International
Virtual
Observatory
Alliance

IVOA Provenance Data Model Version 1.0

IVOA Working Draft 2018-05-30

Working group
DM
This version
http://www.ivoa.net/documents/ProvenanceDM/20180530
Latest version
http://www.ivoa.net/documents/ProvenanceDM
Previous versions
WD-ProvenanceDM-1.0-20170921.pdf
WD-ProvenanceDM-1.0-20161121.pdf
ProvDM-0.2-20160428.pdf
ProvDM-0.1-20141008.pdf
Author(s)
Kristin Riebe, Mathieu Servillat, François Bonnarel, Anastasia Galkin, Mireille Louys, Markus Nullmeier, Florian Rothmaier, Michèle Sanguillon, Ole Streicher, IVOA Data Model Working Group
Editor(s)
Kristin Riebe, Mathieu Servillat

Abstract

This document describes how provenance information for astronomical datasets can be modeled, stored and exchanged within the astronomical community in a standardized way. We follow the definition of provenance as proposed by the W3C[1], i.e. that "provenance is information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness." Such provenance information in astronomy is important to enable any scientist to trace back the origin of a dataset (e.g. an image, spectrum, catalog or single points in a spectral energy distribution diagram or a light curve), learn about the people and organizations involved in a project and assess the quality of the dataset as well as the usefulness of the dataset for her own scientific work.

Status of this document

Contents

Acknowledgments

This document has been developed in part with support from the German Astrophysical Virtual Observatory, funded by BMBF Bewilligungsnummer 05A14BAD and 05A08VHA. The Provenance Working Group acknowledges support from the Astronomy ESFRI and Research Infrastructure Cluster – ASTERICS project, funded by the European Commission under the Horizon 2020 Programme (GA 653477).

Thanks for fruitful discussions to: Catherine Boisson, Karl Kosak and Jose Enrique Ruiz for the binding to the Cerenkov Telescope Array (CTA) project, Gerard Lemson and Laurent Michel for the VODML expression of the data model, Markus Demleitner, Harry Enke, Jochen Klar and Adrian

Partl for fruitful discussions, remarks and comments during the different stages of this specification.

Conformance-related definitions

The words "MUST", "SHALL", "SHOULD", "MAY", "RECOMMENDED", and "OPTIONAL" (in upper or lower case) used in this document are to be interpreted as described in IETF standard, Bradner (1997).

The *Virtual Observatory (VO)* is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

In this document, we discuss a draft of an IVOA standard data model for describing the provenance of astronomical data. We follow the definition of provenance as proposed by the W3C (Belhajjame and B'Far et al., 2013), i.e. that provenance is "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability or trustworthiness".

In astronomy, such entities are generally datasets composed of VOTables, FITS files, database tables or files containing values (spectra, light curves), logs, parameters, etc. The activities correspond to processes like an observation, a simulation, or processing steps (image stacking, object extraction, etc.). The people involved can be individual persons (observer, publisher, . . .), groups or organisations. An example for activities, entities and agents as they can be discovered backwards in time is given in Figure 1.

The currently discussed Provenance Data Model is sufficiently abstract that its core pattern could be applied to any kind of process using either observation or simulation data. It could also be used to describe the workflow for observation proposals or the publication of scientific articles based on (astronomical) data. However, here we focus on astronomical data. The links between the Provenance Data Model and other IVOA data models will be discussed in Section B. We note here that the provenance of simulated data is already covered by the Simulation Data Model (SimDM, Lemson and Wozniak et al., 2012). Therefore we also give a mapping between SimDM and the Provenance Data Model in Section B.

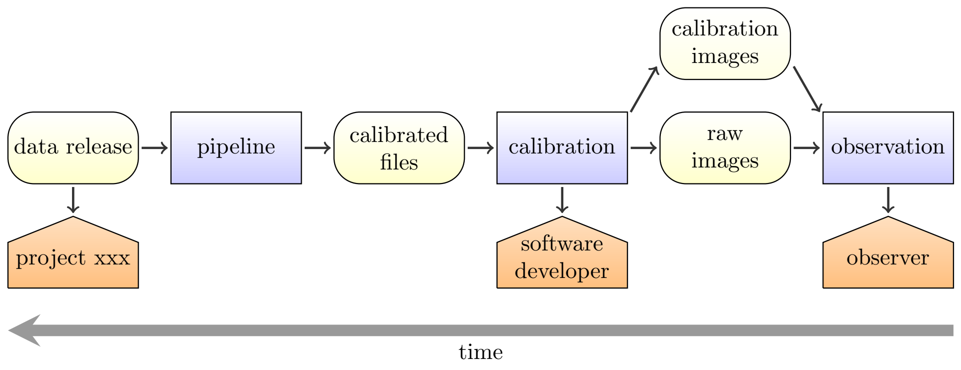

Figure 1: An example graph of provenance discovery. Starting with a released dataset (left), the involved activities (blue boxes), progenitor entities (yellow rounded boxes) and responsible agents (orange pentagons) are discovered.

1.1 Goal of the provenance model

The goal of this Provenance Data Model is to describe how provenance information can be modeled, stored and exchanged. Its scope is mainly modeling of the flow of data, of the relations between data, and of processing steps.

Characteristics of observation activities such as ambient conditions and instrument characteristics can be associated to provenance information. Experimental configuration or contextual information during the execution of processing activities (computer structure, nodes, operating system used, . . .) can also be connected to provenance information. However, they will not be modeled here explicitly. This additional information can be included in the form of metadata or entities linked to those activities.

In general, the model shall capture information in a machine-readable way that would enable a scientist who has no prior knowledge about a dataset to get more background information. This will help the scientist to decide if the dataset is adequate for her research goal, assess its quality and get enough information to be able to trace back its history as far as required or possible.

Provenance information may be recorded in minute detail or by using coarser elements, depending on the intended usage and the desired level of detail for a specific project that records provenance. This granularity depends on the needs of the project and the intended usage when implementing a system to track provenance information.

The following list is a collection of tasks which the Provenance Data Model should help to solve. They are flagged with [S] for problems which are more interesting for the end user of datasets (usually a scientist) and with [P] for tasks that are probably more important for data producers and publishers. More specific use cases in the astronomy domain for different types of datasets and workflows along with example implementations are given in the Implementation Note (Riebe and Servillat et al., 2017).

A: Tracking the production history [S]

Find out which steps were taken to produce a dataset and list the methods/tools/software that were involved. Track the history back to the raw data files / raw images, show the workflow (backwards search), or return a list of progenitor datasets.
Examples:

- Is an image already calibrated? What about dark field subtraction? Were foreground stars removed? Which technique was used?
- Is the background noise of atmospheric muons still present in my neutrino data sample?

We do not go as far as to consider easy reproducibility as a use case – this would be too ambitious. But at least the major steps undertaken to create a piece of data should be recoverable.

B: Attribution and contact information [S]

Find the people involved in the production of a dataset, the people/organizations/institutes that need to be cited or can be asked for more information.
Examples:

- I want to use an image for my own work – who was involved in creating it? Who do I need to cite or who can I contact to get this information? Is a license attached to the data?
- I have a question about column xxx in a data table. Who can I ask about that?
- Who should be cited or acknowledged if I use this data in my work?

C: Locate error sources [S, P]

Find the location of possible error sources in the generation of a dataset.
Examples:

- I found something strange in an image. Where does the image come from? Which instrument was used, with which characteristics, etc.? Was there anything strange noted when the image was taken?
- Which pipeline version was used – the old one with a known bug for treating bright objects or a newer version?

- This light curve doesn't look quite right. How was the photometry determined for each data point?

D: Quality assessment [P]

Judge the quality of an observation, production step or dataset.
Examples:

- Since wrong calibration images may increase the number of artifacts on an image rather than removing them, knowledge about the calibration image set will help to assess the quality of the calibrated image.

E: Search in structured provenance metadata [P, S]

This would allow one to also do a "forward search", i.e. locate derived datasets or outputs, e.g. finding all images produced by a certain processing step or derived from data which were taken by a given facility.
Examples:

- Give me more images that were produced using the same pipeline.
- Give me an overview on all images reduced with the same calibration dataset.
- Are there any more images attributed to this observer?
- Which images of the Crab Nebula are of good quality and were produced within the last 10 years by someone not from ESO or NASA?
- Find all datasets generated using this given algorithm for this given step of the data processing.

This task is probably the most challenging. It also includes tracking the history of data items as in A, but we still have listed this task separately, since we may decide that we can't keep this one, but we definitely want A.

1.2 Minimum requirements for provenance

We derived from our goals and use cases the following minimum requirements for the Provenance Data Model:

- Provenance information must be stored in a standard model, with standard serialization formats.
- Provenance information must be machine readable.
- Provenance data model classes and attributes should be linked to other IVOA concepts when relevant (DatasetDM, ObsCoreDM, SimDM, VOTable, UCDs, ...).

- Provenance information should be serializable into the W3C provenance standard formats (PROV-N, PROV-XML, PROV-JSON) with minimum information loss.
- Provenance metadata must contain information to find immediate progenitor(s) (if existing) for a given entity, i.e. a dataset.
- An entity must point to the activity that generated it (if the activity is recorded).
- Activities must point to input entities (if applicable).
- Activities may point to output entities.
- Provenance information should make it possible to derive the chronological sequence of activities.
- Provenance information can only be given for uniquely identifiable entities, at least inside their domain.
- Released entities should have a main contact.
- It is recommended that all activities and entities have contact information and contain a (short) description or link to a description.

1.3 Role within the VO architecture

The IVOA Provenance Data Model is structuring and adding metadata to trace the original process followed during the data production for providing astronomical data. Even if it borrows the main general concepts from data management science, it binds to the specific context of astronomical metadata description and re-uses or interacts with existing IVOA models. It takes benefits from existing IVOA notations and standards like UCD, VOUnits, VO protocols and service design; and it is planned for a full integration into the VO landscape.

Fig. 2 shows the dependencies of this document with respect to other existing standards.

1.4 Previous efforts

The provenance concept was early introduced by the IVOA within the scope of the Observation Data Model (see IVOA note by IVOA Data Model Working Group, 2005), as a class describing where the data is coming from. A full observation data model specifically dedicated to spectral data was then designed (Spectral Data Model, McDowell and Salgado et al., 2016), as well as a fully generic characterisation data model of the measurement

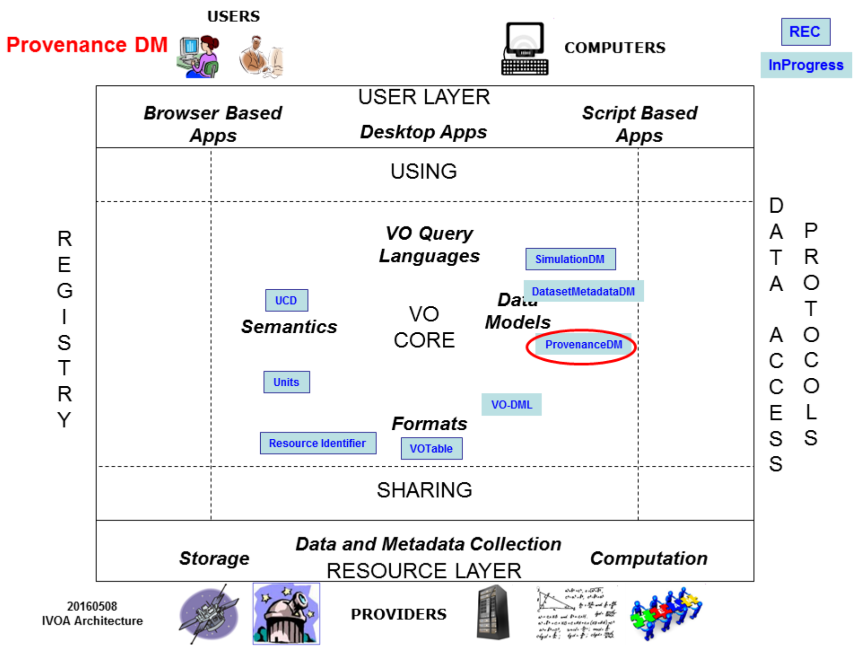

Figure 2: Architecture diagram for the Provenance Data Model. It is based on existing concepts defined in existing IVOA data models, and existing formats and semantics and fully integrated in the IVOA framework

axes of data (Characterisation Data Model, IVOA Data Model Working Group, 2008), while the progress on the provenance data model was slowing down.

