
Typing and Type Registries

- Two usage scenarios for TypeReg:
  - Typing of data entities
  - Typing of PID record value fields
  - Reference to data type in properties record
Use cases: Applications for typed PID records

- Replication management
- Version management
- Provenance tracing
- Access control
- Composition
- …
Conformance queries reach beyond single properties

- A PID record may contain various properties, which however make up useful groups
  - Fixity (Format, Checksum, Size, ...)
  - Accessibility (Format, License, Owner, ...)
- A particular service may require a distinct set to be present
## Example type list from the final report – to be continued...?

<table>
<thead>
<tr>
<th>Name</th>
<th>Range</th>
<th>Identifier</th>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type: Citation Information (EXAMPLE namespace)</strong></td>
<td>11314.2/d5396a97c316a0eaca055846ba4233ac</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Title</strong></td>
<td>STRING</td>
<td>11314.2/07841c3f84cbe0d4ff8687d0028c2622</td>
<td></td>
</tr>
<tr>
<td><strong>Creator</strong></td>
<td>STRING</td>
<td>11314.2/31810b2c24913929bb5e0d4d949de97</td>
<td></td>
</tr>
<tr>
<td><strong>Publication date</strong></td>
<td>DATE</td>
<td>11314.2/daed5901fbb2570ee95c4009c739de2</td>
<td></td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>STRING</td>
<td>11314.2/56211d62153b3500ce3b16cf86d6b403</td>
<td>optional</td>
</tr>
<tr>
<td><strong>License</strong></td>
<td>STRING</td>
<td>11314.2/2f305c8320611911a9926bb53f6ad8c9</td>
<td>optional</td>
</tr>
<tr>
<td><strong>Type: System level access information (EXAMPLE namespace)</strong></td>
<td>11314.2/09d35f22e48b60284029ba51c17e2944</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Creation date</strong></td>
<td>DATE</td>
<td>11314.2/6b3e1230d1b68965e290b16a43d2f46d</td>
<td></td>
</tr>
<tr>
<td><strong>Deletion date</strong></td>
<td>DATE</td>
<td>11314.2/7e78be9736ad7f6bb5fb31218821eba5</td>
<td>optional</td>
</tr>
<tr>
<td><strong>Permissions</strong></td>
<td>STRING</td>
<td>11314.2/d057258f7b406fd9aad5a3893aba8208</td>
<td>optional</td>
</tr>
<tr>
<td></td>
<td>STRING</td>
<td>11314.2/56bb4d16b75ae50015b3ed34bb519f</td>
<td></td>
</tr>
<tr>
<td><strong>Object size (in bytes)</strong></td>
<td>STRING</td>
<td>11314.2/0006e28e2f6e1ecce836e593bed38ae</td>
<td></td>
</tr>
<tr>
<td><strong>Type: Aggregation information (EXAMPLE namespace)</strong></td>
<td>11314.2/699d487eff50c2e10982f4b85ed053a9</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Parent object identifier</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/699d487eff50c2e10982f4b85ed053a9</td>
<td>optional</td>
</tr>
<tr>
<td><strong>Child object identifier</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/699d487eff50c2e10982f4b85ed053a9</td>
<td>optional</td>
</tr>
<tr>
<td><strong>Type: Versioning information (EXAMPLE namespace)</strong></td>
<td>11314.2/6b507d787d06e4eb8f23b5b56a88bb</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Predecessor identifier</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/6b507d787d06e4eb8f23b5b56a88bb</td>
<td>optional</td>
</tr>
<tr>
<td><strong>Successor identifier</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/6b507d787d06e4eb8f23b5b56a88bb</td>
<td>optional</td>
</tr>
<tr>
<td><strong>Type: Preliminary example for EUDAT core information (EUDAT namespace)</strong></td>
<td>11314.2/5f45666fc8689e3565728ca51c1b5e7</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Checksum</strong></td>
<td>STRING</td>
<td>see above</td>
<td></td>
</tr>
<tr>
<td><strong>Format</strong></td>
<td>STRING</td>
<td>11314.2/1a4f53a28b72d4bf4f8fdda7a2089595</td>
<td></td>
</tr>
<tr>
<td><strong>Data identifier</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/24dd85c4a3d39fb0d7e83a510a5041c6</td>
<td></td>
</tr>
<tr>
<td><strong>Metadata identifier</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/58a44100d2bcd1a34f887eb87bc6f701</td>
<td></td>
</tr>
<tr>
<td><strong>Repository of Record</strong></td>
<td>IDENTIFIER</td>
<td>11314.2/5546b0166091d9ae869f081f5548f3fc</td>
<td></td>
</tr>
<tr>
<td><strong>Mutability flag</strong></td>
<td>BOOLEAN</td>
<td>11314.2/7c81e954aead6a2f772aad83986d3e9</td>
<td></td>
</tr>
<tr>
<td><strong>Landing page address</strong></td>
<td>URL</td>
<td>11314.2/66af2639d388977e81b85f6413df1e2c</td>
<td></td>
</tr>
<tr>
<td><strong>Date of deposition</strong></td>
<td>DATE</td>
<td>11314.2/35837218ff18dcc56a2d32e0fb30fa7fb</td>
<td></td>
</tr>
</tbody>
</table>
The API focuses on reading and making sense of typed PID record information.
There are interfaces via Java and HTTP.

GET /peek/{identifier}
GET /property/{identifier}
GET /type/{identifier}

GET /pid/{identifier}?...
- Conformance information included if Types are given
Thanks to Tom Zastrow, there is also a small demonstrator running at RZG.
How to get the PIT output


- Prototype source code available via git: git clone git://redmine.dkrz.de/rdapit.git

- Final overview report available from the RDA websites: https://rd-alliance.org/groups/pid-information-types-wg.html

- More formal outcome package in the loop.
- Licenses: CC0 / simple BSD
Even with very simple information, each use case favors a different set of types
There is no single set of types fitting all cases – we have to live with that in practice and look towards the Type Registries to help us
Community processes must define types from practical adoption
How does the PID Information Types effort continue?

- The API is a prototype that has to see further refinement and further practical adoption.
- DKRZ follows through with future plans in the context of an international data infrastructure (ESGF) and EUDAT.
  - This will also shed more light on essential types.
- Interest was also stated by e.g. Deep Carbon Observatory and the Materials Genome Initiative.
Take-home messages

- Work is not over – now comes the clash with practice
- Assigning PIDs is the first step. Typing is the second.
- Political consensus in a community/infrastructure is crucial – challenge too big for single institution
- Keep It Simple & Stupid – also in the future
- Local motivation – automate our workflows at DKRZ

- Continuing efforts in RDA regarding Collections
Thank you for your attention.