IVOA Provenance Simple Access Protocol (ProvSAP)

Version 1.0

IVOA Working Draft 2018-09-26

Working group
DM
This version
http://www.ivoa.net/documents/ProvSAP/20180926
Latest version
http://www.ivoa.net/documents/ProvSAP
Previous versions

Author(s)
François Bonnarel, Kristin Riebe, Mathieu Servillat, Mireille Louys, Markus Nullmeier, Florian Rothmaier, Michèle Sanguillon, IVOA Data Model Working Group
Editor(s)
Kristin Riebe

Abstract

This document describes the ProvSAP protocol for accessing provenance information according to the IVOA ProvenanceDM standard. It is a simple, light weight protocol. For extensive querying features, please see the ProvTAP protocol.

Status of This Document

This is an IVOA Working Draft for review by IVOA members and other interested parties. It is a draft document and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use IVOA Working Drafts as reference materials or to cite them as other than "work in progress".

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

1 Introduction

We envision two access protocols:

- ProvSAP: retrieve provenance information based on given ID of a data entity or activity.
- ProvTAP: allows detailed queries for provenance information, discovery of datasets based on e.g. code version. ProvTAP will be desribed in a separate standard document.

There are two main reasons for ProvSAP:

- ProvSAP allows users to retrieve serializations for a given entity, activity or agent as W3C compatible serializations in one of the supported formats. This makes the provenance metadata that are provided via a ProvSAP interface very useful, since it can even be processed by tools that understand W3C provenance metadata.

- Projects that have their own internal provenance data structure may not want to or cannot completely reorganize their database to comply with the ProvTAP specification. However, a ProvSAP layer on top of their individual data structure is less of an overhead and gives the opportunity to share the provenance data in a common serialization format with other tools.

Other reasons are:

- ProvSAP is a rather simple protocol and easy to implement
- ProvSAP returns valid IVOA or W3C serializations, which can be processed or uploaded with other tools, or also placed into FITS-headers (e.g. in an extra block)
- ProvSAP urls with ID parameter could be used as a point of reference, for refering to the provenance for a given entity.

2 ProvSAP

ProvSAP is a simple data access layer interface (see DALI specification of the VO, Dowler and Demleitner et al., 2013) that can be implemented by a web service to serve provenance information to a client. The client sends a GET request to the basic URL endpoint (`{provsap-base-url}`) of a ProvSAP service, providing at least the main parameter **ID**, the (unique, qualified) identifier of an entity (obs_publisher_did of an ObsDataSet for example), activity or an agent. This parameter can occur more than once in a request in order to retrieve provenance details for several activities, datasets or agents at the same time. Here are two simple example requests:

```
{provsap-base-url}?ID=rave:dr4
{provsap-base-url}?ID=rave:dr4&ID=rave:act_irafReduction
```

Additional parameters can complete the request to refine the query. They are described in the next paragraphs and summarized in Table 1.

RESPONSEFORMAT The format of the response can be defined using the RESPONSEFORMAT parameter. Its value is one of the provenance serialization formats: PROV-N, PROV-JSON, PROV-XML, PROV-VOTABLE.

DEPTH The DEPTH parameter 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 1, which returns all relations and nodes that can be reached by

Parameter	**Values**	**Description**
ID	qualified ID	a valid qualified identifier for an entity, activity or agent (can occur multiple times)
DEPTH	0,1,2,..., ALL	number of relations to be followed or `ALL` for everything, independent of the relation type
RESPONSEFORMAT	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 (1) or false (0)	if true/1, retrieve and track members of collections
STEPS	true (1) or false (0)	if true/1, retrieve and track steps of activityFlows
AGENT	true (1) or false (0)	if true/1, explore all relations for agents, i.e. find out what an agent is responsible for
MODEL	IVOA or W3C	compatibility of the serialization to the IVOA or W3C provenance data model

Table 1: ProvSAP request parameters. Options that are **required** to be implemented by ProvSAP services are marked with bold face. Default values are underlined. The parameter names are case-insensitive, but the parameter values are not.

following 1 relation. If `DEPTH=ALL` is requested, the server should return the complete provenance history that the service has stored for the given entity, activity or agent. Services may restrict the returned data by redirecting `DEPTH=ALL` to `DEPTH={maxdepth}`, where `{maxdepth}` is an integer defining the maximum depth number that the server allows.