The IVOA Data Model Working Group first gathered various use cases coming from different communities of observational astronomy (optical, radio, X-ray, interferometry). Common motivations for a provenance tracing of their history included: quality assessment, discovery of dataset progenitors, and access to metadata necessary for reprocessing. The provenance data model was then designed as the combination of *Data processing*, *Observing configuration*, and *Observation ambient conditions* data model classes. The *Processing class* was embedding a sequence of processing stages which were hooking specific ad hoc details and links to input and output datasets, as well as processing step descriptions. Despite the attempts at an UML description of the model and writing XML serialization examples, the IVOA efforts failed to provide a workable solution: the scope was probably too ambitious and the technical background too unstable. A compilation of these early developments can be found on the IVOA site (Bonnarel and the IVOA Data Model Working Group, 2016). From 2013 onwards, the IVOA concentrated on use cases related to processing description and decided to design the model by extending the basic W3C provenance structure, as described in the current specification.

Outside of the astronomical community, the Provenance Challenge series (2006 – 2010), a community effort to achieve inter-operability between different representations of provenance in scientific workflows, resulted in the Open Provenance Model (OPM) (Moreau and Clifford et al., 2010). Later, the W3C Provenance Working Group was founded and released the W3C Provenance Data Model as Recommendation in 2013 (Belhajjame and B'Far et al., 2013). OPM was designed to be applicable to anything, scientific data as well as cars or immaterial things like decisions. With the W3C model, this becomes more focused on the web. Nevertheless, the core concepts are still in principle the same in both models and are very general, so they can be applied to astronomical datasets and workflows as well. The W3C model was taken up by a larger number of applications and tools than OPM, we are therefore basing our modeling efforts on the W3C Provenance Data Model, making it less abstract and more specific, or extending it where necessary.

The W3C model even already specifies PROV-DM Extensibility Points (section 6 in Belhajjame and B'Far et al. 2013) for extending the core model. This allows one to specify additional roles and types for each entity, agent or relation using the attributes `prov:type` and `prov:role`. By specifying well-defined values for the IVOA model, we can adjust the model to our needs while still being compliant with W3C.

2 The provenance data model

In this section, we describe the currently discussed Provenance Data Model. We start with an UML class diagram, explain the core elements, and then, in the following sections, we give more details for each class and relation. An auto-generated documentation of all classes (VO-DML compliant) is available in the Volute repository at `https://volute.g-vo.org/svn/trunk/projects/dm/provenance/vo-dml/ProvenanceDM.html`.

2.1 Overview: Conceptional UML class diagram and introduction to core classes

Figure 3 shows the conceptional UML diagram for an IVOA Provenance Data Model. The core elements of the Provenance Data Model are *Entity*, *Activity* and *Agent*. For these elements we chose the same names that were used in the Provenance Data Model of the World Wide Web Consortium (W3C, Belhajjame and B'Far et al. 2013), which defines a very abstract pattern that can be reused here. Here are the core classes with short descriptions and some examples:

- *Entity:* a thing in a certain state

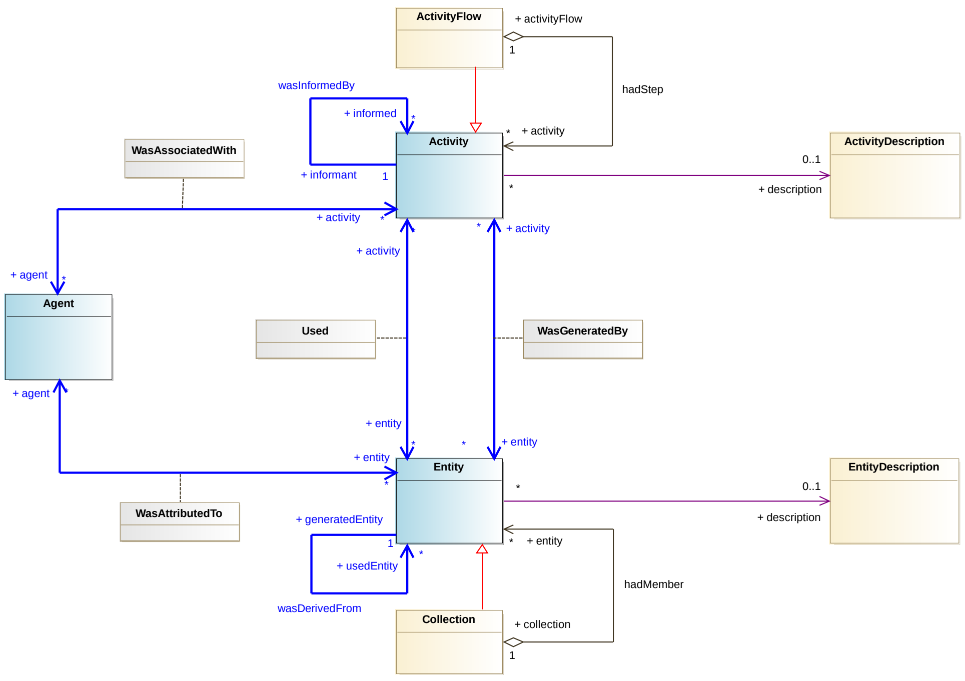

Figure 3: Overview of the classes for the Provenance Data Model in a conceptual class diagram. The blue classes are core elements. There are a number of many-to-many relationships with attached association classes (grey) that may contain additional attributes.

examples: data products such as images, catalogs, parameter files, calibration data, instrument characteristics

- *Activity:* an action/process or a series of actions, occurring over a period of time, performed on or caused by entities, usually resulting in new entities
  examples: data acquisition like observation, simulation; regridding, fusion, calibration steps, reconstruction

- *Agent:* executes/controls an activity, is responsible for an activity or an entity
  examples: telescope astronomer, pipeline operator, principal investigator, software engineer, project helpdesk

These core classes, along with their relations to each other, are displayed in Figure 4. We use the following relation classes to specify the mapping between the three core classes. The relation names were, again, chosen to match the W3C model names:

- *WasGeneratedBy:* a new entity is generated by an activity
  (entity “image m31.fits” wasGeneratedBy activity “observation”)

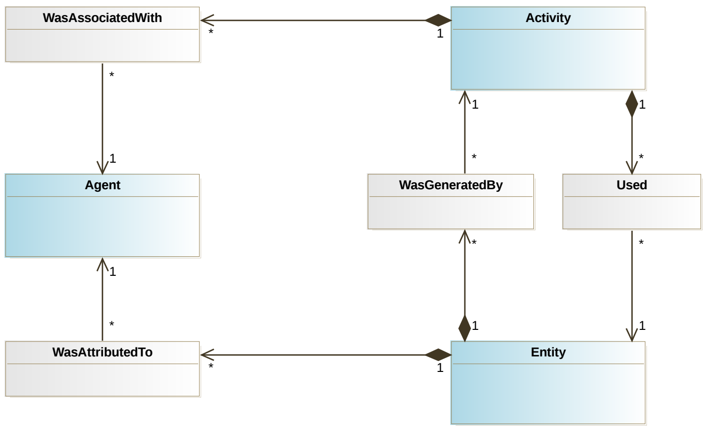

Figure 4: The main core classes and relations of the Provenance Data Model, which also occur in the W3C model.

- *Used:* an entity is used by an activity
  (activity "calibration" used entities "calibration data", "raw images")
- *WasAssociatedWith:* agents have responsibility for an activity
  (agent "observer Max Smith" wasAssociatedWith activity "observation")
- *WasAttributedTo:* an entity can be attributed to an agent
  (entity "image m31.fits" wasAttributedTo "M31 observation campaign")

Note that the relations appear as extra classes (and thus boxes in the diagrams, instead of just having annotated relations), because they can have additional attributes – when mapping the model to a relational database, these relations would appear as mapping tables.

In the domain of astronomy, certain processes and steps are repeated over and over again, using different parameters. We therefore separate the descriptions of activities from the actual processes and introduce an additional *ActivityDescription* class (see Figure 3). Likewise, we also apply the same pattern for *Entity* and add an *EntityDescription* class. Defining such descriptions allows them to be reused, which is very useful when performing a series of tasks of the same type, as is typically done in astronomy.

A similar normalization of descriptions of the actual processes and datasets can be found in the IVOA Simulation Data Model (SimDM, Lemson and Wozniak et al., 2012)), which describes simulation metadata. The SimDM classes *Experiment* and *Protocol* correspond to the Provenance terms *Activity* and *ActivityDescription*.

This separation into two classes may not be needed for each and every project, and everyone is free to choose which classes make sense for his/her use case. Also, when serializing provenance, one can integrate the description side into the other classes. More details about all these classes and relations are given in the following section.

2.2 Model description

2.2.1 Class diagram and VO-DML compatibility

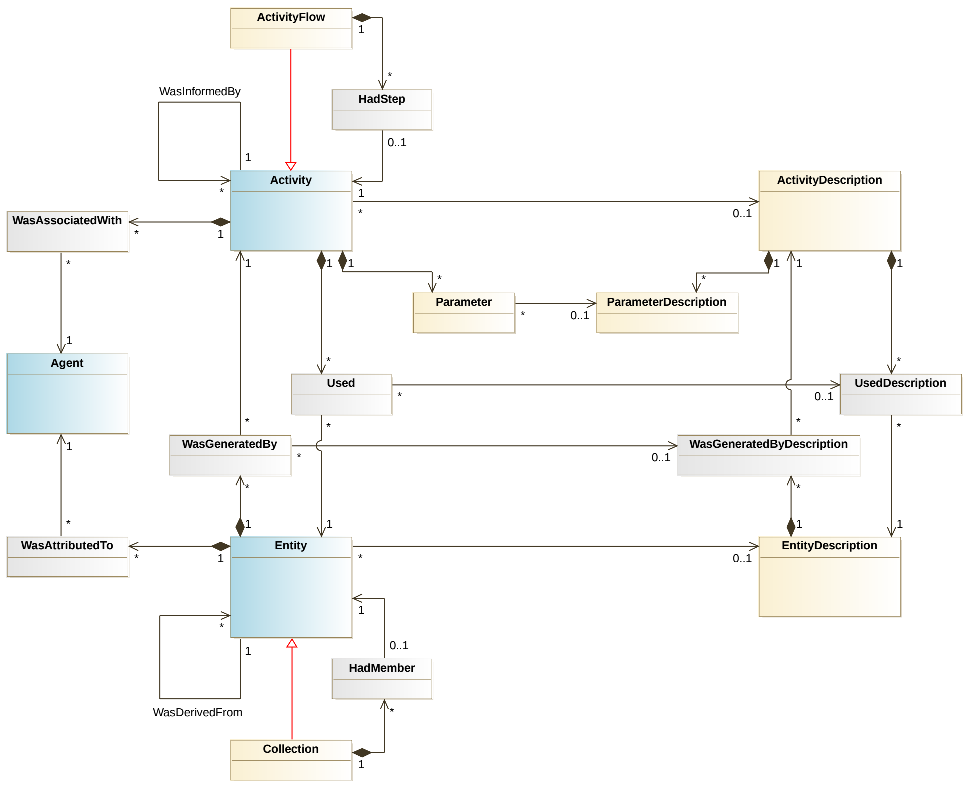

Figure 5: More detailed overview of the classes for the Provenance Data Model. Note that this UML class diagram is compatible with VO-DML.

Figure 5 shows the full class diagram, with the association classes for the many-to-many relations modeled more directly as mapping classes. When implementing the model in a relational database, these classes can be represented as individual tables for mapping the relation. We model one of each of the associations of the many-to-many relationships as a composition (full diamond) if the mapping class belongs more strongly to one of its linked classes; e.g. the *Used* relations are strongly dependent on the corresponding *Activities*. The documentation of all classes and an automatically generated figure based on the underlying xmi-description behind this UML diagram is available in the Volute repository at `https://volute.g-vo.org/svn/trunk/projects/dm/provenance/vo-dml/ProvenanceDM.html`.

