



---

# ProvTAP evolution after first community feedback

F.Bonnarel, CDS

on behalf of M.Servillat, M.Louys, M.Nullmeier, M.Sanguillon,  
L.Michel



# Why a ProvTAP specification ?

- Provenance information can be attached to data in various ways :
  - Embedded in the data « header » itself
  - Linked to the data record via DataLink or URL
  - Retrievable via ProvSAP via data id.
- In addition to that , ProvTAP allows to discover « data » by constraining Provenance features.
  - It's a « reverse » mechanism.



# Why a ProvTAP specification ?

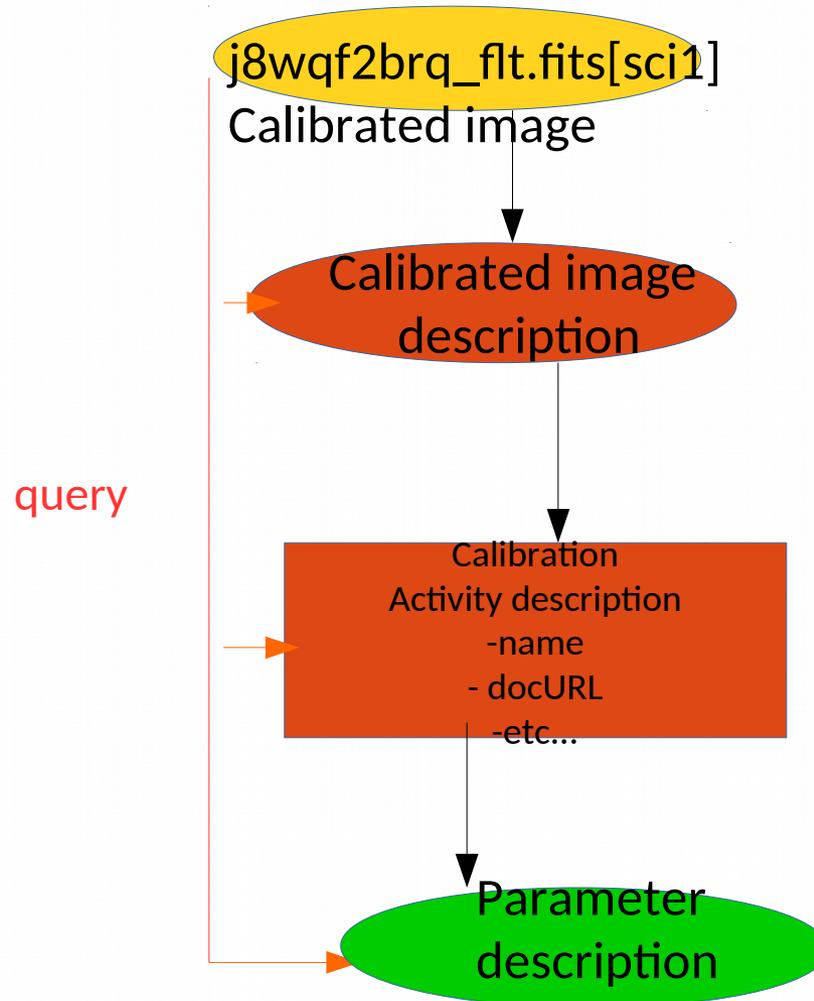
## Examples :

- Discovering data produced with the same version of a given software.
- Discovering data produced with some specific value of a software configuration parameter :
  - Known by its name
  - Known by its ucd
  - ....
- Discovering a « family » of data : datasets produced by ancestor activities of a given datasets
- Discovering data or activities related to some agent with a given role (operator, editor, author, etc...)



# ProvHiPS ADQL query examples :

Find out parameter descriptions of parameter used to generate a calibrated file



# ProvHiPS ADQL query examples :

Find out parameter descriptions of parameter used to generate a calibrated file

The screenshot displays the TOPCAT Table Browser interface. On the left, a metadata tree shows a hierarchy of tables, with 'entity' selected under the 'provenance' folder. The main window shows a table with 3 rows and 14 columns. A 'Table Browser' window is overlaid on top, showing the same table data.

	e_name	pd_activityd...	pd_id	pd_name	pd_description	pd_do...	pd_val...	pd_unit	pd_ucd	pd_uty...	pd_min	pd_max	pd_def...	pd_opt...
1	j8wqf2brqflt.fits[sc1]	calibAdescri	pd_1002	pftfile	pixel to pixel flat field file name		char		meta.file:obs.calib.flat					
2	j8wqf2brqflt.fits[sc1]	calibAdescri	pd_1001	biasfile	bias image file name		char		meta.file:obs.calib.bias					
3	j8wqf2brqflt.fits[sc1]	calibAdescri	pd_1000	darkfile	dark image file name		char		meta.file:obs.calib.dark					

