RESEARCH DATA ALLIANCE

Profiles, policies and processes

Tobias Weigel, Beth Plale

RDA PID Training, Garching, 2016/08/31

research data sharing without barriers

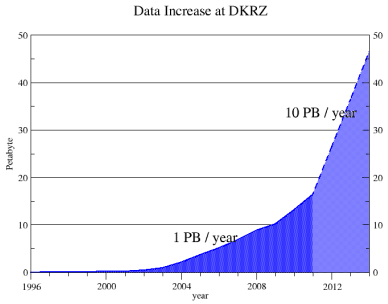
rd-alliance.org

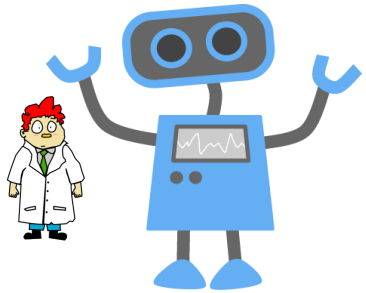
How this session will unfold...

1. Introduction to profiles, policies and processes
2. Split into groups – group exercise
3. Discussion on exercise

The greater context: Challenges in managing research data

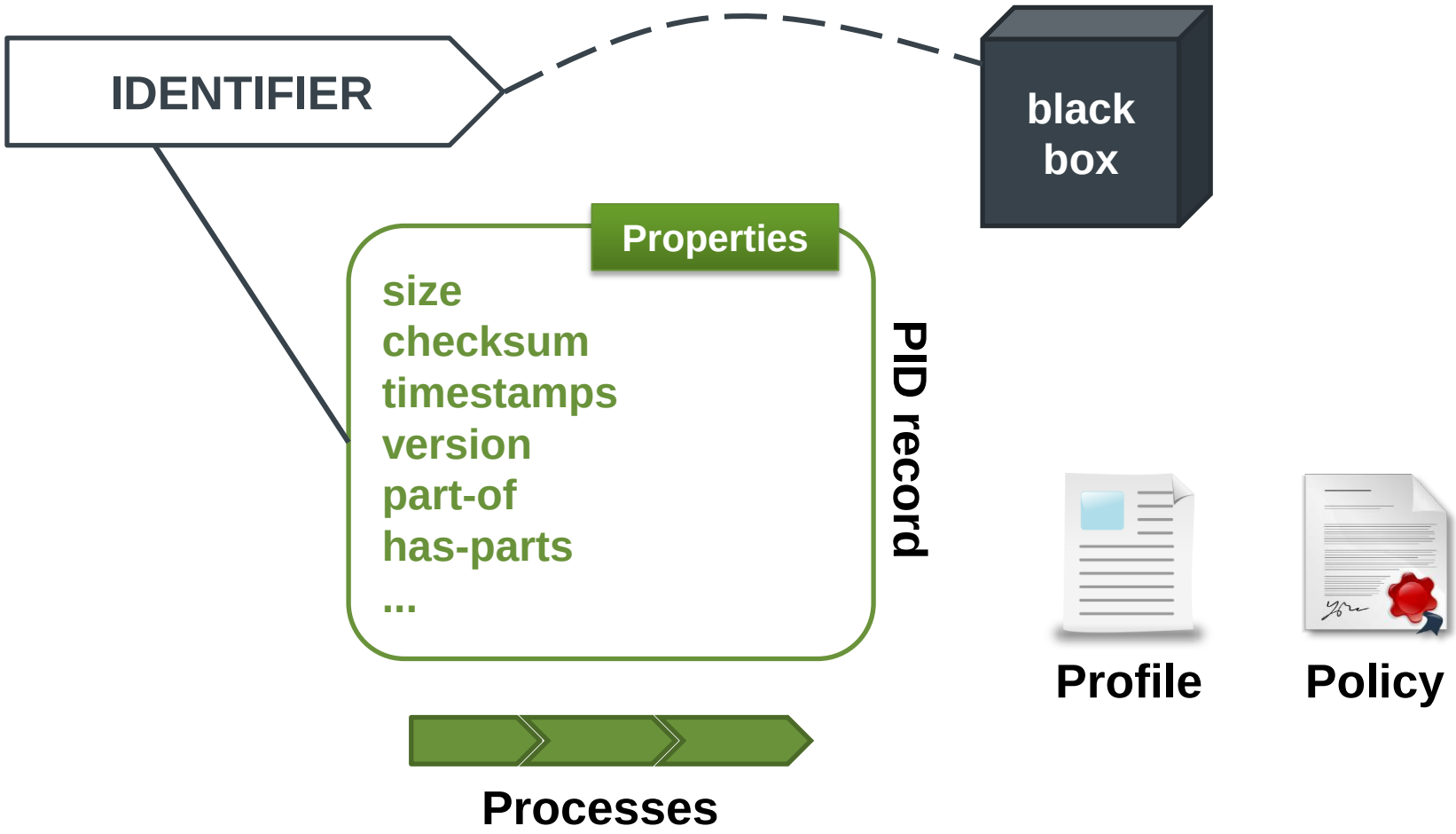
- Exponential growth in object numbers, management resources remain constant
  - Automation, machine interaction vs. human interaction





