1 Accessing provenance information

1.1 State of ProvDAL at Paris meeting, July 2017

1.1.1 ProvDAL

ProvDAL is a service the interface of which is organized around one main parameter, the "ID" of an entity (obs_publisher_did of an ObsDataSet for example) or activity. The response is given in one of the following formats: PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE. Additional parameters can complete ID to refine the query: FORMAT allows to choose the output format. BACKWARD gives the number of relations that shall be tracked in backward direction, i.e. along the provenance history. Its value is either 0, a positive integer or ALL. If this parameter is omitted, the default is ALL, wich returns the complete provenance history. The optional parameter FORWARD defines the number of forward relations; it's also either a positive integer or ALL, but default is 0. That means if neither FORWARD nor BACKWARD are specified, then the complete provenance history is returned.

The ID parameter is allowed more than once in order to retrieve several data set provenance details at the same time. An example request could look like this:

```
{provdal-base-url}?ID=rave:dr4&BACKWARD=1&FORMAT=PROV-JSON
```

Each of the provenance relation has a direction, BACKWARD follows these directions whereas FORWARD follows the relations in reverse direction, independent of the relation type. This is easier to implement, but has the (for a user unexpected) side effect that e.g. agent relations are only retrieved when using BACKWARD, but never with FORWARD. Similarly for membership (hadStep, hadMember) relations: members of a collection or activityFlow are retrieved only in BACKWARD direction, and collections or activityFlows that contain an entity or activity are only found in FORWARD direction. In order to provide a more user-friendly interface with less surprising behaviour, we define three more request parameters: EXPAND_AGENT, EXPAND_COLLECTION and EXPAND_ACTIVITYFLOW. They take TRUE or FALSE as arguments. If they are set to TRUE, the relations with agents, collections and activityFlows will be included in any case, independent of the direction in which the provenance graph is retrieved.

TODO:
Draw a provenance graph picture here with different relation types and arrows for direction.

TODO:
Implementations need to show if this is really the best way.

TODO:
If EXPAND_AGENT=TRUE: include all agent relations, but if EXPAND_AGENT=FALSE, then use default behaviour? Or do not include any of the agent relations? Which one would it be?

A ProvDAL service MUST implement the parameters ID, BACKWARD and FORMAT; the remaining parameters are optional. If a service does not implement the optional parameters, but they appear in the request, then the service should return with an error.

Table 2 summarizes the parameters for such a ProvDAL service interface.

Parameter	**Requirement**	**Value/options**	**Default**	**Description**
ID	required	qualified ID	–	a valid qualified identifier for an entity or activity (can occur multiple times)
BACKWARD	required	0,1,2,..., ALL	ALL	number of relations to be followed backwards or `ALL` for everything
FORWARD	optional	0,1,2,..., ALL	0	number of relations to be followed forward or `ALL` for everything
FORMAT	required	PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE	?	serialisation format of the response
EXPAND_AGENT	optional	TRUE or FALSE	TRUE	include agent relations in any case
EXPAND_COLLECTION	optional	TRUE or FALSE	TRUE	include relations with collections in any case
EXPAND_ACTIVITYFLOW	optional	TRUE or FALSE	TRUE	include relations with activityFlows in any case

Table 1: ProvDAL request parameters

1.2 New proposal after Paris meeting, after first implementation

1.2.1 ProvDAL

ProvDAL is a service the interface of which is organized around one main parameter, the **ID** of an entity (obs_publisher_did of an ObsDataSet for example), activity or an agent. The response is given in one of the following formats: PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE. Additional parameters can complete the ID to refine the query: **FORMAT** allows to choose the output format. **DEPTH** gives the number of relations that shall be tracked along the provenance history, independent of the type of relation. Its value is either 0, a positive integer or ALL. If this parameter is omitted, the default is ALL, which returns the complete provenance history that the service has stored or the provenance according to a maximum depth number that the server allows.

The ID parameter is allowed more than once in order to retrieve provenance details for several activities or datasets at the same time. Here are a few example requests:

```
{provdal-base-url}?ID=rave:dr4&FORMAT=PROV-JSON
{provdal-base-url}?ID=rave:dr4&ID=rave:act_irafReduction&DEPTH=2
```

The format can also be specified via the HTTP accept header, e.g.

```
wget -d --header="Accept: application/json" \
   {provdal-base-url}?ID=rave:dr4
```

would return the provenance information in PROV-JSON format. If both FORMAT and the accept header are used and FORMAT specifies a format that is incompatible with the HTTP accept header, then the service should return with a HTTP status 406: Not Acceptable.