This version of the UML diagram is fully VO-DML compliant, i.e. we just used the restricted subset of UML to model Provenance and reused the IVOA datatypes.

Entity

Attribute	W3C ProvDM	Data type	Description
id	prov:id	(qualified) string	a unique id for this entity (unique in its realm)
name	prov:label	string	a human-readable name for the entity (to be displayed by clients)
location	prov:location	string	a path or a geographical location
type	prov:type	string	a provenance type, i.e. one of: prov:collection, prov:bundle, prov:plan, prov:entity; not needed for a simple entity
annotation	prov:description	string	text describing the entity in more detail
value	prov:value		provides a value that is a direct representation of an entity
rights	–	string	access rights for the data, values: public, secure or proprietary; see Curation.Rights, RightsType in DatasetDM
creationTime	–	datetime	date and time at which the entity was created (e.g. timestamp of a file)
→ description		link	link to *EntityDescription*
Additional attributes:			
Further project-specific attributes (e.g. size, path, url, . . .) can be added (see Section B.1)			

Table 1: Attributes of *Entities*. Mandatory attributes are marked in **bold**, references are indicated with an arrow (→). Attributes from *EntityDescription* (see next section) may appear here as well.

2.2.2 Entity and EntityDescription

Entities in astronomy are usually astronomical or astrophysical datasets in the form of images, tables, numbers, etc. But they can also be observation or simulation log files, files containing system information, environment variables, names and versions of packages, ambient conditions, or, in a wider sense, also observation proposals, scientific articles, or manuals and other documents.

An entity is not restricted to being a file. It can even be just a number in a table, depending on how fine-grained the provenance shall be described. An entity can also carry its value directly in its value attribute.

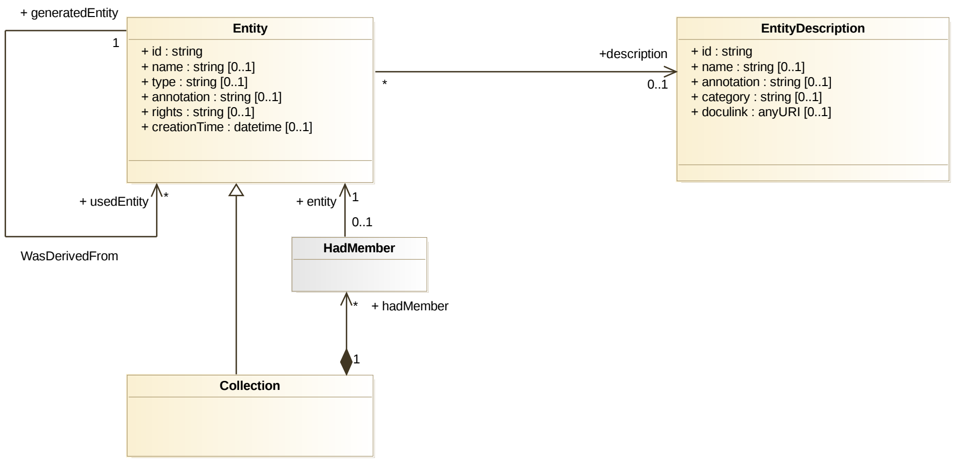

Figure 6: The relations between Entity, EntityDescription and Collection (see Section 2.2.3). Links to the Dataset class from the Dataset Metadata Model are described in Section B.

The VO concept closest to *Entity* is the notion of *Dataset*, which could mean a single table, an image or a collection of them. The Dataset Metadata Model (Bonnarel and Laurino et al., 2015) specifies a *Dataset* as “a file or files which are considered to be a single deliverable”. Most attributes of the *Dataset* class can be mapped directly to attributes of the *Entity* and *EntityDescription* class, see the mappings of Table 17 in Section B.

Entities have the attributes given in Table 1. If an attribute also exists in the W3C Provenance Data Model, we list its name in the second column.

The difference between entities that are used as input data and those used as output data becomes clear by specifying the relations between the data and the activities producing or using these data. More details on this will follow in Section 2.2.6.

EntityDescription. The types of entities, or datasets in astronomy, can be predefined using a description class *EntityDescription*. This class is meant to store descriptive information about an entity that is known before an *Entity* instance is created. For example, if we run an activity to create an RGB image from three greyscale images, we may know a mandatory `format` for the input and output images before the activity's execution (JPG, PNG,

EntityDescription

Attribute	Data type	Description
id	(qualified) string	a unique identifier for this description
name	string	a human-readable name for the entity description
annotation	string	a decriptive text for this kind of entity
category	string	specifies if the entity contains information on logging, system (environment), calibration, simulation, observation, configuration, ...
doculink	url	link to more documentation

Additional attributes:
Further project-specific attributes (e.g. format, content_type) and/or attributes from other data models (e.g. dataproduct_type and -_subtype, version, calibLevel from DatasetDM) can be added (Section B.1).

Table 2: Attributes of *EntityDescription*. For simple use cases, this description class may be ignored and its attributes may be used for *Entity* instead.

FITS, ...), but we probably cannot know the final `size` of the image that will be created. In this example, `format` would be an *EntityDescription* attribute, while `size` would be an attribute of the *Entity* instance.

Additional attributes that describe the content of the data could be derived from the Dataset Metadata Model (see Section B.1)

The *EntityDescription* class does *not* contain any information about the usage of the data, in particular, it tells nothing about them being used as input or as output. This kind of information is exclusively provided by the relations (and their relation descriptions) between activities and entities (see Section 2.2.6).

The *EntityDescription* general attributes are summarized in Table 2.

WasDerivedFrom. In Figure 6 there is one more relation that we have not mentioned yet: the *WasDerivedFrom* relation, linking two entities together, which is borrowed from the W3C model. It is used to express that one entity was derived from another, i.e. it can be used to find one (or more) progenitor(s) of a dataset, without having to look for the activities in between. It can therefore serve as a shortcut.

The information this relation provides is somewhat redundant, since progenitors for entities can be found through the links to activity and the corresponding descriptions. Nevertheless, we include *WasDerivedFrom* for those

WasDerivedFrom

Attribute	Data type	Description
→ **generatedEntity**	link	link to the *Entity*
→ **usedEntity**	link	link to the progenitor *Entity*, from which the generatedEntity was derived

Table 3: Attributes of the *WasDerivedFrom* relation. These are the same as those used in W3C's ProvDM. **Mandatory** attributes are marked in bold, references in the data model are indicated with an arrow (→). The W3C model contains additional optional links to the related *Activity*, *WasGeneratedBy* and *Used* relations, which we do not include here for simplicity.

cases where an explicit link between an entity and its progenitor is useful (e.g. for speeding up searches for progenitors, or if the activity in between is not important).

Note that the *WasDerivedFrom* relation cannot always automatically be inferred from following *WasGeneratedBy* and *Used* relations alone: If there is more than one input and more than one output of an activity, it is not clear (without consulting the activityDescription and entity roles in the relation descriptions) which entity was derived from which. Only by specifying the descriptions and roles accordingly, or by adding a *WasDerivedFrom* relation, this direct derivation becomes known.

2.2.3 Collection

Collections are entities that are grouped together and can be treated as one single entity. From the provenance point of view, they have to have the *same origin*, i.e., they were produced by the same activity (which could also be the activity of collecting data for a publication or similar). The term "collection" is also used in the Dataset Metadata Model for grouping datasets. As an example, a collection with the name 'RAVE survey' could consist of a number of database tables and spectra files.

The *Entity-Collection* relation can be modeled using the *Composite* design pattern: *Collection* is a subclass of *Entity*, but also an aggregation of 1 to many entities, which could be collections themselves. In order to be compliant to VO-DML, we model the membership relation explicitly by including a *HadMember* class in our model, which is connected to the *Collection* class via a composition. It may contain an additional role attribute.

Collections are also known in the W3C model, in the same sense as used here. The relation between entity and collection is also called *HadMember* in the W3C model.

An additional class *CollectionDescription* is only needed if it has different attributes than the *EntityDescription*. This class should therefore only be introduced if a use case requires it.

Advantages of collections: Collections can be used to collect entities with the same provenance information together, in order to hide complexity where necessary. They can be used for defining different levels of detail (granularity).

2.2.4 Activity and ActivityDescription

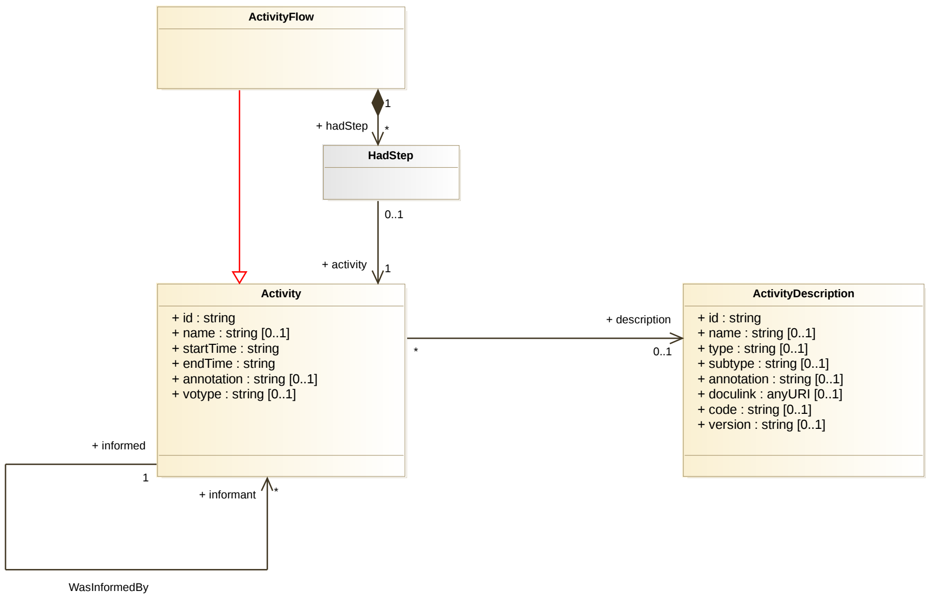

Figure 7: Details for Activity, ActivityDescription and ActivityFlow (Sections 2.2.4 and 2.2.5).

Activities in astronomy include all steps from obtaining data to the reduction of images and production of new datasets, such as image calibration, bias subtraction, image stacking, light curve generation from a number of observations, radial velocity determination from spectra, post-processing steps of simulations, etc.

ActivityDescription. The method underlying an activity can be specified by a corresponding *ActivityDescription* class (previously named *Method*, corresponds to the *Protocol* class in SimDM). This could be, for instance, the name of the code used to perform an activity or a more general description of the underlying algorithm or process. An activity is then a concrete case

Activity

Attribute	W3C ProvDM	Data type	Description
id	prov:id	(qualified) string	a unique id for this activity (unique in its realm)
name	prov:label	string	a human-readable name (to be displayed by clients)
startTime	prov:startTime	datetime	start of an activity
endTime	prov:endTime	datetime	end of an activity
annotation	prov:description	string	additional explanations for the specific activity instance
votype		string	can be either "activity" or "activityFlow"
→ description		link	link to *ActivityDescription*

Table 4: Attributes of *Activity*, their data types and equivalents in the W3C Provenance Data Model, if existing. Attributes in bold are **mandatory**, references are indicated with an arrow (→). If no *ActivityDescription* class is used, those attributes can be used here as well.

(instance) of using such a method, with a startTime and an endTime, and it refers to a corresponding description for further information.

There MUST be exactly zero or one *ActivityDescription* per *Activity*. If steps from a pipeline shall be grouped together, one needs to create a proper *ActivityDescription* for describing all the steps at once. This method can then be referred to by the pipeline activity.

When serializing the data model, the attributes of the description class may be assigned to the activity for simplification.

