AIP Leibniz-Institut für Astrophysik Potsdam

# ProvDAL

## Retrieving provenance metadata

**IVOA Interoperability Meeting**

**October 2017, Santiago de Chile**

Kristin Riebe
Ole Streicher

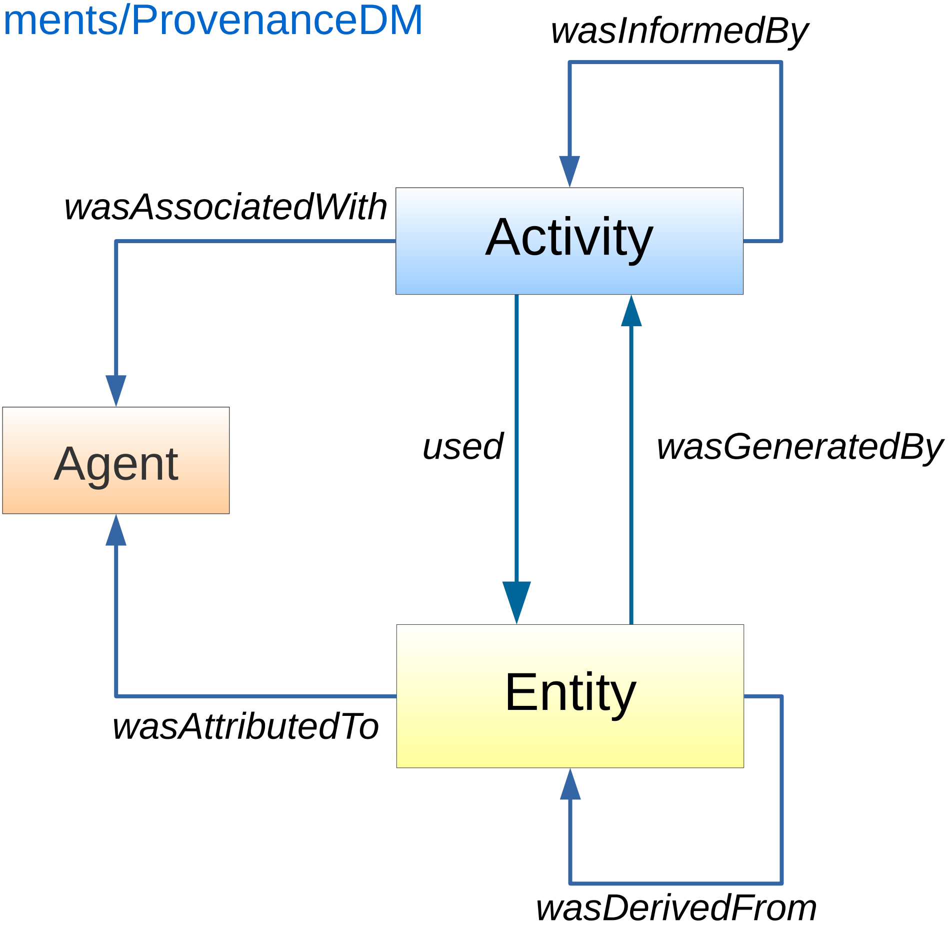
IVOA Data Model Working Group

German Astrophysical GAVO Virtual Observatory

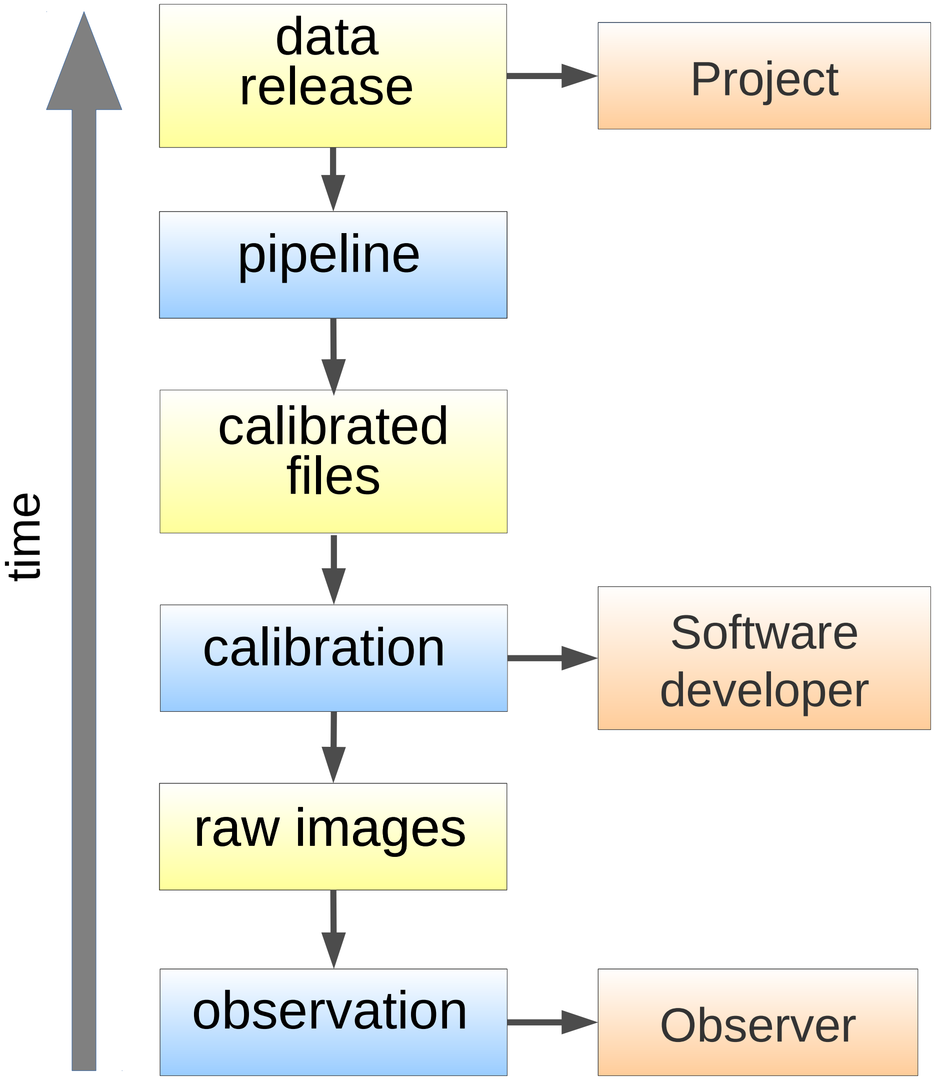
SPONSORED BY THE Federal Ministry of Education and Research

# ProvenanceDM

- Current draft: http://www.ivoa.net/documents/ProvenanceDM
- Defines 3 core classes:
  - Activity: observations, processing, ...
  - Entity: image, catalog, dataset, ...
  - Agent: observer, developer, ...
- And their relations + description classes
  - e.g. used, wasGeneratedBy, ...



# Example in astronomy



- Provenance is defined by the relations between data, activities and the people/projects involved
- Could be stored in relational or graph database
- How to access provenance metadata, when stored at a provenance web service?

# ProvenanceDM access protocols

- ProvDAL:
  - Retrieve provenance metadata
  - Simple DAL interface

- ProvTAP:
  - Explore provenance metadata
  - Advanced search functionalities

# ProvDAL - definition

- Interface for retrieving serialized provenance description for a given entity/activity/agent ID
- GET request with main parameter “ID”
- **Parameters:**

required

- **ID** *(of entity, activity or agent, can occur multiple times)*
- **DEPTH** *(= 1,2,... or ALL)*
- **RESPONSEFORMAT**
  *(PROV-N, PROV-JSON, PROV-XML, PROV-VOTable)*

