International
Virtual
Observatory
Alliance

Simulation Data Access Layer

Version 1.0

IVOA Recommendation 2017-03-20

Working group
DAL
This version
http://www.ivoa.net/documents/simdal/20170320
Latest version
http://www.ivoa.net/documents/simdal
Previous versions
PR-SimDAL-1.0-20170130
Author(s)
David Languignon, Franck Le Petit, Carlos Rodrigo, Gerard Lemson, Marco Molinaro, Hervé Wozniak
Editor(s)
David Languignon, Franck Le Petit

Abstract

The Simulation Data Access Layer protocol (SimDAL) defines a set of resources and associated actions to discover and retrieve simulations and numerical models in the Virtual Observatory. SimDAL and the Simulation Data Model are dedicated to cover the needs for the publication and retrieval of any kind of simulations: N-body or MHD simulations, numerical models of astrophysical objects and processes, theoretical synthetic spectra, etc...

SimDAL is divided in three parts. First, SimDAL Repositories store the descriptions of theoretical projects and numerical codes. They can be used by clients to discover theoretical services associated with projects of interest. Second, SimDAL Search services are dedicated to the discovery of precise datasets. Finally, SimDAL Data Access services are dedicated to retrieve the original simulation output data, as plain raw data or formatted datasets cut-outs.

To manage any kind of data, eventually large or at high-dimensionality, the SimDAL standard lets publishers choose any underlying implementation technology.

Status of This Document

This document has been reviewed by IVOA Members and other interested parties, and has been endorsed by the IVOA Executive Committee as an IVOA Recommendation. It is a stable document and may be used as reference material or cited as a normative reference from another document. IVOA's role in making the Recommendation is to draw attention to the specification and to promote its widespread deployment. This enhances the functionality and interoperability inside the Astronomical Community.
A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/Documents/.

Contents

Acknowledgments

This standard has been developed with the support of the French VO (ASOV) and of the Paris Observatory (Paris Data Center and LERMA) as well as with the support of the Spanish Virtual Observatory (http://svo.cab.inta-csic.es) supported from the Spanish MINECO through grant AyA2014-55216.
In addition, this document was prepared with the IVOATeX package. We thank Markus Demleitner for his help with this tool.

1 Introduction

The Simulation Data Access Layer protocol (SimDAL) defines a set of resources and associated actions to discover and retrieve simulations and numerical models in the Virtual Observatory.

The diffusion of simulations is a more complex topic than the diffusion of astronomical images or spectra. Most of the difficulty comes from the heterogeneity of simulations and the fact that a code can compute many things. SimDAL provides solutions to overcome these challenges.
Many notions in this document refer to classes of the Simulation Data Model (SimDM). SimDM and SimDAL are designed so that any theoretical data (N-body simulations, MHD simulations, theoretical spectra, evolutions of galaxies, structures of astrophysical objects, ...) can be published in the Virtual Observatory, though a few specific use cases are still not considered.

This document is divided in three parts corresponding to the three SimDAL components:

1. **SimDAL Repository** to discover published simulation projects
2. **SimDAL Search** to discover simulation datasets
3. **SimDAL Data Access** to retrieve data

The SimDAL Repository component can be implemented independently of the two other components. It is a fine grain registry for numerical codes and simulations in the Virtual Observatory. Only a few Data Centers should implement and maintain SimDAL Repositories. They are the reference source of code and simulation projects descriptions to which other components refer.

To publish theoretical data, data publishers must implement the SimDAL Data Access component. If they need to provide a search in the parent simulations they have to implement a SimDAL Search component.