ADQL Text:

```
1
select top 3 e_name,parameterdescription.* from entity
join datasetdescription on e_description = dd_id
join generationdescription on gd_entitydescription =dd_id
join activitydescription on ad_id = gd_activitydescription
join parameterdescription on pd_activitydescription = ad_id
where e_name = 'j8wqf2brqflt.fits[sc1]';
```

Run Query

## ProvHiPS ADQL query examples :

Find out parameter descriptions of parameter used to generate a calibrated file

```
select top 3 * from entity
join datasetdescription on e_description = dd_id
join generationdescription on gd_entitydescription = dd_id
join activitydescription on ad_id = gd_activitydescription
join parameterdescription on pd_activitydescription = ad_id
where e_name = 'j8wqf2brqflt.fits[sci1]' ;
```



# ProvTAP = where are we ?

- There is an internal draft on the IVOA DAL pages
- Was not a WD by lack of discussion
- TAP schema mapping classes as tables
- ProvHiPS (provenance of HiPS and HiPS tiles) is an implementation prototype
- Discussion among authors on various points (see DAL running meeting slides)



## IVOA Provenance Table Access Protocol (ProvTAP)

Version 1.0

IVOA Working Draft 2018-03-22

Working group

DM

This version

<http://www.ivoa.net/documents/ProvTAP/20180322>

Latest version

<http://www.ivoa.net/documents/ProvTAP>

Previous versions

Author(s)

François Bonnarel, Mireille Louys, Markus Nulmeier, Kristin Riebe, Michèle Sanguillon, Mathieu Servillat, IVOA Data Model Working Group

Editor(s)

François Bonnarel

### Abstract

This document describes the ProvTAP protocol for accessing provenance information according to the IVOA ProvenanceDM standard. It defines how the elements of ProvenanceDM are described in the TAP schema tables and provides guidelines for implementing with TAP 1.1.

# ProvTAP = where are we ?

## External point of view

- DAL chairs
  - 1 to 1 class/table mapping too ambitious. Need for simplification/ denormalization
- ESFRI projects (within ESCAPE)
  - Looking for simplified/partial views
    - Tracing the last step ?
    - Concept of « ProvCore » : minimal model attributes ?
    - Depends from project requirements and uses cases



1 table  
per  
Class ?