optional

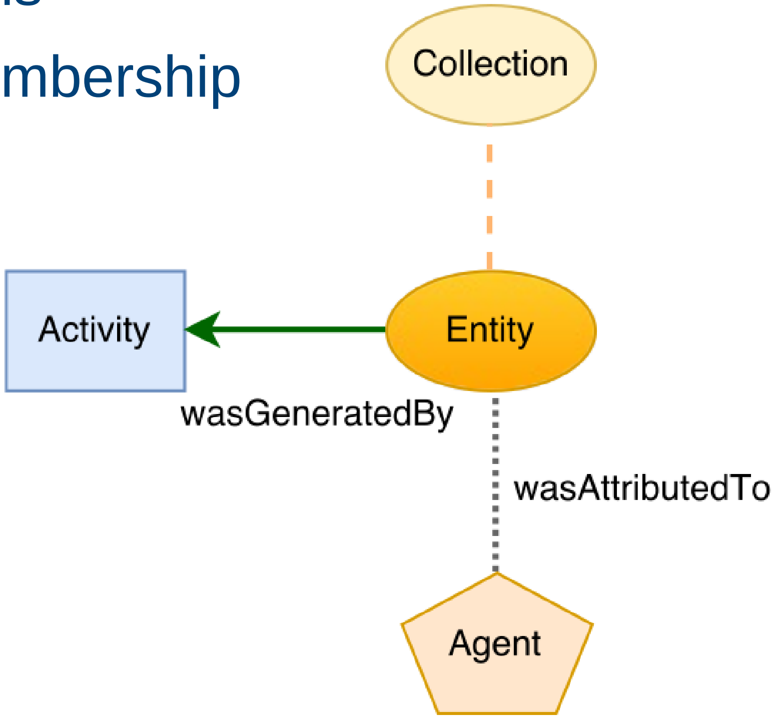
- DIRECTION *(= BACK or FORTH)*
- MEMBERS *(include members of collections)*
- STEPS *(include steps of activityFlows)*
- AGENT *(explore relations beyond agent)*
- MODEL *(= IVOA or W3C)*

# ProvDAL - Parameters

- **ID**
  - Identifier for an activity, entity or agent
- **RESPONSEFORMAT**
  - = format of the response
  - one of the W3C serialization formats (PROV-JSON, PROV-N, PROV-XML) or PROV-VOTable
- **DEPTH**
  - How much of the provenance graph shall be retrieved?
  - Everything (DEPTH=ALL) or just the most recent processing steps?
  - DEPTH=1: go exactly 1 relation backwards
  - DEPTH=ALL: services may also restrict to a max. depth instead (HTTP 302 redirect to DEPTH=<MAXDEPTH>)