1.1 Role within the VO Architecture

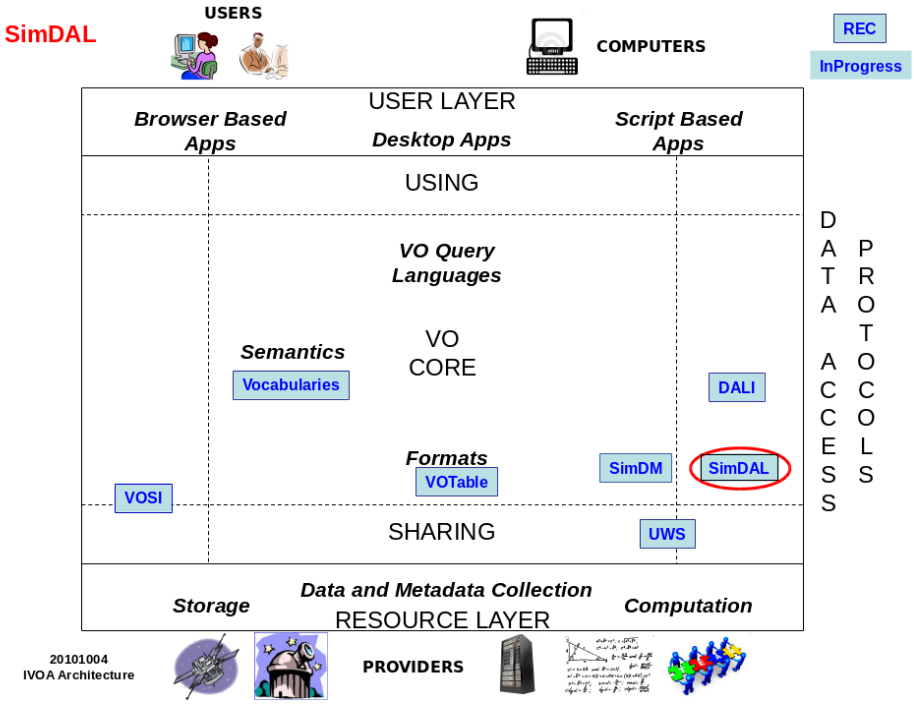

Figure 1: SimDAL in the IVOA architecture

2 Overview

2.1 Rationale

Astronomy and astrophysics have a long tradition of data publication but observational datasets historically deserved much more attention than numerical simulations. Only a few sets of simulations have been published (cf Sect. 3.1 of EuroVO-DCA Theory white paper - Cassisi et al. (2008)). However, access to state-of-the art numerical simulations is more and more required either to prepare new observations or to interpret the masses of astronomical data.

The goal of the Simulation Data Model (SimDM) and of the Simulation Data Access Layer (SimDAL) is to provide a standard frame to describe and publish numerical data and numerical models so that third parties can discover and access them with the Virtual Observatory.

Here is a basic example of such a workflow where scientists try to find simulations to interpret some observations of stars:

Discover a numerical simulations project: First, scientists look for numerical projects simulating *isochrones* (that provide stellar properties as a function of the mass, for several values of the age of the star). This is done through a standard text search for the term `isochrone`. As we will see, this search is done in a SimDAL Repository that stores descriptions of codes and numerical simulations published in the VO.

Among the resulting projects, scientists pick one and go to the linked SimDAL Search service allowing them to search for data inside this particular project.

Search for a relevant dataset in a project: Scientists have some observations of a star with `Age=1.2 Gyr` and they want to know how some other theoretical quantities look like in this particular conditions. So, they query the service for simulation datasets having `Age=1.2 Gyr` (or in a range between 1 and 1.5 Gyr). This is done in a SimDAL Search service.

Among the matching datasets in the query result, they pick the `dataset id` of the datasets they want to further analyze and go to the linked SimDAL Data Access service allowing them to access the data associated to the dataset.

Access data of a dataset: Scientists would like to get the raw data of the simulation associated with the dataset. Thanks to the `dataset id`, they can download these data from the corresponding SimDAL Data Access service.

In some cases the dataset is very large and scientists are only interested in some subsets of the data. For instance, for isochrones, a dataset could provide a lot of columns with many different stellar properties and scientists are only interested in some of those properties (mass, temperature and luminosity of the star and this only for masses between 0.5 and 2 solar masses). In that case, scientists would only query for those properties with the appropriate restrictions and the answer would be a VO-Table with the corresponding data.

Other scientific use cases that led the design of SimDAL version 1.0 can be found in appendix A.

2.2 SimDAL

SimDAL is at the interface between end-users and services publishing theoretical data. It is intended to fulfill the following core requirements:

- *R1:* Discover sets of simulations published in the Virtual Observatory and find the associated services.
  Examples: Discover services publishing simulations of large structures in a cosmological context or services publishing synthetic stellar spectra.
- *R2:* Find particular simulations and their associated datasets according to specific input parameters or characteristics of the outputs (for e.g. statistics on the results).
  Example: In a service publishing synthetic stellar spectra, find models of stars with an effective temperature between 2000 and 3000 K.
- *R3:* Download simulation raw datasets or subsets extracted with a cut-out.
  Example: Download a data cube of a simulation of dark matter or download a theoretical spectrum in a specific wavelength range.