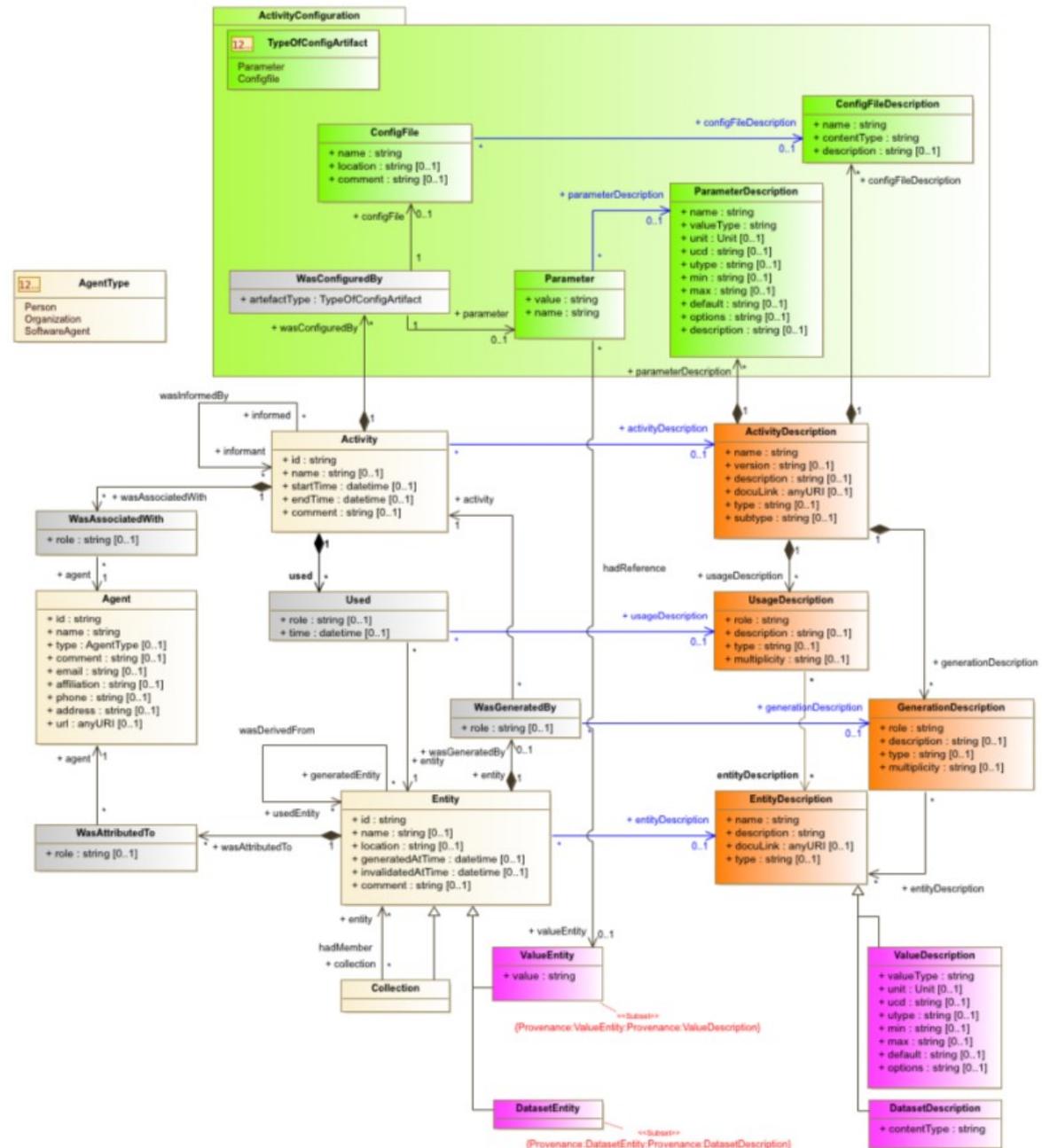


Figure 8: Full class diagram of the IVOA Provenance Data Model.

# Solutions

## -1 Single step = single table (= join)

- The join is a permanent view described in the TAP schema
- Columns :
  - entity\_name, entity\_location, entity\_comment, ...
  - generating\_activity\_name, generating\_activity\_starttime, ....
  - agent\_role, agent\_name, ....
  - used\_entity\_name
- → Redundancy (several lines for a single entity/activity).
- → possible Recursivity



# Solutions

-1 Single step = single table (= join)

- View (in postgres)

create view last\_step\_provenance as select

e.e\_name as entity\_name, e.e\_location as entity\_location, e.e\_generated as entity\_generated, e.e\_invalidated as entity\_invalidated, e.e\_comment as entity\_comment,

a\_name as generating\_activity\_name, a\_starttime as generating\_activity\_starttime, a\_endtime as generating\_activity\_endtime, a\_comment as generating\_activity\_comment,

wat\_role as agent\_role, ag\_name as agent\_name, ag\_type as agent\_type, ag\_affiliation as agent\_affiliation, ag\_email as agent\_email, ag\_address as agent\_address, ag\_phone as agent\_phone, ag\_comment as agent\_comment,

ee.e\_name as used\_entity\_name from entity as e

**join wasgeneratedby on wgb\_entity = e.e\_id**

*join activity on a\_id = wgb\_activity*

**join used on u\_activity = a\_id**

*join entity as ee on ee.e\_id = u\_entity*

**join wasattributedto on wat\_entity = e.e\_id**

*join agent on ag\_id = wat\_agent ;*



# Solutions

-1 Single step = single table (= join)

Mode: Synchronous



1

```
select * from last_step_provenance where entity_name = 'j90x37020_drc' ;
```





# Solutions

## 2- denormalization

- Adding « description » classes attributes to the « execution » classes.
  - This is another permanent view
  - A lot of redundancy.

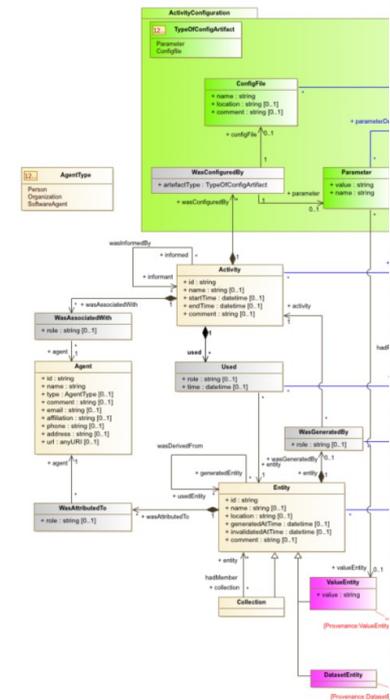


Figure 8: Full class diagram of the IVO

# Solutions

## 2- denormalization

- Full entity :
  - entity\_name
  - entity\_location
  - entity\_comment
  - entity\_generated
  - entity\_invalidated
  - entitydescription\_content
  - entitydescription\_type
  - entitydescription\_description
  - entitydescription\_docurl



# Solutions

## 2- denormalization

- Full activity :
  - activity\_name
  - activity\_comment
  - activity\_starttime
  - activity\_endtime
  - activitydescription\_description
  - activitydescription\_type
  - activitydescription\_docurl
- And parameter/parameterdescription



# Solutions

## 3 - Simplification of « descriptions » « linkage »

- Suppress UsageDescription and GenerationDescription ?
- May introduce other difficulties ?



# How to go on ?

- Keep full ProvTAP schema and propose views.
- Make these views become simplified TAP schemata part of the standard.
- Limited or full services possibility  
(implementing only views or the whole TAP schema)
- Upgrade the specification and open prototype