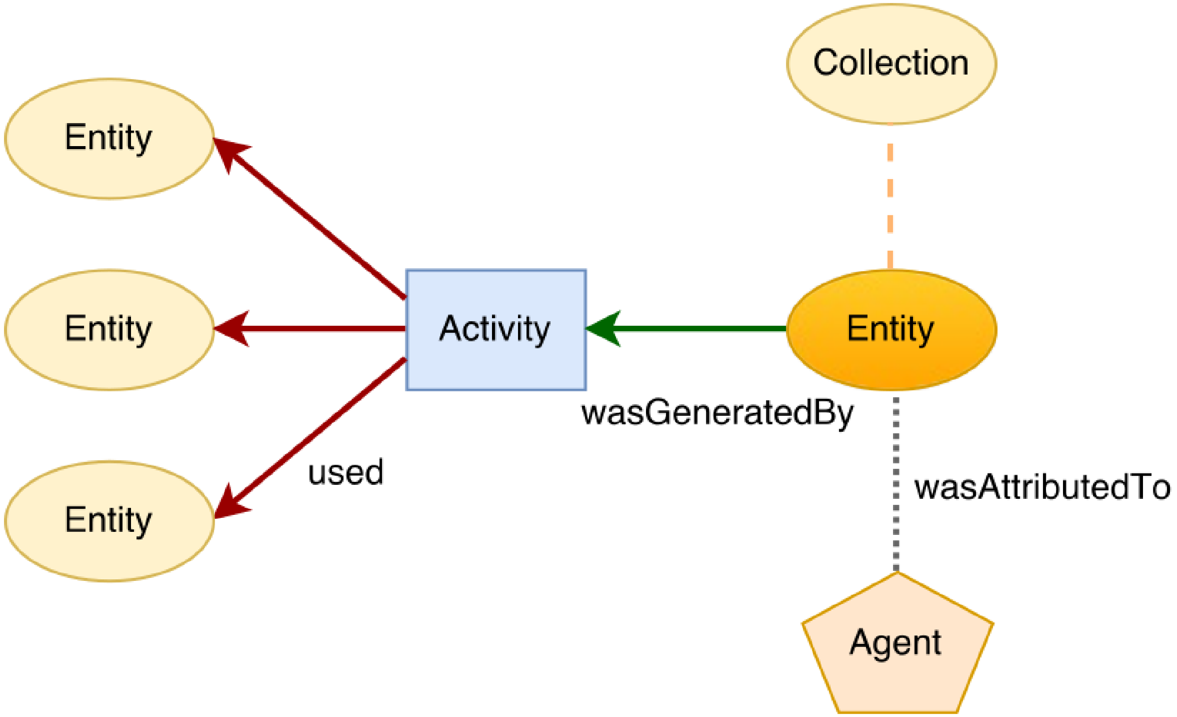
# ProvDAL – Parameter DEPTH

- DEPTH=1: start with given object (e.g. entity)
  - walk exactly one relation (back)
  - agent relations
  - collection membership