To fulfill these core use cases, three components are defined.

The first one is the *SimDAL Repository*. SimDAL Repositories store the reference metadata about codes, projects and services. These metadata are structured following the Simulation Data Model. SimDAL Repositories are used by end-users to discover services publishing theoretical data and by publishers to publish reference descriptions of codes or simulations. Like the IVOA registries, SimDAL Repositories are not intended to be implemented by all data publishers.
Data publishers must register in SimDAL Repositories the descriptions, as SimDM instances, of their protocols (codes) and projects if they want to be found in the VO architecture.

The second component is the *SimDAL Search*. Published simulations / datasets are characterized by a set of metadata. In most cases, these metadata are the values of the input parameters and / or other quantities that characterize the simulations (for example statistics on the results). The SimDAL Search component allows end-users to query on these quantities to discover simulations.

Finally the third component is the *SimDAL Data Access*. Numerical codes produce different kind of outputs. That can be data cubes as those produced by N-body and MHD simulation codes. Those cubes are generally stored in specific binary format (HDF5, VTK, ...). Outputs can also be synthetic spectra stored in FITS or VOTable format. Other ones can produce catalogs that can be stored on a file system in ASCII format or in relational databases. Thus, code outputs are very heterogeneous. The SimDAL Data Access API defines how to interact with such data to download all or a part of them. Since there is no standard data format for simulation outputs, SimDAL standard supports any format. Obviously, it is highly recommended to provide data in VO-compatible formats whenever possible (VO-Table, FITS) thus they can be manipulated by VO tools.

These three parts are independent (Fig. 2). Only a few data centers will implement SimDAL Repositories. Data publishers must implement a SimDAL Data Access component. If they need to provide a search in the parent simulations they have to implement a SimDAL Search component.

2.3 SimDAL and the Simulation Data Model

A standard description format is necessary to share descriptions of codes and their results between the different parties (scientific data producers, developers of diffusion systems and clients, etc.). In the Virtual Observatory context, this standard format is the XML serialization of the Simulation Data Model (SimDM - Lemson et al. (2012)). As a reminder, key elements of SimDM and how they are related are summarized in Fig. 3.

This document uses several SimDM concepts. Some important ones are:

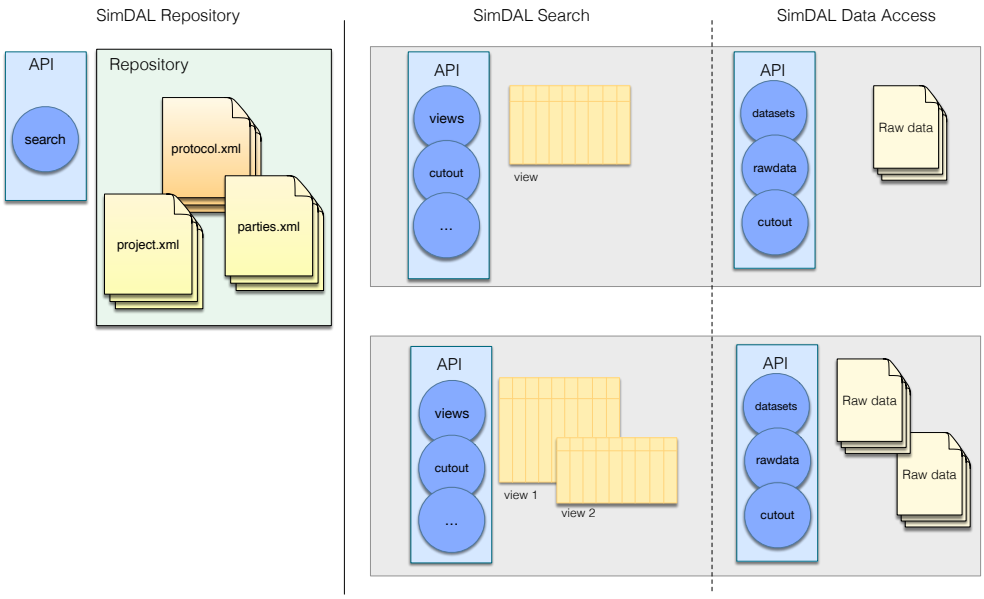

Figure 2: Overview of the SimDAL architecture. The three SimDAL components are represented on this Figure. On the left, one finds the SimDAL Repository that is used to discover SimDAL search services. On the right two SimDAL services (grey boxes) developed by two different publishers. Both of them are composed of a SimDAL Search component used to discover numerical models (for example by querying on the input parameters of codes) and a SimDAL Data Access component used to download the data. The service at the top is a simple service. This is the standard case. There is only one view in this SimDAL Search component. This view can be seen as a table with, in columns, the various input parameters of the runs and on each line a model / simulation. Queries on this table allow to find models (more precisely datasets IDs). With these IDs, it is possible to query the SimDAL Data Access component to retrieve the simulated data. The bottom service is a bit more complex. The code produces several datasets and the publisher implements several views to allow different ways to discover the data.

- *Project*: a project is an abstract entity that gathers all the metadata describing a scientific project (the code that has been used, the description of the simulations, ...), the inputs and the outputs (see 3.29 of the SimDM document (Lemson and al., 2012)).

  In the context of SimDM, elements of a project are mapped to classes holding metadata and relations to the other elements. The main SimDM classes used correspond to the very simplified process depicted in Fig. 4:

- *Dataset*: In general a simulation project generates a set of outputs. These outputs represent calculated quantities that can be attached to entities or generic objects (e.g star in astronomy, atoms in chemistry).

  In SimDM, the output of a simulation project is segmented in sets of homogeneous objects (i.e having the same object type, i.e the same set of attributes/properties). This corresponds to the OutputDataset SimDM class (3.21 of the SimDM document (Lemson and al., 2012)).

  In SimDM, these datasets can be characterized by statistics and property values that can be used to discover them.

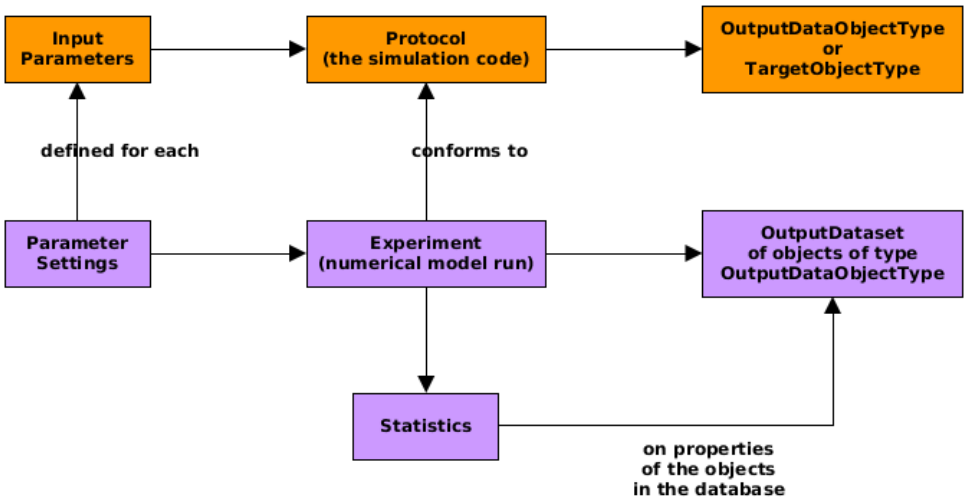

Figure 3: SimDM basic elements

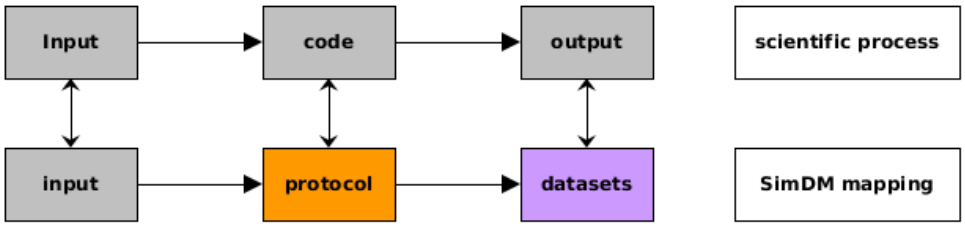

Figure 4: Simplified view of the main SimDM classes mapped to scientific process elements

The SimDM serialization is composed of several XML files (one per UML package) that, put together, describe a simulation project. In particular, the description for a whole simulation project is compound of:

- **XML serialization of SimDM Project package** (1 file) holding metadata about the project to which the simulations belong to.
- **XML serialization of SimDM Protocol package** (1 file) holding metadata about the protocol that the simulations followed (i.e code used to run the models)
- **XML serialization of SimDM Experiment package** (1 file per model run), holding the metadata about each model (code settings, computed values, etc. . .).
- **XML serialization of SimDM of Parties package** (1 file) holding the metadata about the people, team, research groups implied in the simulation project and their roles.