WasInformedBy. The individual steps of a pipeline can be chained together directly, without mentioning the intermediate datasets, using the *WasInformedBy* relation. This relation can be used as a shortcut if the skipped datasets are deemed to be not important enough to be recorded. For grouping activities, also see the next Section 2.2.5.

2.2.5 ActivityFlow

For facilitating grouping of activities (and their related entities, etc.) we introduce the class *ActivityFlow*. It can be used for hiding and grouping a part of the workflow/pipeline or provenance description, if different levels of granularity are required. Such pipelines and workflows are very common in astronomical data production and processing. Figure 8 illustrates an

ActivityDescription

Attribute	Data type	Description
id	string	a unique id for this activity description (unique in its realm)
name	string	a human-readable name (to be displayed by clients)
type	string	type of the activity, from a vocabulary or list, e.g. data acquisition (observation or simulation), reduction, calibration, publication
subtype	string	more specific subtype of the activity
annotation	string	additional free text description for the activity
code	string	the code (software) used for this process, if applicable
version	string	a version number, if applicable (e.g. for the code)
doculink	url	link to further documentation on this process, e.g. a paper, the source code in a version control system etc.

Table 5: Attributes of *ActivityDescription*.

WasInformedBy

Attribute	Data type	Description
→ **informed**	link	link to the *Activity* being informed by another ("second" activity)
→ **informant**	link	link to the informing *Activity* ("first" activity)

Table 6: Attributes of the *WasInformedBy* relation. We just use this class to link chained activities together. The attribute names correspond to the W3C PROV-DM names.

example provenance graph at a detailed level (left side), and also using an *ActivityFlow* (right side).

We also explored the different ways to describe a set of activities within the W3C provenance model. This model uses *Bundle*, i.e. an entity with type "Bundle", for wrapping a provenance description. Each part of a provenance description can be put into a bundle, and the bundle can then be reused in other provenance descriptions. W3C's *Plan* is an entity with type "Plan" and is used for describing a set of actions or steps. Both, *Bundle*

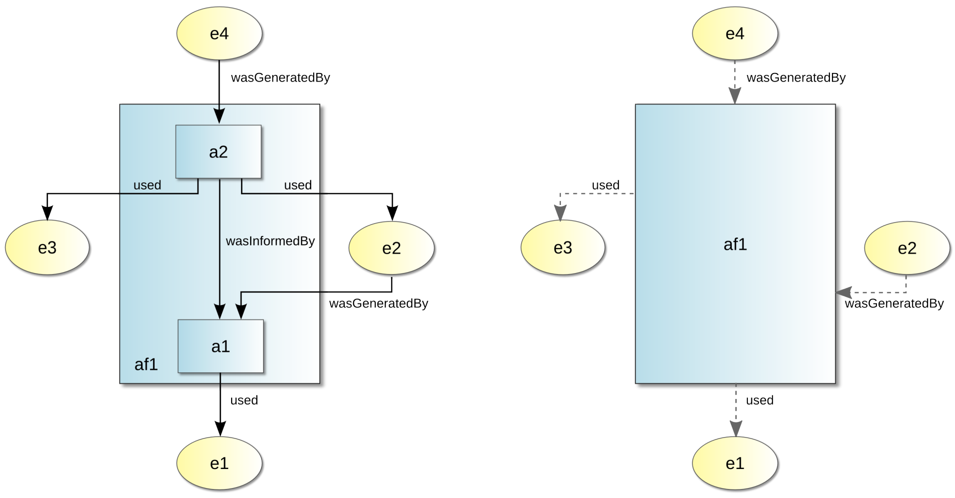

Figure 8: An example provenance graph. The detailed version is shown on the left side. It also shows the shortcut *WasInformedBy* to connect two activities, which could be used if the entity e2 would not be needed anywhere else. An ActivityFlow can be used to "hide" a part of the provenance graph as is shown on the right side. Activities are marked by blue rectangles, entities by yellow ellipses.

and *Plan*, are entities and have the attributes and relations of this class (and thus one can define provenance of bundles and plans as well).

But we would like to consider a set of activities as being an *Activity* itself, with all the relations and properties that characterize activities. Therefore, we do not reuse W3C's classes for describing workflows and plans, but add the class *ActivityFlow* as an activity composed of other activities. The composition is represented by the *HadStep* relation, as is shown in Figure 7. In implementations, *ActivityFlow* can either be represented as an extra class or as *Activity* with attribute `votype="ActivityFlow"`.

2.2.6 Entity-Activity relations

For each data flow it should be possible to clearly identify entities and activities. Each entity is usually a result from an activity, expressed by a link from the entity to its generating activity using the *WasGeneratedBy* relation, and can be used as input for (many) other activities, expressed by the *Used* relation. Thus the information on whether data is used as input or was produced as output of some activity is given by the *relation-types* between activities and entities.

We use two relations, *Used* and *WasGeneratedBy* (see Tables 7 and 8), instead of just one mapping class with a flag for input/output, because their descriptions and role-attributes can be different.

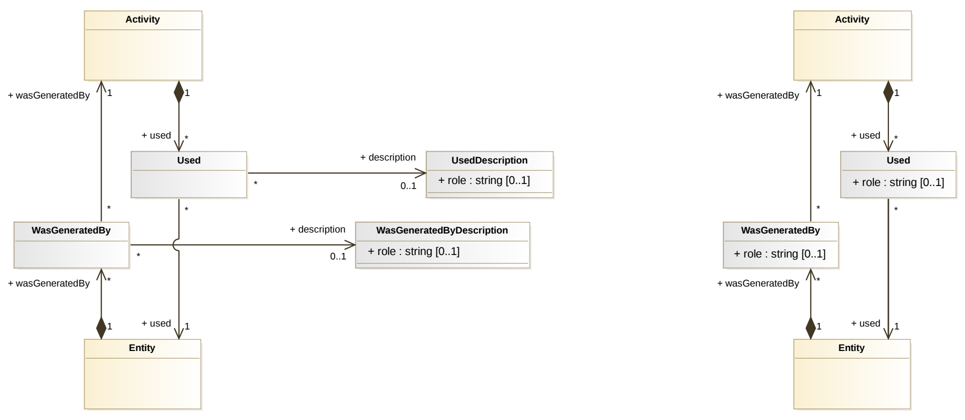

Figure 9: Entity and *Activity* are linked via the *Used* and *WasGeneratedBy* relations. In the left image, the `role` that an entity played when being used or generated by an activity is recorded with the corresponding *UsedDescription* and *WasGeneratedByDescription*, also see Section 2.2.6. If these description classes are not used, the *role* can be used directly as an attribute within the *Used* and *WasGeneratedBy* classes (right image).

Used

Attribute	**W3C ProvDM**	**Data type**	**Description**
id	prov:id	string	an identifier for this relation
role	prov:role	string	role of the entity, defines as what it is being used
time	prov:time	datetime	time at which the usage of an entity started
→ **activity**	prov:activity	link	link to an *Activity*
→ entity	prov:entity	link	link to an *Entity*
→ description		link	link to the corresponding *UsedDescription*, if existing

Table 7: Attributes and references of *Used* relation class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→). The `role` attribute can also be defined in the *UsedDescription* class instead.

The *Used* and *WasGeneratedBy*-relation can have the optional attribute `time` – this is the time when usage of an entity started or the generation of an entity is finished.

WasGeneratedBy

Attribute	**W3C ProvDM**	**Data type**	**Description**
id	prov:id	string	an identifier for this relation
role	prov:role	string	role of the entity that is generated by an activity, defines which output type it is
time	prov:time	datetime	time at which the generation of an entity is finished
→ **entity**	prov:entity	link	link to an *Entity*
→ activity	prov:activity	link	link to an *Activity*
→ description		link	link to the corresponding *WasGeneratedByDescription*, if existing

Table 8: Attributes and references of *WasGeneratedBy* relation class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→). The `role` attribute can also be defined in the *WasGeneratedByDescription* class instead.

Compositions and multiplicities In principle, an entity is produced by just one activity. However, by introducing the *ActivityFlow* class for grouping activities together, one entity can now have many wasGeneratedBy-links to activities. One of them must be the actual generation activity, the other activities can only be activityFlows containing this generation-activity. This restriction of having only one "true" generation activity is not explicitly expressed in the current model[2].

The *Used* relation is closely coupled to the *Activity*, so we use a composition here, indicated in Figure 5 by a filled diamond: if an activity is deleted, then the corresponding used relations need to be removed as well. The entities that were used still remain, since they may have been used for other activities as well. We need a multiplicity * between *Used* and *Entity*, because an entity can be used more than once (by different activities).

Similarly, the *WasGeneratedBy* relation is closely coupled with the *Entity* via a composition, since a wasGeneratedBy relation makes no sense without its entity. So if an entity is deleted, then its wasGeneratedBy relation must be deleted as well. There is a multiplicity * between *Activity* and *WasGeneratedBy*, because an activity can generate many entities.

[2]The reason for this is that we want to keep the model simple and avoid introducing even more classes.

UsedDescription

Attribute	Data type	Description
id	string	identifier
role	string	entity role; defines the role of an entityDescription, as what it is used for the linked type of activityDescription
→ **activityDescription**	link	link to *ActivityDescription*
→ entityDescription	link	link to *EntityDescription*

Table 9: Attributes and references of *UsedDescription* class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→).

WasGeneratedByDescription

Attribute	Data type	Description
id	string	identifier
role	string	entity role; defines the role of an entityDescription, which kind of output it is
→ **entityDescription**	link	link to *EntityDescription*
→ activityDescription	link	link to an *ActivityDescription*

Table 10: Attributes and references of *WasGeneratedByDescription* class. Attributes/references in bold are **mandatory**, references to other classes are indicated with an arrow (→).

Entity roles Each activity requires specific roles for each input or output entity, thus we store this information with description classes, in the role-attributes for the *UsedDescription* and *WasGeneratedByDescription* relation (see Tables 9 and 10). For example, an activity for darkframe subtraction requires two input images. But it is very important to know which of the images is the raw image and which one fulfils the role of dark frame.

The role is in general NOT an attribute for *EntityDescription* or *Entity*, since the same entity (e.g. a specific FITS file containing an image) may play different roles with different activities. If this is not the case, if the image can only play the same role everywhere, only then it is an intrinsic property of the entity and should be stored in the *EntityDescription*.

Some example roles are given in Table 11. Note that these roles don't have to be unique, many datasets may play the same role for a process. For example, many image entities may be used as science-ready-images for an image stacking process.

Role	**Example entities**
configuration	configuration file
auxiliary input	calibration image, dark frame, etc.
main input	raw image, science-ready images
main result	image, cube or spectrum
log	logging output file
red	image used for red channel of a composite activity

Table 11: Examples for entity roles as attributes in the *UsedDescription* and *WasGeneratedByDescription*.

In order to facilitate interoperability, the possible entity-roles could be defined and described for each activity by the IVOA community, in a vocabulary list or thesaurus.

2.2.7 Parameters

The concept of activity configuration, generally a set of parameters that can be configured, is tightly linked to the concept of provenance, even though it is not core provenance information.

Activity configuration typically implies a set of parameters attached to the activity before execution. The concept of parameter is very general, and already exists in the VO context, for example as PARAM elements in VOTable (Ochsenbein and Williams et al., 2013), as uws:param elements in the UWS pattern (Harrison and Rixon, 2010), or as POST/GET parameters for web services (Bonnarel and Demleitner et al., 2017, see for example). The Simulation DM also contains a ParameterSetting class (Lemson and Wozniak et al., 2012).

In order to provide a link between configuration and provenance information, we add a *Parameter* class along with a *ParameterDescription*, so that configuration information is structured and stored with the provenance information (and can thus be queried simultaneously).

The *Parameter* class is directly connected to an activity without complex *Entity-Activity* relations. The optional attributes of *ParameterDescription* are taken from the FIELD and PARAM elements in VOTable (Ochsenbein and Williams et al., 2013).