- Digital objects have a long life – technology changes
  - Goal: Parsimonious PID record
  - Technical development focus on interfaces and protocols

Quick introduction – conceptual model



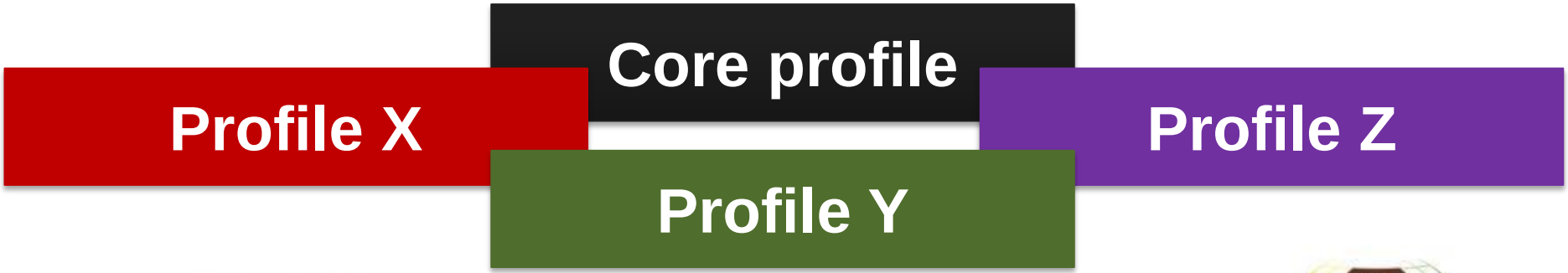
What is a profile and how does it relate to PID records?

Climate sciences	Material sciences	Linguistics
size	size	size
checksum	checksum	checksum
version	timestamps	granularity
part-of	version	part-of
has-parts	predecessor	has-parts
replica locations	successor	version
...	...	...

Core profile

What is a profile and how does it relate to PID records?

Climate sciences	Material sciences	Linguistics
size	size	size
checksum	checksum	checksum
version	timestamps	granularity
part-of	version	part-of
has-parts	predecessor	has-parts
replica locations	successor	version
...	...	...



What is a profile and how does it relate to PID records?

- Base assumption: There is a minimal core set of information associated with each PID.
- That minimal set should be useful not only to the maintainer of a PID or object, but also to third parties.
- Each user community may design their own profiles.
  - No single size fits all – but recurring elements should be reused

size
checksum
timestamps
version
part-of
has-parts
...

How do profiles work – technically in the Data Type Registry

- PID records contain key-value pairs.
- Each PID record can be related to one or several profiles.
- There are two models for binding records and profiles:
  - Explicitly: Dedicated record attribute
  - Implicitly: Conformance to a given profile
- **Ultimately, profiles should be registered in a Data Type Registry**

Can a PID record contain elements beyond what is specified in a profile?

Example profile: Artificial profile, also for the exercise

Property name	Target type	Mandatory?
Location	URL	Yes
Policy	Policy specification*	Yes
Time stamp (last modified)	Date/Time	Yes
Predecessor version	PID	No
Successor version	PID	No
Deletion flag	Boolean	No
Deletion reason	String	No

Does an object move affect the value of the time stamp field?

Example profile: EUDAT profile

PID Record Structure Unification

- Document including the comments by TC at the TC Meeting in March/April 2016 –

Prepared by Merret Buurman (buurman@dkrz.de, PID Subtask 5.3.3), based on a series of VCs with Tobias Weigel, Claudio Cacciari, Carl Johann Håkansson, Shaun De Witt, Johannes Reetz, Mark Van De Sanden, Daan Broeder and Heinrich Widmann.