For services which allow tracking the provenance information forward, e.g. in order to check for which activities an entity was used, the optional parameter **DIRECTION** can be set to FORTH. Its default value is BACK. This influences the direction in which the used, wasGeneratedBy, wasDerivedFrom and wasInfluencedBy relations are followed.

The provenance data model defines also the hierarchical relations *hadMember* for entity collections and *hadStep* for activityFlows. If a node belongs to a collection or activityFlow, these relations shall be returned as well, independent of the specified tracking direction. If one is interested in more details and wants to follow the *members* of an entity collection or the *steps* of an activityFlow, these can be included by setting the optional parameter **MEMBERS** or **STEPS** to TRUE, respectively. The default is FALSE.

By default, it is recommended to stop any further tracking at an agent node, unless an additional optional parameter **AGENT** is set to TRUE. Note that this means that the request for any agent will always return just the

agent node itself and nothing else, unless AGENT=TRUE is used. Thus, if one wants to know which entities and activities an agent has influenced, the request looks like this:

```
{provdal-base-url}?ID=org:rave&AGENT=TRUE&DEPTH=1
```

DEPTH=1 was used here in order to avoid following the found entities and activities any further.

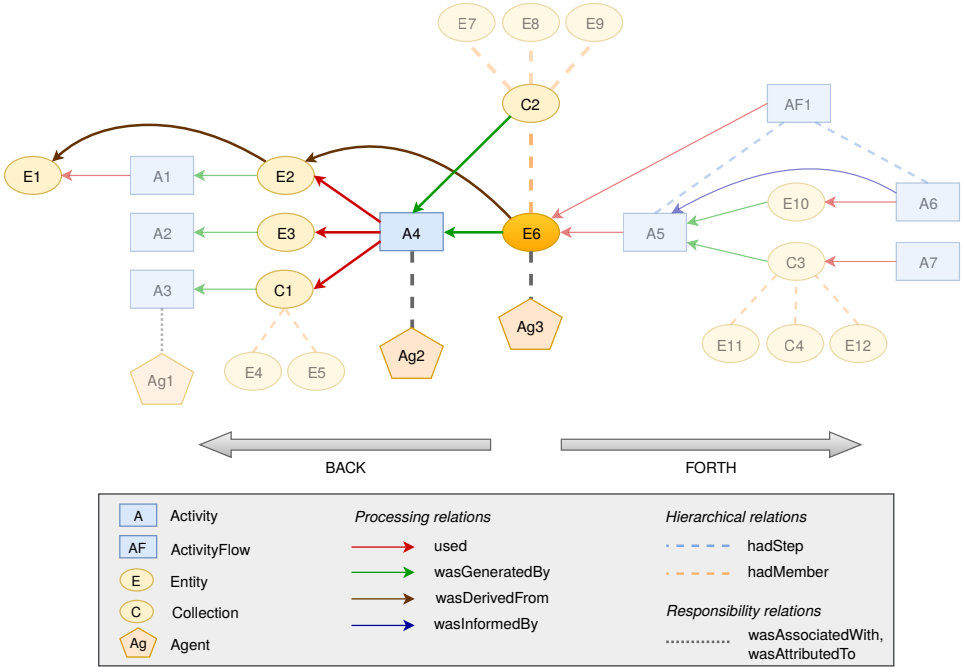

Figure 1: An example provenance graph, highlighting the objects and relations returned from a ProvDAL service with ID=E6 and DEPTH=2. The BACK and FORTH values for DIRECTION are only important for the processing relations (solid lines). Hierarchial (dashed) and responsibility relations (dotted) are only followed "upwards" and towards agents by default. If they should also be followed in the other direction, then the additional optional parameters MEMBERS, STEPS and AGENT need to be set to TRUE.

A ProvDAL service MUST implement the parameters ID, DEPTH and FORMAT; the remaining parameters are optional. If a service does not implement the optional parameters, but they appear in the request, then the service should return with an error.

Table 2 summarizes the parameters for such a ProvDAL service interface.

Parameter	Value/options	Description
ID	qualified ID	a valid qualified identifier for an entity or activity (can occur multiple times)
DEPTH	0,1,2,..., ALL	number of relations to be followed or `ALL` for everything, independent of the relation type
FORMAT	PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE	serialisation format of the response
DIRECTION	BACK, FORTH	BACK = track the provenance history, FORTH = explore the results of activities and where entities have been used
MEMBERS	TRUE or FALSE	if TRUE, retrieve and track members of collections
STEPS	TRUE or FALSE	if TRUE, retrieve and track steps of activityFlows
AGENT	TRUE or FALSE	if TRUE, retrieve all relations for agents, i.e. find out what an agent is responsible for

Table 2: ProvDAL request parameters. Options that are **required** to be implemented by ProvDAL services are marked with bold face. Default values are underlined.