We note that a parameter can be seen as a simplified entity, though it can only be used by an activity and it has no provenance. In some cases, it

Parameter

Attribute	Data type	Description
id	string	parameter unique identifier
value	(value dependent)	the value of the parameter, type depends on ParameterDescription.datatype and xtype; follows same rules as VOTable TABLEDATA and DALI
→ activity	link	link to *Activity*
→ description	link	link to *ParameterDescription*

Table 12: Attributes of *Parameter*. Attributes in bold are **mandatory**, references to other classes are indicated with an arrow (→). Attributes of *ParameterDescription* can be added here as well, if that class is not used.

may be relevant to define a specific parameter as an entity if its provenance should be traced.

For example, in the case of a processing activity that cleans an image with a sigma-clipping method, the input and output images would be entities and the value of the number of sigma for sigma-clipping could be a parameter instead of an entity. We may also want to define a 3-sigma-clipping activity where this parameter is fixed to 3.

2.2.8 Agent

An *Agent* describes someone who is responsible for a certain task or entity, e.g. who pressed a button, ran a script, performed the observation or published a dataset. The agent can be a single person, a group of persons (e.g. MUSE WISE Team), a project (CTA) or an institute. This is also reflected in the IVOA Dataset Metadata Model, where *Party* represents an agent, and it has two types: *Individual* and *Organization*, which are explained in more detail in Table 14 (also see Section B for comparison between *Agent* and *Party*). Both agent types are also used in the W3C Provenance Data Model, though *Individual* is called *Person* there. We decided to not include the type *SoftwareAgent* from W3C (yet), since it is not required for our current use cases. This may change in the future.

A definition of organizations is given in the IVOA Recommendation on Resource Metadata (Hanisch and the IVOA Resource Registry Working Group et al., 2007), hereafter referred to as RM: "An organisation is [a] specific type of resource that brings people together to pursue participation in VO applications." It also specifies further that scientific projects can be considered as organisations on a finer level: "At a high level, an organisation

ParameterDescription

Attribute	**Data type**	**Description**
id	string	parameter unique identifier
name	string	parameter name
annotation	string	additional free text description
datatype	string	datatype
arraysize	number	number of values of specified datatype, if there is more than one
unit	string	physical unit
ucd	string	Unified Content Descriptor, supplying a standardized classification of the physical quantity
utype	string	UType, meant to express the role of the parameter in the context of an external data model
xtype	string	extended datatype as in VOTable 1.2 and above
min	number	minimum value
max	number	maximum value
options	list	list of accepted values
→ activityDescription	link	link to *ActivityDescription*

Table 13: Attributes of *ParameterDescription*.

could be a university, observatory, or government agency. At a finer level, it could be a specific scientific project, space mission, or individual researcher. A provider is an organisation that makes data and/or services available to users over the network."

For each agent a `name` should be specified, a summary of the attributes for *Agent* is given in Table 15. We added the optional attributes `address` and `email`, since they appeared in our use cases and are quite commonly used. Not every project will need them; e.g. an advanced system may use permanent identifiers (e.g. ORCIDs) to identify agents and retrieve their properties from an external system instead. It would also increase the value of the given information if the (current) affiliation of the agent (and a project leader/group leader) were specified in order to maximize the chance of finding any contact person later on. The contact information is needed in case more information about a certain step in the past of a dataset is required, but also in order to know who was involved and to fulfill our "Attribution" requirement (Section 1.2), so that proper credits are given to the right people/projects.

AgentType

Class or type	W3C ProvDM	DatasetDM	Comment
Agent	Agent	Party	
Individual	Person	Individual	a person, specified by name, email, address, (though all these parts may change in time)
Organization	Organization	Organization	a publishing house, institute or scientific project

Table 14: Agent class and types of agents/subclasses in this data model, compared to W3C ProvDM and DatasetDM.

Agent

Attribute	W3C ProvDM	Data type	Description
id	prov:id	(qualified) string	unique identifier for an agent
name	prov:name	string	a common name for this agent; e.g. first name and last name; project name, agency name...
type	prov:type	string	type of the agent: either Individual (Person) or Organization
email		string	Contact email of the agent
address		string	Address of the agent

Table 15: Agent attributes

It is desired to have at least one agent given for each activity (and entity), but it is not enforced. There can also be more than one agent for each activity/entity with different `roles` and one agent can be responsible for more than one activity or entity. This many-to-many relationship is made explicit in our model by adding the two following relation classes:

- wasAssociatedWith: relates an *activity* to an agent
- wasAttributedTo: relates an *entity* to an agent

We adopted here the same naming scheme that was used in the W3C ProvDM. Note that the attributed-to-agent for a dataset may be different from the agent that is associated with the activity that created an entity. Someone who is performing a task is not necessarily given full attribution, especially if he acts on behalf of someone else (the project, university, ...).

AgentRoles

role	type or sub class	Comment
author	Individual	someone who wrote an article, software, proposal
contributor	Individual	someone who contributed to something (but not enough to gain authorship)
editor	Individual	editor of e.g. an article, before publishing
creator	Individual	someone who created a dataset, creators of articles or software are rather called "author"
curator	Individual	someone who checked and corrected a dataset before publishing
publisher	Organization	organization (publishing house, institute) that published something
observer	Individual	observer at the telescope
operator	Individual	someone performing a given task
coordinator	Individual	someone coordinating/leading a project
funder	Organization	agency or sponsor for a project as in Prov-N
provider	Organization	"an organization that makes data and/or services available to users over the network" (definition from RM)

Table 16: Examples for roles of agents and the typical type of that agent

In order to make it clearer what an agent is useful for, we suggest the possible roles an agent can have (along with descriptions partially taken from RM) in Table 16. For comparison, SimDM contains following roles for their contacts: owner, creator, publisher and contributor. Note that the *Party* class in Dataset and SimDM are very similar to the *Agent* class, which is explained in more detail in Section B.

This list is *not* complete. We consider providing a vocabulary list for this in a future version of this model, collected from (future) implementations of this model.

3 Serialization of the provenance data model

3.1 Introduction

Serialization files constitute the building blocks of the client/server and client/client dialogs. The provenance information as represented in the data model is split in three main concepts that can be searched following many different relations between the main 3 classes, *Activity*, *Entity* and *Agent*. The selection of the relations to expose when distributing the provenance information depends on the usage and will be described more extensively in the Implementation Note (Riebe and Servillat et al., 2017) and the links therein.

To give a very simple example, suppose a client asks for the context of execution for one specified activity, which computes a simple RGB color composition. On the server side, exposing the provenance information for this activity or for an entity, corresponding to a monocolor or RGB image, means to just expose the structure of the classes and relation tables and feed them with the related tuples in the database. On the client side, the content of a VO-Provenance serialization document can then be explored and represented using graphical interfaces, as inspired by the Provenance Southampton suite or by customized visualization tools.

3.2 Serialization formats: PROV-N, PROV-JSON and PROV-XML

The serialization formats PROV-N, PROV-JSON and PROV-XML are proposed in the W3C Provenance framework for storing and exchanging the provenance metadata: PROV-N, PROV-JSON and PROV-XML, defined in Moreau and Missier (2013), Huynh and Jewell et al. (2013) and Hua and Tilmes et al. (2013), respectively. They can be reused here as well for serializations of our data model. For producing fully W3C compatible serializations, see Section 3.5.

Here is an example serialization instance document for an entity being processed by an activity, in PROV-N notation:

```
document
  prefix ivo <http://www.ivoa.net/documents/rer/ivo/>
  prefix ex <http://www.example.com/provenance/>
  prefix voprov <http://www.ivoa.net/documents/dm/provdm/voprov/>
  entity(ivo://example#Public_NGC6946, [voprov:name="Processed image of NGC 6946"])
  entity(ivo://example#DSS2.143, [voprov:name="Unprocessed image of NGC 6946"])
  activity(ex:Process1, 2017-04-18T17:28:00, 2017-04-19T17:29:00, [voprov:name="Process 1"])
  used(ex:Process1, ivo://example#DSS2.143, -)
  wasGeneratedBy(ivo://example#Public_NGC6946, ex:Process1, 2017-05-05T00:00:00)
endDocument
```

Here is the same example in PROV-JSON format:

```
{
  "prefix": {
    "ivo": "http://www.ivoa.net/documents/rer/ivo/",
    "voprov": "http://www.ivoa.net/documents/dm/provdm/voprov/",
    "ex": "http://www.example.com/provenance/"
  },
  "activity": {
    "ex:Process1": {
      "voprov:startTime": "2017-04-18T17:28:00",
      "voprov:endTime": "2017-04-19T17:29:00",
      "voprov:name": "Process 1"
    }
  },
  "wasGeneratedBy": {
    "_:id4": {
      "voprov:time": "2017-05-05T00:00:00",
      "voprov:entity": "ivo://example#Public_NGC6946",
      "voprov:activity": "ex:Process1"
    }
  },
  "used": {
    "_:id1": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143",
      "voprov:activity": "hips:AlaRGB1"
    }
  }
  "entity": {
    "ivo://example#DSS2.143": {
      "voprov:name": "Unprocessed image of NGC6946"
    },
    "ivo://example#Public_NGC6946": {
      "voprov:name": "Processed image of NGC 6946"
    }
  }
}
```

PROV-JSON, PROV-N and PROV-XML can be converted into each other, e.g. using the prov or voprov python package (see Section "voprov" in Implementation Note (Riebe and Servillat et al., 2017)).

3.3 VOTable format for Provenance metadata

TODO:
Move this section to ProvTAP document?

To emphasize the compatibility to the IVOA framework, where the VOTable-XML format is a reference to circulate metadata, we define a VOTable mapping specification. All classes' declarations and relations described for this data model are translated into separated tables, one for each class of the model. All attributes of these classes are translated to columns, i.e. VOTable FIELDS. In addition, the specification defines the VOTable values of the FIELD and PARAM attributes `ucd`, `datatype`, `utype`, `unit`, `description`, etc.

This can be appropriately used for two goals:

- Publishing full provenance metadata for data collections in VOTable format. This can be produced by data processing workflows or as output of databases containing provenance metadata.
- Providing the backbone for the TAP schema describing IVOA provenance metadata which isused for ProvTAP

These VOTable serializations can be produced using the voprov python module, available to the community, see also Section "voprov" in Implementation Note (Riebe and Servillat et al., 2017).

Here is a VOTable document transcription of the serialization example given above in PROV-N and PROV-JSON:

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.2" xmlns="http://www.ivoa.net/xml/VOTable/v1.2"
  xmlns:ex="http://www.example.com/provenance"
  xmlns:ivo="http://www.ivoa.net/documents/rer/ivo/"
  xmlns:voprov="http://www.ivoa.net/documents/dm/provdm/voprov/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.2 http://www.ivoa.net/xml/VOTable/VOTable-1.2.xsd">
  <RESOURCE type="provenance">
    <DESCRIPTION>Provenance VOTable</DESCRIPTION>
    <TABLE name="Usage" utype="voprov:used">
      <FIELD arraysize="*" datatype="char" name="activity" ucd="meta.id" utype="voprov:Usage.activity"/>
      <FIELD arraysize="*" datatype="char" name="entity" ucd="meta.id" utype="voprov:Usage.entity"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ex:Process1</TD>
            <TD>ivo://example#DSS2.143</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Generation" utype="voprov:wasGeneratedBy">
      <FIELD arraysize="*" datatype="char" name="entity" ucd="meta.id" utype="voprov:Generation.entity"/>
      <FIELD arraysize="*" datatype="char" name="activity" ucd="meta.id" utype="voprov:Generation.activity"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://example#Public_NGC6946</TD>
            <TD>ex:Process1</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Activity" utype="voprov:Activity">
      <FIELD arraysize="*" datatype="char" name="id" ucd="meta.id" utype="voprov:Activity.id"/>
      <FIELD arraysize="*" datatype="char" name="name" ucd="meta.title" utype="voprov:Activity.name"/>
      <FIELD arraysize="*" datatype="char" name="start" ucd="" utype="voprov:Activity.startTime"/>
      <FIELD arraysize="*" datatype="char" name="stop" ucd="" utype="voprov:Activity.endTime"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ex:Process1</TD>
            <TD>Process 1</TD>
            <TD>2017-04-18 17:28:00</TD>
            <TD>2017-04-19 17:29:00</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Entity" utype="voprov:Entity">
      <FIELD arraysize="*" datatype="char" name="id" ucd="meta.id" utype="voprov:Entity.id"/>
      <FIELD arraysize="*" datatype="char" name="name" ucd="meta.title" utype="voprov:Entity.name"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://example#DSS2.143</TD>
            <TD>Unprocessed image of NGC6946</TD>
          </TR>
          <TR>
            <TD>ivo://example#Public_NGC6946</TD>
            <TD>Processed image of NGC 6946</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <INFO name="QUERY_STATUS" value="OK"/>
  </RESOURCE>
</VOTABLE>
```

This VOTable serialization can be considered as a flat view on the various tables stored in a database implementing the datamodel structure explained in Section 2. More examples of serialization documents are provided in Appendix A.

Such serializations can be retrieved through access protocols (see separate document(s))or directly integrated in dataset headers or “associated metadata” in order to provide provenance metadata for these datasets. E.g. for FITS files, a provenance extension called “PROVENANCE” could be added which contains provenance information of the workflow that generated the FITS file. This information could be stored directly using one of the serialization formats, for example as a unique cell in an ASCII TABLE extension.

TODO:
Add reference(s) to access protocols!

3.4 Serialization of description classes for web services

Description classes in ProvenanceDM gather information on the activity which can preexist to the activity itself. First, the *ActivityDescription* class gives generic information on the activity (`name`, `description`, `doculink`...) and the parameters expected as an input. In addition, *UsedDescription* and *WasGeneratedByDescription* classes indicate the expected roles of the input and output entities respectively. Finally, the activity may expect specific kinds of entities as inputs or outputs, for which there may be detailed descriptions stored as *EntityDescription* records.

The serialization of an ActivityDescription, that includes all those description classes, is based on the IVOA DataLink Service Descriptors for service resources (Dowler and Bonnarel et al., 2015), and can thus be stored as a VOTable (Ochsenbein and Williams et al., 2013). Indeed, a service descriptor points to a service that may execute an activity using input parameters, some of which probably point to entities. One may thus easily translate an ActivityDescription VOTable to a DataLink service descriptor VOTable block, and vice-versa.

The VOTable contains one resource with attributes type="meta" and utype="voprov:ActivityDescription". This resource contains PARAM elements to describe the activity and then GROUP elements gathering additional PARAM elements to describe:

- the input parameters (group name="InputParams"), which is similar to the group defining input parameters in a DataLink service descriptor,
- the input entities (group name="Used"),
- the output entities (group name="Generated").

The standard PARAM elements for an activity resource correspond to the attributes of the *ActivityDescription* class (see Section 2.2.4) and may include an Agent name and email. Only one agent can be given, corresponding to the main contact for this activity ("contact_name" and "contact_email"). The `utype` attribute is used to connect the PARAM element to its location in the ProvenanceDM, so that other optional elements can be added.

For the input parameters, each attribute located in the *ParameterDescription* class in the model (e.g. `units`, `ucd`, `utype`, `min`, ...) is mapped to an attribute of a PARAM element in the VOTable (both have the same structure, see Section 2.2.7). The `type` attribute of the PARAM element can be set to "no_query" to indicate that the parameter is optional for the activity (i.e. a default value will be used).

For the input and output entity groups, each entity is described with a GROUP block with the name attribute set to the EntityDescription identifier. It contains PARAM elements with e.g. the following names (all other attributes describing the entity are optional):

- name="default" for the default identifier of the entity (e.g. a file name) or its default value (if the entity is a value),
- name="role" that gives the role of the entity with respect to the Used or WasGeneratedBy relation (e.g. "red", "green" or "blue" channel image, or the output "RGB" file),
- name="content_type" for the MIME type expected by the activity for the input or output entity,

- other additional PARAM elements corresponding to attributes of *EntityDescription*, *UsedDescription*, *WasGeneratedByDescription* or possibly *Agent*.

It is possible to reference an input parameter using the `ref` attribute of PARAM, if an input or output entity is also referenced as an input parameter to the activity (e.g. the name of an input file, or an identifier). In the following example the output file name "RGB.jpg" is expected as a parameter to the activity and is thus referenced by the id PARAM element located in the GROUP of generated entities (with ref="RGB").

Here is an example of an *ActivityDescription* VOTable that describes an activity to create an RGB image from three input images mapped to the red, green, blue image planes in the composition.

```
<VOTABLE xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.ivoa.net/xml/VOTable/v1.3" version="1.3"
    xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.3
    http://www.ivoa.net/xml/VOTable/v1.3">
  <RESOURCE ID="make_RGB_image" name="make_RGB_image"
      type="meta" utype="voprov:ActivityDescription">
    <DESCRIPTION>Create an RGB image from 3 images</DESCRIPTION>
    <LINK content-role="doc" href="..." />
    <PARAM name="name" datatype="char" arraysize="*"
        value="make_RGB_image" utype="voprov:ActivityDescription.label" />
    <PARAM name="type" datatype="char" arraysize="*"
        value="..." utype="voprov:ActivityDescription.type"/>
    <PARAM name="subtype" datatype="char" arraysize="*"
        value="..." utype="voprov:ActivityDescription.subtype" />
    <PARAM name="version" datatype="float"
        value="..." utype="voprov:ActivityDescription.version" />
    <PARAM name="contact_name" datatype="char" arraysize="*"
        value="..." utype="voprov:Agent.name" />
    <PARAM name="contact_email" datatype="char" arraysize="*"
        value="...@..." utype="voprov:Agent.email" />
    <GROUP name="InputParams" utype="voprov:ParameterDescription">
      <PARAM ID="RGB" arraysize="*" datatype="char" name="RGB"
          type="no_query" value="RGB.jpg">
        <DESCRIPTION>RGB image name</DESCRIPTION>
      </PARAM>
    </GROUP>
    <GROUP name="Used" utype="voprov:UsedDescription">
      <GROUP name="R" utype="voprov:EntityDescription">
        <DESCRIPTION>Image for red channel</DESCRIPTION>
        <PARAM name="default" value="R.jpg" arraysize="*" datatype="char"
            utype="voprov:Entity.id" />
        <PARAM name="role" value="red" arraysize="*" datatype="char"
            utype="voprov:UsedDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
            utype="voprov:EntityDescription.content_type" />
      </GROUP>
      <GROUP name="G" utype="voprov:EntityDescription">
        <DESCRIPTION>Image for green channel</DESCRIPTION>
        <PARAM name="default" value="G.jpg" arraysize="*" datatype="char"
            utype="voprov:Entity.id" />
        <PARAM name="role" value="green" arraysize="*" datatype="char"
            utype="voprov:UsedDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
```

```
          utype="voprov:EntityDescription.content_type" />
      </GROUP>
      <GROUP name="B" utype="voprov:EntityDescription">
        <DESCRIPTION>Image for blue channel</DESCRIPTION>
        <PARAM name="default" value="B.jpg" arraysize="*" datatype="char"
            utype="voprov:Entity.id" />
        <PARAM name="role" value="blue" arraysize="*" datatype="char"
            utype="voprov:UsedDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
            utype="voprov:EntityDescription.content_type" />
      </GROUP>
    </GROUP>
    <GROUP name="Generated" utype="voprov:WasGeneratedByDescription">
      <GROUP name="RGB" utype="voprov:EntityDescription">
        <DESCRIPTION>RGB image generated</DESCRIPTION>
        <PARAM name="role" value="RGB" arraysize="*" datatype="char"
            utype="voprov:WasGenereratedByDescription.role" />
        <PARAM name="content_type" value="image/jpeg" arraysize="*" datatype="char"
            utype="voprov:EntityDescription.content_type" />
      </GROUP>
      </PARAM>
    </GROUP>
  </RESOURCE>
</VOTABLE>
```

3.5 W3C PROV-DM compatible serializations

According to our minimum requirements (see Section 1.2), it must be possible to serialize the provenance metadata into a format compatatible with the W3C Provenance Data Model (W3C PROV-DM), so that it can be exchanged within a wider context and can be processed by already existing tools, e.g. for visualizing provenance. W3C PROV-DM is a larger set of classes and relations compared to this model, but sharing the same core structure. It allows the possibility to add IVOA or *ad hoc* attributes to the basic ones in each class. Thus we can add our additional attributes without problems and still be W3C conform. However, we also defined a few additional classes and relations that are not W3C conform and thus need to be restructured. Using "voprov" as the namespace prefix for our model and "prov" for W3C PROV-DM, the necessary changes for mapping from ProvenanceDM to W3C PROV-DM are listed in the following paragraphs.

Mapping of classes and attributes

- namespace `voprov` → `prov` for those attributes that are the same in W3C (e.g. ID, role, startTime, endTime)
- attribute `voprov:name` → `prov:label`
- attribute `voprov:annotation` → `prov:description`
- attribute `prov:role` is not allowed in W3C's *WasAttributedTo*, thus use `voprov:role`

- *hadMember* has no ID and no optional attributes in W3C
- *Collection* → *Entity* with `prov:type = 'prov:collection'`

Description classes

- *ActivityDescription* becomes an Entity with `prov:type = 'voprov:ActivityDescription'`. It is close to the Plan concept in W3C PROV, and it could have in addition `prov:type = 'prov:plan'` (note that W3C PROV accepts several types). This class is used by the Activity class with `prov:role = 'voprov:ActivityDescription'`.
- *EntityDescription* becomes an Entity of type `prov:type = 'voprov:EntityDescription'`. It is then linked to an Entity with the specializationOf relation.
- *UsedDescription* and *WasGeneratedByDescription* also become entities.

We also envision to group all description classes into a prov:Bundle that is then connected to the Activity with the Used relation and `prov:role = 'voprov:ActivityDescription'`.

Parameter class A parameter can be seen as a simplified entity and should thus be serialized in the same way as an entity, with prov:type = voprov:Parameter. The ParameterDescription becomes an Entity with prov:type = voprov:ParameterDescription and a specializationOf relation to the parameter entity.

ActivityFlow class

- *ActivityFlow* → *Activity* with additional attribute `voprov:votype = 'voprov:activityFlow'`
- replace *HadStep* relation by W3C's general *WasInfluencedBy* relation with additional attribute `voprov:votype = 'voprov:hadStep'` or just use `voprov:hadStep` as attribute in activities of type activityFlow.

This way, it is possible to produce W3C compatible serializations of our model with minimum information loss. W3C tools would ignore the voprov-attributes, whereas VO clients could make sense of this additional information and could even uncover the original structure or convert it to a VO serialization.

4 Accessing provenance information

We envision two possible access protocols:

- ProvDAL: 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.

Both protocols will be specified in separate documents.

Appendix A Serialization Examples

Here is a a simple example of serialization of ProvenanceDM metadata for describing an activity of color composition and the entity used as input as well as the resulting RGB image.

The PROV-N format Moreau and Missier (2013) as proposed by the W3C is a text format which allows the description of instances of the 3 main classes, as well as the various relations between each instance involved.

Listing 1: PROV-N serialisation example for a color composition activity