As explained below, project.xml, protocol.xml and parties.xml can be accessed from every SimDAL component whereas URIs of experiment.xml files are provided only in SimDAL Search components.

2.4 Registration in IVOA registries

SimDAL services may be discovered through IVOA registries queries. The common use case is 1) a client searches for a SimDAL Repository so that it will be able to search for projects of interest and access the detailed description of these projects and of associated codes. From there it will find the links towards the associated SimDAL Search and/or SimDAL Data Access services. Or 2), a client searches for a SimDAL Data Access service directly in the IVOA registries for direct access to the data. This case implies that the client has an apriori knowledge of who provides the service and which data is there. This can be obtained both by a previous visit to the service or by searching in a SimDAL Search service.

So, the three SimDAL components can be registered in the IVOA registries with the common basic metadata (name, description, publishers, etc...). The role of providing detailed metadata, numerical simulation specific search feature, and being the reference provider of codes descriptions using standard SimDM serialization format is carried out by the SimDAL repository.

3 SimDAL APIs

The three SimDAL components are exposed with APIs following a REST design (Hunter, 2013) that conforms to the DALI resource description (Dowler et al., 2013). As a result, the term *resource* will be used extensively in this document in the context of a RESTful architecture. General considerations on the SimDAL APIs design can be checked with Appendix B.

3.1 Resources

The list of resources for each of the three SimDAL components are presented below, with their status, required or not.
Publishers are free to name (set paths) resources whatever they wish; clients will find resource paths thanks to the VOSI-capabilities resource.
Resources with a leading * in the table below have their name (access URI) located in other resources responses instead of in the service capability resource (see Hypermedia Controls, Fowler (2010)). This allows more flexibility for URI definition. For example, a data publisher would prefer URIs as

http://`<SimDAL-search-uri>`/views/3

whereas another one would prefer

http://`<SimDAL-search-uri>`/views?viewid=3

3.2 Parameters

Unless stated otherwise, all resources parameters are not repeatable.

3.3 JSON

In this document, we refer to the ECMA-404 JSON standard, as defined in ECMA (2013)

3.4 Response format

The response format of the SimDAL APIs conforms to DAL guidelines for VOTable usage.

SimDAL Repository		
Resource type	Resource name	Required
DALI-sync	{search}	no
DALI-sync	{projects}	yes
DALI-sync	{protocols}	yes
VOSI-availability	/availability	yes
VOSI-capabilities	/capabilities	yes
SimDAL Search		
Resource type	Resource name	Required
DALI-sync	{experiments}	no
DALI-sync	{views}	yes
DALI-sync	∗{fields}	no
DALI-sync	∗{cutouts}	yes
DALI-async	∗{async/cutouts}	no
DALI-sync	∗{cutouts-preview}	no
VOSI-availability	/availability	yes
VOSI-capabilities	/capabilities	yes
SimDAL Data Access		
Resource type	Resource name	Required
DALI-sync	{datasets}	yes
DALI-sync	∗{cutouts}	no
DALI-async	∗{async/cutouts}	no
DALI-sync	∗{fields}	no
DALI-sync	∗{rawdata}	no
VOSI-availability	/availability	yes
VOSI-capabilities	/capabilities	yes

Table 1: Resources defined for each of the SimDAL components with their mandatory status

Error The SimDAL API extends the DALI specification in case of service call error. The returned VOTable contains, in addition to the DALI error INFO element (which contains a human readable error message up to the implementer), a table containing rows consisting of a string-valued column `error_msg` and an integer-valued column `error_code`.

```
<RESOURCE type="results">
  <INFO name="QUERY_STATUS" value="ERROR"/>
  ...
  <FIELD name="error_msg" datatype="char" arraysize="*"></FIELD>
  <FIELD name="error_code" datatype="int"></FIELD>

  <TABLE>...</TABLE>
</RESOURCE>
```

Result Unless explicitly indicated, the result data in a resource representation is located in the VOTable TABLE element named `results` inside the RESOURCE element of type `results`.