If description classes (EntityDescription etc.) are used, we expect that the descriptions are retrieved together with each corresponding main class. For example, for each entity node also return the entityDescription – either by adding the description attributes to the entity in the serialized response, by adding a link to a URL with entityDescription information or by returning a direct serialization of the description class along with a link from the entity to the entityDescription.

Note that the relations *wasDerivedFrom* and *wasInformedBy* are "shortcuts" in a provenance graph. Thus for e.g. `DEPTH=2` more progenitors of an entity may be reached via *wasDerivedFrom* relations than via the "long path" along the corresponding *used* and *wasGeneratedBy* relations (see e.g. progenitor entity E1 in Figure 1). (A better solution for the future may be to use 1/2*DEPTH for walking along these short-cut relations, but we don't want to make ProvSAP more complex for now.)

DIRECTION 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 only influences the direction in which the used, wasGeneratedBy, wasDerivedFrom and wasInfluencedBy relations are followed. Any other relations are tracked according to the behaviour specified below, independent of the DIRECTION value.

Figure 1 shows an example provenance graph with different relations and nodes. Only the relations marked by solid lines are influenced by the DIRECTION parameter. A ProvSAP GET request with `ID=E6` and `DEPTH=2` returns only the highlighted nodes and relations (thick lines) by default.

MEMBERS, STEPS The provenance data model defines 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 someone 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. As detailed in DALI (Dowler and Demleitner et al., 2013), the values 1 and 0 are equivalent to true and false.

AGENT 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. An example request if one wants to know which entities and activities an agent has influenced could look like this:
`{provsap-base-url}?ID=org:rave&AGENT=true&DEPTH=1`.
`DEPTH=1` is used here in order to avoid following the found entities and activities any further (can be omitted, since this is the default for DEPTH).

MODEL If a provenance service supports it, the MODEL parameter is used to distinguish between serializations compatible with the IVOA or the W3C provenance data model. The default value is IVOA.