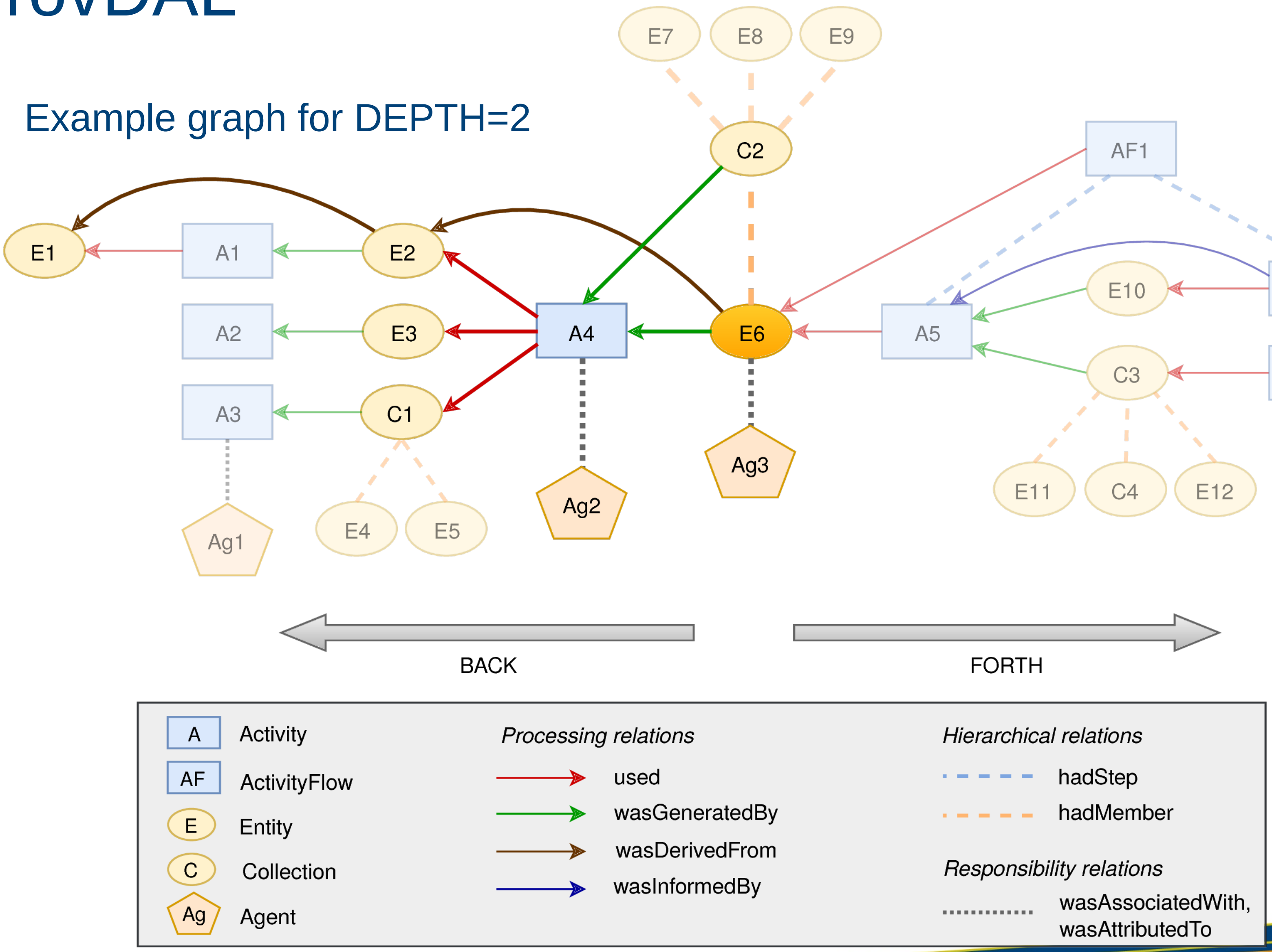
# ProvDAL – Parameter DEPTH

- DEPTH=2



# ProvDAL

- Example graph for DEPTH=2



A Activity

AF ActivityFlow

E Entity

C Collection

Ag Agent

*Processing relations*

→ used

→ wasGeneratedBy

→ wasDerivedFrom

→ wasInformedBy

*Hierarchical relations*

- - - hadStep

- - - hadMember

*Responsibility relations*

........ wasAssociatedWith, wasAttributedTo

# ProvDAL parameters

- DIRECTION = BACK or FORTH
  - Allow to track provenance forward, i.e. find out which processes used an image, which images were derived from a certain image, what output files an activity produced etc.
  - Use cases
    - pipeline development
    - bug tracking
  - Only affects the processing relations:
    - Used
    - WasGeneratedBy
    - Because FORTH/BACK makes not much sense for e.g. hadMember or wasAttributedTo relations; thus the hierachical/responsibility relations are always tracked, independent of DIRECTION

# ProvDAL parameters

- MEMBERS, STEPS = true/false
  - Collection groups entities together  
    => hadMember relationship
  - ActivityFlow groups activities together  
    (e.g. pipeline, workflow)  
    => hadStep relationship
- If tracking members of collections and activityFlows by default, a lot of data is returned
- => always follow the relations “up” (to the “container”), but only follow the “children”, if MEMBERS=true or STEPS=true

# ProvDAL parameters

- AGENT = true/false
  - Usually stop tracking when an agent is reached,
    but maybe want to know which other activities/entities an agent was involved with?
  - => allow tracking the agent further, using AGENT=true
- Discussion:
  - AGENT = false may be misleading
  - Better ideas?
    - EXPLORE_AGENT = true/false
    - TRACK_AGENT = true/false
    - AGENT = STOP/EXPLORE

# ProvDAL parameters

- Discussion:
  - Rather use one parameter for each relation with 4 values?
    - both, none and up/down or back/forth or to/from (depending on type)
    - e.g.
      - Used=BOTH: track used relationship in both directions
      - WasAttributedTo = to: just go to an agent and stop there
  - => would provide much more flexibility, more powerful extraction of provenance
  - => would increase number of parameters from 8 to 13
  - => interface would become more complex
  - => more “loops” in querying, thus need to be careful with implementations

# ProvDAL parameters

- MODEL:
  - Allow to choose between IVOA and W3C serialization
  - IVOA:
    - directly map the classes to JSON, VOTable, …
    - For exchange in the VO
    - To be used with VO tools, e.g. for loading into a ProvTAP service for further querying
  - W3C:
    - rename and restructure classes and attributes to produce W3C compatible serialization
    - For exchange with the world outside of the VO
    - For usage with W3C tools (e.g. ProvStore)

# ProvDAL implementation

- Created a prototype web application, using Django framework (Python)
- Implements ProvenanceDM classes, relational database tables
  (no description classes and parameters, so far)
- Implements **ProvDAL** interface
- Live version for RAVE:
  - https://escience.aip.de/provenance-rave
- Decoupled django-prov_vo package as reusable web app:
  - https://github.com/kristinriebe/django-prov_vo