```
document
  prefix ivo <http://www.ivoa.net/documents/rer/ivo/>
  prefix hips <http://cds.u-strasbg.fr/data/>
  prefix voprov <http://www.ivoa.net/documents/dm/provdm/voprov/
     >

  entity(ivo://CDS/P/DSS2color#RGB_NGC6946,
        [voprov:annotation="PNG RGB image  built  from  DSS2  with
           Aladin  for  galaxy NGC 6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gml",
        voprov:name="RGB DSS2  image  for  NGC  6946"])

  entity(ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143,
        [voprov:annotation="DSS2  digitization  of  the  Blue  POSSII
            Schmidt  survey  around   NGC 6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gm",
        voprov:name="POSSII  Blue  Survey  DSS2  NGC6946"])

  entity(ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143,
        [voprov:annotation="DSS2  digitization  of  the  Red  POSSII
           Schmidt  survey  around NGC  6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gml",
        voprov:name="POSSII  Red  Survey  DSS2  NGC6946"])

  entity(ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143,
        [voprov:annotation="DSS2 digitization of the Infra red
            POSSII Schmidt survey around NGC 6946",
        voprov:doculink="http://cds.u-strasbg.fr/aladin.gm",
        voprov:name="POSSII Infra Red Survey DSS2 NGC6946"])

  activity(hips:AlaRGB1, 2017-04-18T17:28:00, 2017-04-19
      T17:29:00, [
        voprov:name="Aladin RGB 1",
        voprov:annotation="Aladin RGB image generation for NGC
            6946",
        voprov:desc_id="AlaRGB",
        voprov:desc_type="RGBencoding",
        voprov:desc_name="Aladin RGB image generation algorithm"
            ,
        voprov:desc_doculink="http://cds.u-strasbg.fr/aladin.gml
            "])

  used(hips:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143'
      , -)
  used(hips:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143'
      , -)
  used(hips:AlaRGB1, 'ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143'
      , -)

  wasGeneratedBy('ivo://CDS/P/DSS2color#RGB_NGC6946',
      hips:AlaRGB1, 2017-05-05T00:00:00)
endDocument
```

Here is the transcription of the same metadata in the PROV-JSON format (Huynh and Jewell et al., 2013). Each class and relation of the provenance model lists its corresponding database table tuples grouped by the name of the table.

Listing 2: JSON serialization example for a color composition activity

```
{
  "prefix": {
    "ivo": "http://www.ivoa.net/documents/rer/ivo/",
    "voprov": "http://www.ivoa.net/documents/dm/provdm/voprov/",
    "hips": "http://cds.u-strasbg.fr/data/"
  },
  "activity": {
    "hips:AlaRGB1": {
      "voprov:name": "Aladin RGB 1",
      "voprov:startTime": "2017-04-18T17:28:00",
                          "voprov:endTime": "2017-04-19T17:29:00",
      "voprov:annotation": "Aladin RGB image generation for NGC
          6946",
      "voprov:desc_type": "RGBencoding",
      "voprov:desc_name": "Aladin RGB image generation algorithm
          ",
      "voprov:desc_doculink": "http://cds.u-strasbg.fr/aladin.
          gml",
      "voprov:desc_id": "AlaRGB"
    }
  },
  "wasGeneratedBy": {
    "_:id4": {
      "voprov:time": "2017-05-05T00:00:00",
      "voprov:entity": "ivo://CDS/P/DSS2color#RGB_NGC6946",
      "voprov:activity": "hips:AlaRGB1"
    }
  },
  "used": {
    "_:id1": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2
          .143",
      "voprov:activity": "hips:AlaRGB1"
    },
    "_:id3": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2
          .143",
      "voprov:activity": "hips:AlaRGB1"
    },
    "_:id2": {
      "voprov:entity": "ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2
          .143",
      "voprov:activity": "hips:AlaRGB1"
    }
  },
  "entity": {
    "ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143": {
      "voprov:name": "POSSII Blue Survey DSS2 NGC6946",
      "voprov:annotation": "DSS2 digitization of the Blue POSSII
           Schmidt survey around  NGC 6946",
      "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gm"
    },
    "ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143": {
      "voprov:name": "POSSII Red Survey DSS2 NGC6946",
      "voprov:annotation": "DSS2 digitization of the Red POSSII
          Schmidt survey around NGC 6946",
      "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gml"
    },
    "ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143": {
      "voprov:name": "POSSII Infra Red Survey DSS2 NGC6946",
      "voprov:annotation": "DSS2 digitization of the Infra Red
          POSSII Schmidt survey around NGC 6946",
      "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gm"
    },
    "ivo://CDS/P/DSS2color#RGB_NGC6946": {
      "voprov:name": "RGB DSS2 image for NGC 6946",
      "voprov:annotation": "PNG RGB image built from DSS2 with
          Aladin for galaxy NGC 6946",
      "voprov:doculink": "http://cds.u-strasbg.fr/aladin.gml"
    }
  }
}
```

Here is the mapping obtained for the same data description with a serialization in VOTable format.

Listing 3: VOTable serialization example for a color composition activity

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.2" xmlns="http://www.ivoa.net/xml/VOTable/v1
    .2"
xmlns:hips="http://cds.u-strasbg.fr/data/"
xmlns:ivo="http://www.ivoa.net/documents/rer/ivo/"
xmlns:voprov="http://www.ivoa.net/documents/dm/provdm/voprov/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ivoa.net/xml/VOTable/v1.2 http://
   www.ivoa.net/xml/VOTable/VOTable-1.2.xsd">
  <RESOURCE type="results">
    <DESCRIPTION>Provenance VOTable</DESCRIPTION>
    <TABLE name="Used" utype="voprov:Used">
      <FIELD name="activity_id" utype="voprov:Used.activity"
          arraysize="*" ucd="meta.id" datatype="char" />
      <FIELD name="entity_id" utype="voprov:Used.entity"
          arraysize="*" ucd="meta.id" datatype="char" />
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>hips:AlaRGB1</TD>
            <TD>ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="WasGeneratedBy" utype="voprov:WasGeneratedBy">
      <FIELD name="entity" utype="voprov:wasGeneratedBy.entity"
          arraysize="*" datatype="char" ucd="meta.id"/>
      <FIELD name="activity" utype="voprov:wasGeneratedBy.
          activity" arraysize="*" datatype="char" ucd="meta.id"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>ivo://CDS/P/DSS2color#RGB_NGC6946</TD>
            <TD>hips:AlaRGB1</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <TABLE name="Activity" utype="voprov:Activity">
      <FIELD name="id" utype="voprov:Activity.id" ucd="meta.id"
           arraysize="*" datatype="char" />
      <FIELD name="name" utype="voprov:Activity.name" ucd="meta.
          title" arraysize="*" datatype="char" />
      <FIELD name="start" utype="voprov:Activity.startTime" ucd=
          "time.start" arraysize="*" datatype="char" />
      <FIELD name="end" utype="voprov:Activity.endTime" ucd=""
          time.end" arraysize="*" datatype="char" "/>
    <FIELD name="annotation" utype="voprov:Activity.annotation
        " ucd="meta.description" arraysize="*" datatype="char"
        />
    <FIELD name="desc_id" utype="voprov:ActivityDescription.id
        " ucd="meta.id" arraysize="*" datatype="char" />
    <FIELD name="desc_name" utype="voprov:ActivityDescription.
        name" ucd="meta.title" arraysize="*" datatype="char" />
    <FIELD name="desc_type" utype="voprov:ActivityDescription.
        type" ucd="meta.code.class" arraysize="*" datatype="
        char" />
    <FIELD name="desc_doculink" utype="
        voprov:ActivityDescription.doculink" ucd="meta.ref.url"
         arraysize="*" datatype="char" />
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>hips:AlaRGB1</TD>
          <TD>Aladin RGB 1</TD>
          <TD>2017-04-18 17:28:00</TD>
          <TD>2017-04-19 17:29:00</TD>
          <TD>Aladin RGB image generation for NGC 6946</TD>
          <TD>AlaRGB</TD>
          <TD>Aladin RGB image generation algorithm</TD>
          <TD>RGB encoding</TD>
          <TD>http://cds.u-strasbg.fr/aladin.gml</TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
  <TABLE name="Entity" utype="voprov:Entity">
    <FIELD name="id" utype="voprov:Entity.id" ucd="meta.id"
        arraysize="*" datatype="char" />
    <FIELD name="name" utype="voprov:Entity.name" ucd="meta.
        title" arraysize="*" datatype="char" />
    <FIELD name="annotation" utype="voprov:Entity.annotation"
        ucd="meta.description" arraysize="*" datatype="char" />
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>ivo://CDS/P/DSS2/POSSII#POSSII.J-DSS2.143</TD>
          <TD>POSSII Blue Survey DSS2 NGC6946</TD>
          <TD>DSS2 digitization of the Blue POSSII Schmidt
              survey around NGC 6946</TD>
        </TR>
        <TR>
          <TD>ivo://CDS/P/DSS2/POSSII#POSSII.F-DSS2.143</TD>
          <TD>POSSII Red Survey DSS2 NGC6946</TD>
          <TD>DSS2 digitization of the Red POSSII Schmidt
              survey around NGC 6946</TD>
        </TR>
        <TR>
          <TD>ivo://CDS/P/DSS2/POSSII#POSSII.N-DSS2.143</TD>
          <TD>POSSII Infra Red Survey DSS2 NGC6946</TD>
          <TD>DSS2 digitization of the Infra red POSSII
```

```
                Schmidt survey around NGC 6946</TD>
          </TR>
          <TR>
            <TD>ivo://CDS/P/DSS2color#RGB_NGC6946</TD>
            <TD>RGB DSS2 image for NGC 6946</TD>
            <TD>PNG RGB image built from DSS2 with Aladin for
                galaxy NGC 6946</TD>
          </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
    <INFO name="QUERY_STATUS" value="OK"/>
  </RESOURCE>
</VOTABLE>
```

Appendix B Links to other data models

The Provenance Data Model can be applied without making any links to other IVOA data model classes. For example, when the data is not yet published, provenance information can be stored already, but a DatasetDM-description for the data may not yet exist. However, if there are data models implemented for the datasets, then it is very useful to connect the classes and attributes of the other data models with provenance classes and attributes (if applicable), which we are going to discuss in this Section. These links help to avoid unnecessary repetitions in the metadata of datasets, and also offer the possibility to derive some basic provenance information from existing data model classes automatically, e.g. for creating provenance serializations from DatasetDM or SimDM metadata.

B.1 Links with Dataset/ObsCore Model

Entity and *EntityDescription* in ProvenanceDM are tightly linked to the *DataSet*-class in DatasetDM/ObsCoreDM, as well as to *InputDataset* and *OutputDataSet* in the SimulationDM (Lemson and Wozniak et al., 2012). Tables 17 and 18 map classes and attributes from DatasetDM to concepts in ProvenanceDM.

The *Agent* class, which is used for defining responsible persons and organizations in ProvenanceDM, is very similar to the *Party* class in DatasetDM (and in SimDM). Its details are depicted in Figure 10. The main difference between *Agent* and *Party* is that *Individual* and *Person* are subclasses in DatasetDM, whereas we just use the same class *Agent* for both and distinguish between them using the *Agent.type* attribute (which can have the value “Individual” or “Organization”), which is closer to W3C’s provenance data model.

ProvenanceDM attribute	**DatasetDM attribute**	**Comment**
Entity.id	Curation.PublisherDID	unique identifier for the dataset assigned by the publisher
Entity.id	DataID.creatorDID	alternative id for the dataset given by the creator, could be used as Entity.id if no PublisherDID exists (yet)
Entity.name	DataID.title	title of the dataset
Entity.rights	Curation.Rights	access rights to the dataset; one of [...]
Entity.creationTime	DataID.date	date and time when the dataset was completely created
HadMember.collection	DataID.collection	link to the collection to which the dataset belongs
WasGeneratedBy.activityId	DataID.ObservationID	identifier to everything describing the observation
Agent	Curation.Contact	link to Agent with role contact
Agent.id	Curation.PublisherID	link to the publisher, i.e. to an Agent with role="publisher"
Agent.name, wasAttributedTo.role= Creator	DataID.creator	name of agent creating the dataset
Agent.name, wasAttributedTo.role= Publisher	Curation.Publisher	name of the publisher

Table 17: Mapping attributes from DatasetDM classes to (optional) attributes in ProvenanceDM. This list is not complete.

We imagine that services implementing both data models, *Dataset* and *ProvenanceDM* may just use *one* class: either *Agent* or *Party*, enriched with all the necessary (project-specific) attributes. When delivering the data on request, the serialised versions can be adjusted to the corresponding notation. Note that for Provenance queries using a ProvTAP service or for W3C compatible serializations, the name *Agent* for the responsible individuals/organizations is required.