Versions of this document

Version	*Date*	*Changes*	*Changed by*
-/-	*-/-*	*Previous drafts were without version control.*	*M. Buurman (DKRZ)*
1.0	*2016-06-06*	• *Introduced changes suggested by TC:* • *Removed discarded alternative approaches to store alternative endpoints* • *Changed name of alternative endpoint field to EUDAT/AEP* • *Prepend "EUDAT/" in front of all type names* • *Added version number to profile*	*M. Buurman (DKRZ)*

Goal

We would like to make sure that all PID records created in one of the EUDAT CDIs has the same structure, i.e. a common set of mandatory fields, consistent naming of fields, and an agreement of the content of the fields. This structure will be well documented. Any desire for changes or for additional fields should be discussed and agreed upon to avoid inconsistencies and proliferation of new fields. On the long run, it is planned to replace the string types (such as "URL", "CHECKSUM",...) by PIDs registered in a Data Type Registry (DTR).

Current status

We have collected the different ways of using PIDs and the resulting requirements. These lead to a set of mandatory fields common for the services (see table below). The discussions about the optional fields are still ongoing.

Proposed mandatory fields

This is the set of mandatory fields we would like to see in every EUDAT-created PID record.

Type	Mandatory	Explanation	Content format	Comments
URL	Yes	HTTP access to the data object.	"http://..."	Should point to the object (i.e. PIDs for metadata objects point to the metadata object,

				PIDs for data objects point to the data object) Pointing to a landing page in case a human user is resolving the handle is desired, but will not be implemented based on the PID service, but rather by the HTTP API.
EUDAT/CHECKSUM	Yes	Checksum (MD5).	Any string.	EUDAT uses the MD5 checksum. If communities intend to use another checksum method, this information is not stored in the PID record.
EUDAT/CHECKSUM_ TIMESTAMP	Yes	The timestamp (date) of the checksum.	"20160215"	
EUDAT/FIXED_ CONTENT	Yes	Tells whether the referred data object may change or not (e.g. in case of errors or new versions). If True, any change, as small as it may be, needs to lead to a new PID.	True/False	It is important for users to know whether they can rely on always receiving the same version of the content when resolving a PID.
EUDAT/FIO	Yes (in replicas)	Acronym for "First Ingested Object": PID of the first copy that was ingested into the EUDAT CDI.	Handle: "prefix/suffix"	This will be the EUDAT-internal identifier of the data object. Before, the field EUDAT/ROR was used. However, as ROR may contain community identifiers that may change, it is

Example profile: EUDAT profile

Type	Mandatory	Explanation	Content format	Comments
URL	Yes	HTTP access to the data object.	"http://..."	Should point to the object (i.e. PIDs for metadata objects point to the metadata object, PIDs for data objects point to the data object) Pointing to a landing page in case a human user is resolving the handle is desired, but will not be implemented based on the PID service, but rather by the HTTP API.
EUDAT/CHECKSUM	Yes	Checksum (MD5).	Any string.	EUDAT uses the MD5 checksum. If communities intend to use another checksum method, this information is not stored in the PID record.
EUDAT/CHECKSUM_ TIMESTAMP	Yes	The timestamp (date) of the checksum.	"20160215"	
EUDAT/FIXED_ CONTENT	Yes	Tells whether the referred data object may change or not (e.g. in case of errors or new versions). If True, any change, as small as it may be, needs to lead to a new PID.	True/False	It is important for users to know whether they can rely on always receiving the same version of the content when resolving a PID.
EUDAT/FIO	Yes (in replicas)	Acronym for "First Ingested Object": PID of the first copy that was ingested into the EUDAT CDI.	Handle: "prefix/suffix"	This will be the EUDAT-internal identifier of the data object. Before, the field EUDAT/ROR was used. However, as ROR may contain community identifiers that may change, it is important to provide a non-changing identifier.
EUDAT/PARENT	Yes (in replicas)	Double-linked chain between "parents" and "children", i.e. replicas and replicated objects. PARENT leads to the parent PID, REPLICA leads to a replica. (See illustration on last page)	Handle: "prefix/suffix"	These two fields ensure the bidirectional link between replicas and their immediate parents (from where they were replicated). As the forward and backward links are PIDs and not URLs, the links are not expected to change (low maintenance). For technical reasons, the PID of the first ingested object ("FIO") cannot keep a list of all replicas. This is the reason for keeping a forward-backward chain between replicas and their "parents"
EUDAT/REPLICA	Yes (in PIDs of data objects that were		Handle: "prefix/suffix"	

Zoomed in on one example field from the EUDAT profile...