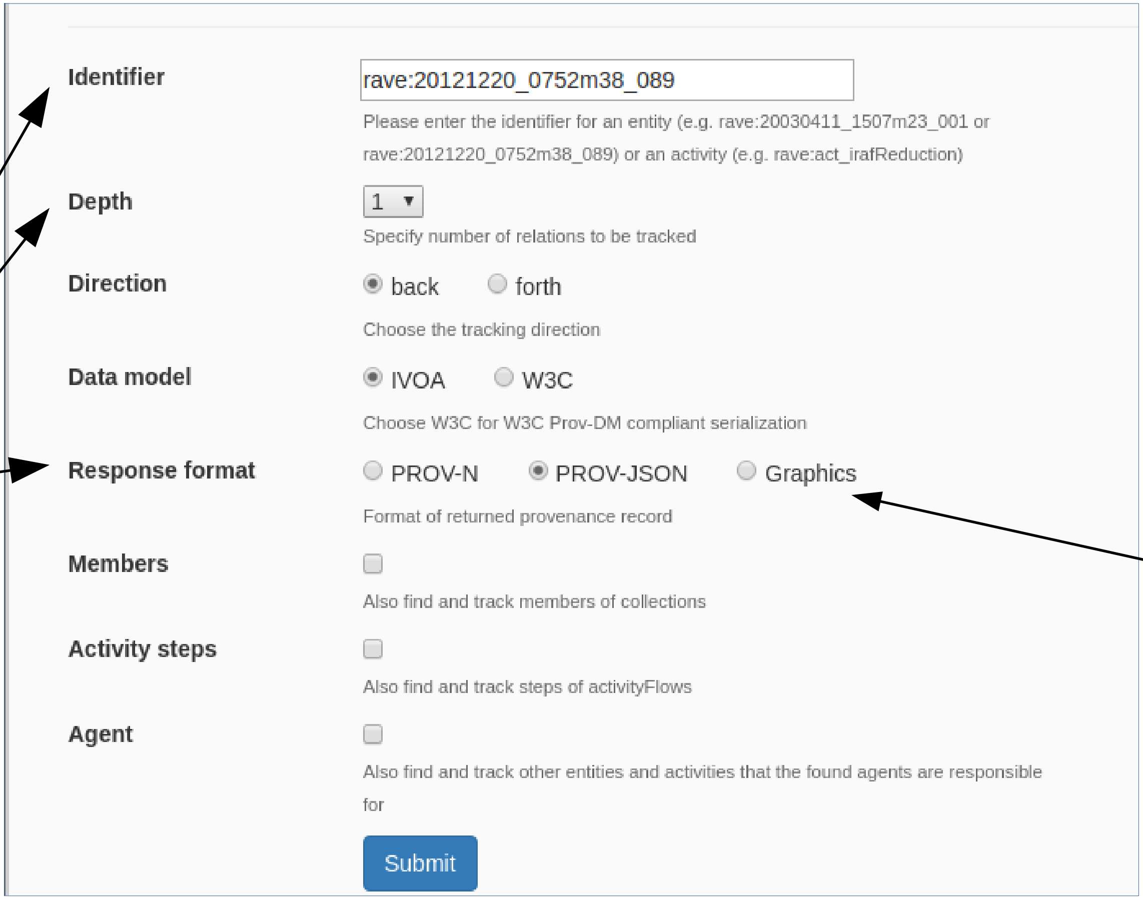
    and an extra package for the VOSI resources (availability/capabilities):
  - https://github.com/kristinriebe/django-vosi

# ProvDAL implementation

- Implemented all parameters from the draft
- Recursive tracking of the relations
- Each visited node of the provenance graph is returned only once (It’s a graph, not a tree → loops possible!)
- Allows W3C compatible serialization (model=W3C)
- Formats: PROV-N or PROV-JSON

- Additionally:
  - Visualization of provenance (Javascript)
    - option RESPONSEFORMAT=GRAPH
  - Web form for nice user interface

# ProvDAL webform



Identifier: rave:20121220_0752m38_089

Please enter the identifier for an entity (e.g. rave:20030411_1507m23_001 or rave:20121220_0752m38_089) or an activity (e.g. rave:act_irafReduction)

Depth: 1 ▾

Specify number of relations to be tracked

Direction: ● back ○ forth

Choose the tracking direction

Data model: ● IVOA ○ W3C

Choose W3C for W3C Prov-DM compliant serialization

Response format: ○ PROV-N ● PROV-JSON ○ Graphics

Format of returned provenance record

Members: □

Also find and track members of collections

Activity steps: □

Also find and track steps of activityFlows

Agent: □

Also find and track other entities and activities that the found agents are responsible for

mandatory parameters

additional option

Automatically generates the ProvDAL GET request URL: https://escience.aip.de/provenance-rave/provapp/provdal/?ID=rave:20121220_0752m38_089&DEPTH=1&RESPONSEFORMAT=PROV-JSON&DIRECTION=BACK&MODEL=IVOA&MEMBERS=false&STEPS=false&AGENT=false

# Questions? Ideas?