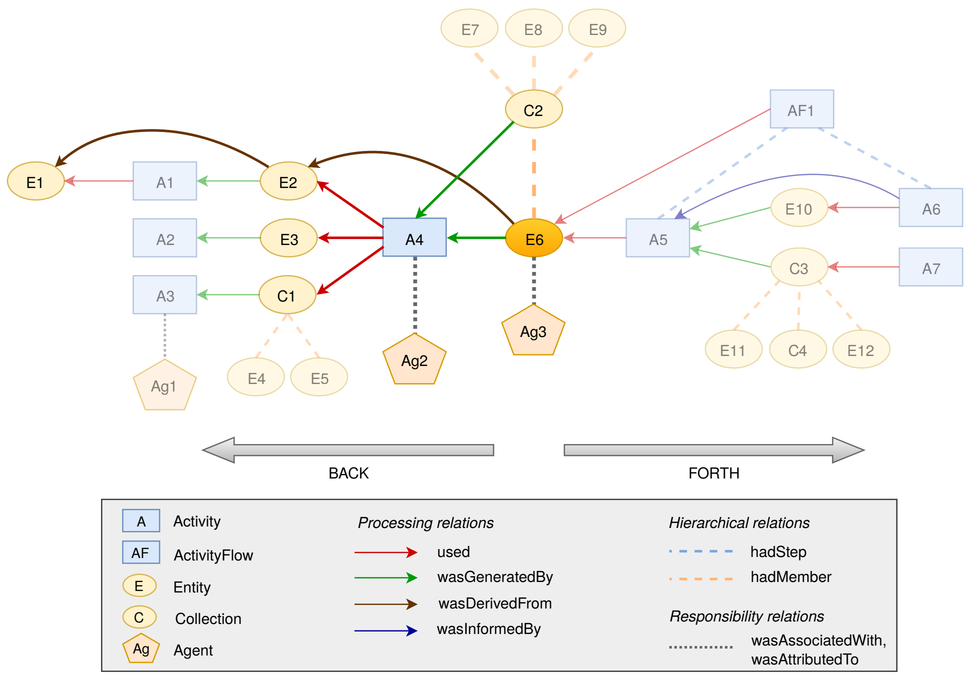

Figure 1: An example provenance graph, highlighting the objects and relations returned from a ProvSAP service with `ID=E6` and `DEPTH=2`. The BACK and FORTH values for DIRECTION are only important for the processing relations (solid lines). Hierarchial (dashed) and responsibility (dotted) relations are only followed "upwards" (to collection/activityFlow) and towards agents by default (unless the optional parameters MEMBERS, STEPS and/or AGENT are set to true.

A ProvSAP service MUST implement the parameters ID, DEPTH and RESPONSEFORMAT; 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. Please note that according to the DALI specification (Dowler and Demleitner et al., 2013), the parameter names are case-insensitive, but the parameter values are not. E.g. `direction=FORTH` is allowed, but `DIRECTION=forth` may not work.

3 Example use cases – maybe expand here, or skip

Here are a few example use cases for ProvSAP in order to show its usefulness for astronomical datasets.

- The RAVE DR4 release contains a main table with stellar properties for each observation of a star. Given the RAVE observation ID, retrieve all the processing steps for this specific observation result:

```
{provsap-base-url}?ID=rave:20121220_0752m38_089&DEPTH=ALL
```

The result will not only contain the processing steps (activities), but also entities and agents. The information that the user is actually interested in can be filtered out by a client application (e.g. using the voprov python package). If a W3C tool shall be used, e.g. when the result shall be loaded to ProvStore[1] for further processing, one needs to retrieve the information in a W3C compatible way like this:

```
{provsap-base-url}?ID=rave:20121220_0752m38_089&DEPTH=ALL&MODEL=W3C
```

- Get the direct progenitor of an entity:

  ```
  {provsap-base-url}?ID=rave:20121220_0752m38_089&DEPTH=1
  ```

  If this request only returns a collection and no "backwards" information about progenitors, then one needs to track the collection further, i.e. repeat the request for the collection entity.

- Get all datasets that were derived from a specific data file in the CTA pipeline:

  ```
  {provsap-base-url}?ID=cta:df1&DEPTH=ALL&DIRECTION=FORTH
  ```

ProvSAP is meant to be used to just retrieve parts of a provenance graph from a provenance web service. It cannot be used to explore the provenance graphs and filter the information based on specific properties, e.g. the creationTime of an entity or a parameter value for an activity. For such cases, a ProvTAP service can be used – which is described in the next section.

4 VOSI availability and capabilities

According to the DALI specification for VO services (Dowler and Demleitner et al., 2013), a provenance service implementing ProvTAP must provide a VOSI availability interface as well as a capabilities interface with entries for ProvTAP. The `standardIds` for these provenance interfaces are:

```
ivo://ivoa.net/std/ProvenanceDM#ProvSAP-1.0
```

The capability for a DAL service to support the ProvenanceDM is expressed by the dataModel element as:

```
<dataModel ivoid="ivo://ivoa.net/std/ProvenanceDM-1.0">
  ProvenanceDM-1.0
</dataModel>
```

[1]https://provenance.ecs.soton.ac.uk/store/

A Changes from Previous Versions

Bibliography

Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2013), 'Data access layer interface, version 1.0', IVOA Recommendation.
`http://www.ivoa.net/documents/DALI/20131129/`