- Type: 12345/FIO
  - Final EUDAT DTR prefix yet undecided
- Mandatory: Yes (in replicas)
- Content format: Handle: prefix/suffix
- Acronym for First Ingested Object:
  PID of the first copy that was ingested into EUDAT.
- This will be the EUDAT-internal identifier of the data object. Before, the field EUDAT/ROR was used. However, as ROR may contain community identifiers that may change, it is important to provide a non-changing identifier.

Examples for services that exploit PID record information

- Verify the integrity of a data object at hand
- Check whether a new version of an object is available
- Investigate provenance across disciplines and repositories
- Build a custom citation reference for a bag of objects from different sources (shopping cart model)
- Services can take the form of command-line tools, web services, ...

More on this tomorrow...

How to create a good profile – some recommendations

- Get clear on your use cases – only put in the record what‘s needed
  - Don‘t overload profiles with subject metadata
  - Identify distinct metadata objects seperately instead of merging them into the record
- Maintain landing pages, keep locations to the actual data objects in separate fields
- Be prepared to deal with semantic issues
  - Ambiguous field names
  - CHECKSUM & CHECKSUM_METHOD vs. CHECKSUM_MD5
- Think ahead of your current scenario, if possible
  - Profile migration is expensive and needs to be carefully managed

What are policies and what are they used for?

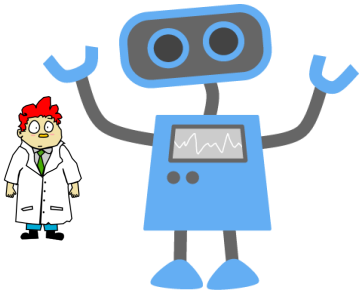
- Policy: Defines the allowed actions on objects
  - Can an object be deleted? Are there requirements for doing so?
  - Can object contents be replaced?
- Policies can be ordered by their strength
  - Weak policy: Object can be deleted.
  - Strong policy: Object can only be deleted under defined circumstances.
  - Even stronger policy: Object cannot be deleted.
- The policy of an object can never decrease in strength
- Policies are defined by repositories or depositors
  - They are registered objects themselves

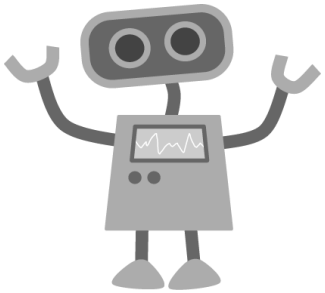
Would you include a policy attribute in the PID record?

What is the role of processes?

- Day-to-day management of objects
- Goals:
  - Reduce workload – work effectively and efficiently
  - Minimize human intervention – automate processes as far as possible
  - Despite the deluge, be able to account for every single object
  - Minimize errors

Example process: Object deposit and PID registration





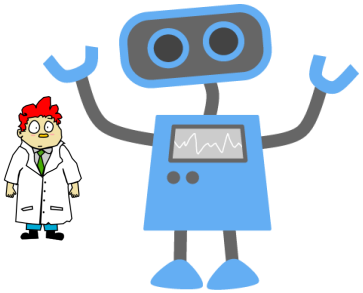
1. End-user **2. Repository agent** **3. PID registrar**

Task: An end-user deposits an object in a repository. The object is then registered.

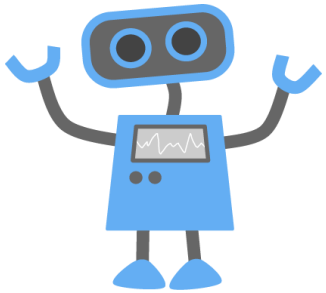
Upload object content → **Specify minimal metadata** → **Assign a PID to the object (subprocess)**

Example process: Object deposit and PID registration

1. End-user



2. Repository agent



3. PID registrar

Task: Register a PID for the deposited object.

Parameters: minimal metadata, repository default profile

Now: Break into groups!

- Show exercise doc here...