B.2 Links with Simulation Data Model

In SimDM one also encounters a normalization similar to our separation of descriptions from actual data instances and executions of processes: the SimDM class "experiment" is a type of *Activity* and its general, reusable

ProvenanceDM class	**DatasetDM attribute**	**Comment**
Entity	Curation.Date	release date of the dataset
Entity	Curation.Version	version of the dataset
Entity	Curation.Reference	link to publication
EntityDescription	DataProductType	the type of a dataproduct from DatasetDM can be used as attribute to EntityDescription
EntityDescription	DataProductSubType	subtype of a dataproduct/entity
EntityDescription	ObsDataset.calibLevel	(output) calibration level, integer between 0 and 3

Table 18: Mapping attributes from DatasetDM classes to the ProvenanceDM classes to which they could be added. Attributes like `EntityDescription.calibLevel` are very specific to entities described with DatasetDM and thus are not included in this ProvenanceDM directly. This list is not complete.

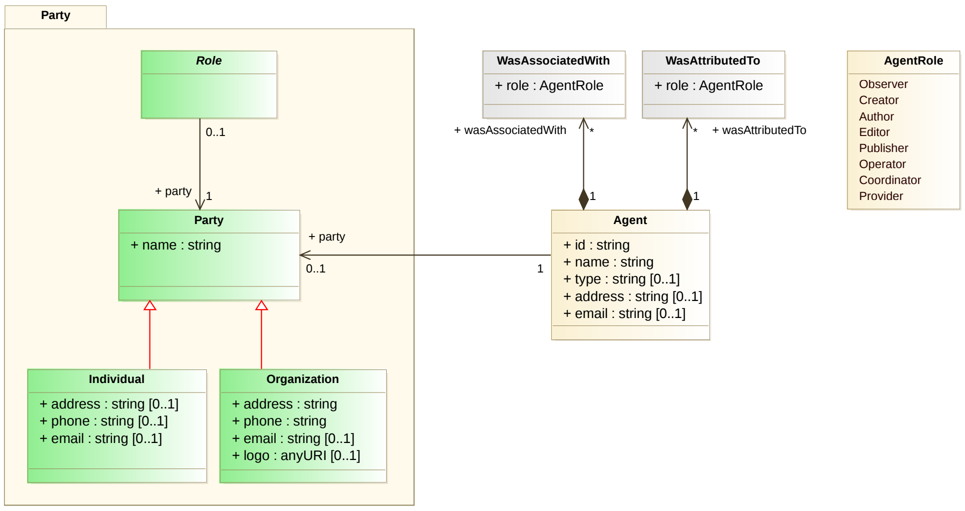

Figure 10: The relations between the *Agent* class within ProvenanceDM (grey and yellow classes) with classes from the DatasetDM, party package (green).

description is called a “protocol”, which can be considered as a type of this model’s *ActivityDescription*. More direct mappings between classes and attributes of both models are given in Table 19.

If simulations are already described with SimDM, this table can be used to map from SimDM properties to ProvenanceDM, e.g. when serving se-

ProvenanceDM	**SimDM**	**Comment**
Activity	Experiment	
Activity.name	Experiment.name	human readable name; name attribute in SimDM is inherited from Resource-class
Activity.endTime	Experiment.executionTime	end time of the execution of an experiment/activity
Activity.description	Experiment.protocol	reference to the protocol or ActivityDescription class
ActivityDescription	Protocol	
ActivityDescription.name	Protocol.name	human readable name
ActivityDescription.doculink	Protocol.referenceURL	reference to a webpage describing it
Parameter	ParameterSetting	value of an (input) parameter
ParameterDescription	InputParameter	description of an (input) parameter
Agent	Party	responsible person or organization
Agent.name	Party.name	name of the agent
WasAssociatedWith	Contact	classes for linking to agent/party
WasAssociated-With.role	Contact.role	role which the agent/party had for a certain experiment (activity); SimDM roles contain: `owner`, `creator`, `publisher`, `contributor`
WasAssociated-With.agent	Contact.party	reference to the agent/party
Entity	DataObject	a dataset, which can be/refer to a collection

Table 19: Mapping between classes and attributes from SimDM to classes/attributes in ProvenanceDM. This list is not complete.

rialized provenance metadata via an additional ProvDAL interface or for storing provenance metadata together with each released simulation dataset (e.g. in a VOTable).

Appendix C Changes from Previous Versions

C.1 Changes from WD-ProvenanceDM-1.0-20170921

- Moved ProvDAL to separate document.
- Moved ProvTAP section and full definition of VOTable serialisation to separate ProvTAP document.
- Moved chapter 6 with use cases and "How to use the data model" to separate Implementation Note (Riebe and Servillat et al., 2017).
- Moved section on links to other data model into appendix.
- ParameterDescription: Added attributes `xtype` and `arraysize`.
- Agent.roles: Removed "PI" alternative to "coordinator".
- Use values of RightsType of DatasetDM, public, secure, proprietary, for `Entity.rights`.
- Minor corrections in HiPS use case, Appendix A and tables in TAP schema.
- Minor correction in role names in Figure 3 for hadStep/hadMember relationship.
- Modified text on Parameters
- Rename 3.4 section to Serialization of description classes for web services
- Modified text on W3C serialization
- add `location` and `value` attributes to *Entity*

C.2 Changes from WD-ProvenanceDM-1.0-20161121

- New appendix for PROV-VOTable/TAP SCHEMA tables added
- Corrected and extended attribute tables and mapping tables for links with DatasetDM and SimDM.
- Restructured Accessing provenance section by splitting it in two: Section 3 for explaining the different serialization formats and differences to W3C serializations, Section "Accessing provenance information"for describing the access protocols ProvDAL and ProvTAP.
- Removed discussion section, since now all the topics are addressed in the main text.

- Added paragraph on how to use the model in Section 6.
- Shortened serialization examples, partially moved them to appendix.
- Added paragraph on VOSI interface.
- Added a proposed serialization of description classes.
- Modified text on the content of EntityDescription, now seen as Entity attributes known before the Entity instance exists.
- Renamed Section 6 to stress that it explains applications of the model (use cases); implementation details and code examples can be found in Implementation Note (Riebe and Servillat et al., 2017).
- Complete rewrite of the ProvDAL section in Section "Accessing provenance information";new parameters, new figure and examples added.
- Added additional figure for entity-activity relations.
- Moved the figure showing relations between Provenance.Agent and Dataset.Party into Section B.
- Extended the entity role examples in table 11.
- Added links to provn and votable-serialization for HiPS-use case, added first part of provn as example in the HiPS-use case section.
- More explanations on links to data models in Section B, introduced subsections, added table with SimDM-mapping.
- Moved detailed implementation section from appendix to a separate document (implementation note), shortened the use cases & implementation section.
- Attribute/class updates:
  - Added attribute *votype* to *Activity*, can be used for ActivityFlows
  - Added attribute *time* to *Used* and *WasGeneratedBy*
  - Added optional attributes *Entity.creationTime* and *EntityDescription.category*
  - Added optional attributes *Parameter.min*, *Parameter.max*, *Parameter.options*
  - Removed the obscore/dataset attributes from EntityDescription, since they are specific for observations only and are not applicable to configuration entities etc.

- Use voprov:type and voprov:role in Table 16 with example agent roles, i.e. replaced prov:person by Individual and prov:organization by Organization.
- Renamed *label* attribute to *name* everywhere, for more consistency with SimDM naming scheme (*label* is reserved there for SKOS labels).
- Renamed attribute *Entity.access* to *Entity.rights* for more consistency with DatasetDM etc.
- Avoid double-meaning of *description* (as reference and free-text description) by renaming the free-text description to *annotation*. Mark description-references with arrows in attribute tables.
- Applied similar naming scheme to *Parameter* and *ParameterDescription*-classes
- Renamed *docuLink* to *doculink*
- Corrected attribute names in Table 17.

List of Figures

List of Tables

Bibliography

Belhajjame, K., B'Far, R., Cheney, J., Coppens, S., Cresswell, S., Gil, Y., Groth, P., Klyne, G., Lebo, T., McCusker, J., Miles, S., Myers, J., Sahoo, S. and Tilmes, C. (2013), 'PROV-DM: The prov data model', W3C Recommendation.
http://www.w3.org/TR/prov-dm/

Bonnarel, F., Demleitner, M., Dowler, P., Tody, D. and Dempsey, J. (2017), 'Ivoa server-side operations for data access, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/SODA/

Bonnarel, F., Laurino, O., Lemson, G., Louys, M., Rots, A., Tody, D. and the IVOA Data Model Working Group (2015), 'IVOA dataset metadata model', IVOA Working draft.
http://www.ivoa.net/documents/DatasetDM/

Bonnarel, F. and the IVOA Data Model Working Group (2016), 'Provenance data model legacy', Webpage.
http://wiki.ivoa.net/twiki/bin/view/IVOA/ProvenanceDataModelLegacy

Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119.
http://www.ietf.org/rfc/rfc2119.txt

Dowler, P., Bonnarel, F., Michel, L. and Demleitner, M. (2015), 'IVOA datalink', IVOA Recommendation 17 June 2015.
http://www.ivoa.net/documents/DataLink/

Hanisch, R., the IVOA Resource Registry Working Group and the NVO Metadata Working Group (2007), 'Resource metadata for the virtual observatory, version 1.12', IVOA Recommendation.
http://www.ivoa.net/documents/latest/RM.html

Harrison, P. and Rixon, G. (2010), 'Universal worker service pattern, version 1.0', IVOA Recommendation.
http://www.ivoa.net/documents/UWS/20101010/

Hua, H., Tilmes, C., Zednik, S. and Moreau, L. (2013), Prov-xml: The prov xml schema, Project report.
https://eprints.soton.ac.uk/356855/

Huynh, T. D., Jewell, M., Sezavar Keshavarz, A., T. Michaelides, D., Yang, H. and Moreau, L. (2013), 'The prov-json serialization'.
https://www.w3.org/Submission/2013/SUBM-prov-json-20130424/

IVOA Data Model Working Group (2005), 'Data model for observation, version 1.00', IVOA Note.
http://www.ivoa.net/documents/latest/DMObs.html

IVOA Data Model Working Group (2008), 'Data model for astronomical dataset characterisation, version 1.13', IVOA Recommendation.
http://www.ivoa.net/documents/latest/CharacterisationDM.html

Lemson, G., Wozniak, H., Bourges, L., Cervino, M., Gheller, C., Gray, N., LePetit, F., Louys, M., Ooghe, B. and Wagner, R. (2012), 'Simulation Data Model Version 1.0', IVOA Recommendation 03 May 2012, arXiv:1402.4744.
http://adsabs.harvard.edu/abs/2012ivoa.spec.0503L

McDowell, J., Salgado, J., Blanco, C. R., Osuna, P., Tody, D., Solano, E., Mazzarella, J., D'Abrusco, R., Louys, M., Budavari, T., Dolensky, M., Kamp, I., McCusker, K., Protopapas, P., Rots, A., Thompson, R., Valdes, F., Skoda, P., Rino, B., Cant, J., Laurino, O., the IVOA Data Access Layer and Groups, D. M. W. (2016), 'Ivoa spectral data model, version 2.0', IVOA Draft.
http://www.ivoa.net/documents/SpectralDM/

Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E. and den Bussche, J. V. (2010), 'The open provenance model core specification (v1.1)', Future Generation Computer Systems, 27, (6), 743-756. (doi:10.1016/j.future.2010.07.005), University of Southampton.
http://openprovenance.org/;http://eprints.soton.ac.uk/271449/

Moreau, L. and Missier, P. (2013), 'PROV-N: The provenance notation', W3C Recommendation.
http://www.w3.org/TR/prov-n/

Ochsenbein, F., Williams, R., Davenhall, C., Demleitner, M., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A., Taylor, M. and Wicenec, A. (2013), 'VOTable format definition, version 1.3', IVOA Recommendation.
http://www.ivoa.net/documents/VOTable/

Riebe, K., Servillat, M., Bonnarel, F., Louys, M., Sanguillon, M. and the IVOA Data Model Working Group (2017), 'Provenance implementation note', IVOA Note.
http://volute.g-vo.org/svn/trunk/projects/dm/provenance/implementation-note/