Also, if not specified, *FIELD f1* means that *f1* is the `name` of the FIELD element. Of course when a FIELD needs to be referenced an `ID` attribute must be defined too, following the VOTable standard.

3.5 The 'links' additional table in VOTable responses

The SimDAL standard uses a special VOTable TABLE element, named `links` in its responses. This table is required to:

- provide links towards resources. Presently, there is no standard for the URI/links form (query string, path segments, ...) of the resources described in the SimDAL protocol. This is why it is necessary to have a way to get these non-standard, publisher specific URIs.

  For example, publisher A could have defined its SimDAL search views child resources as:

  http://<uri>/my−views/for−id/3

  whereas publisher B would have implemented

  http://<uri-b>/views?id=3

  There is no standard URI for resources but a standard way to get the URI for the resources (as done in almost all the recent DAL standards, see DataLink (Dowler et al., 2015), for example). The `links` table is the way to get these specific URIs.

  Hypermedia Controls (Fowler, 2010) done this way is one of the traditional REST best practice so far.

- support mirroring feature, providing backup URIs in case the main resource would become unavailable, or use them as load balancing.

- make the standard easily extendable by just defining more `link-rel` values in future versions of the standard.

The `links` table has the following schema (FIELDs `name`s):

- a `link-rel` FIELD holding the semantic attached with what is pointed by the link
- a `link-uri` FIELD holding the actual URI

Link with other tables One must know to which TABLE/FIELD the elements of the `links` table refer to. This is achieved with a relational-like approach, as suggested in the VOTable 1.3 standard (section 4.1 of VO-Table, Ochsenbein et al. (2013)).

In the following example, the response VOTable contains 2 tables, the `results` and the `links` tables. We use the grouping feature of VOTable to define that the field `view` in the `links` table refers to the field `ident` of the `results` table.

```
...
<TABLE name="results">
  <FIELD ID="ident" name="ident" datatype="char" arraysize="*"></FIELD>
  <FIELD name="created" datatype="char" arraysize="*"></FIELD>
   ...

<TABLE name="links">
  <FIELD name="view" datatype="char" arraysize="*"></FIELD>
  <FIELD name="link-rel" datatype="char" arraysize="*"></FIELD>
  <FIELD name="link-uri" datatype="char" arraysize="*"></FIELD>

  <GROUP name="foreign_key" ref="results">
```

```
   <GROUP>
     <PARAM name="local_field" value="view"/>
     <FIELDRef ref="ident"/>
   </GROUP>
  </GROUP>
...
```

Thus, one defines the link between the table `links` and other tables of the VOTable by defining:

- a GROUP with the special name `foreign_key` and a ref attribute set to the foreign table name
- a child GROUP with a FIELDRef's `ref` set to the FIELD ID referenced within the foreign table in the VOTable
  - a PARAM with name `local_field` set to the *links* table field pointing to the foreign table.
  - a FIELDRef with `ref` to the foreign table referenced field.

Note: Here we use only one field to reference a foreign table, but it is possible to do the link with several fields, like a multi-attributes primary key in relational databases. In this case, the GROUP with name `foreign_key` would have as many child groups as attributes composing the primary key.

3.6 Pagination

Because some resources can represent very large collections, a pagination mechanism must be defined.

Easy handling of collections for the client is achieved by doing the maximum of processing server-side to deal with the pagination. In particular, for collections resources, the response contains the links for direct access to the next and/or previous page of items.

The response can contain links towards the next page, the previous one, both or none in each data table. The special `content-role` for the associated `LINK` elements are:

- `next_page`
- `previous_page`

Given the nature (size in particular) of the data in SimDAL, collections are often considered as infinite streams and, in this case, we consider only one operation on it: go forward (i.e give me the next page, until infinite...).

The page size is defined automatically by the server and provided to the client through a special `PARAM` in the `results RESOURCE` of a result VOTABLE with the special name `page_size`.

Example:

```
<?xml version="1.0" ?>
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
        xmlns:stc="http://www.ivoa.net/xml/STC/v1.30"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <RESOURCE type="results">
    <PARAM name="page_size" value="10"/>
    <TABLE name="v1 cut-out">
      <LINK content-role="next_page"
           href="http://<uri>/api/simdal/fd5924f9/views/v1/cutouts/3?
```

```
                limit=10&amp;offset=10"/>
    <FIELD name="c/output_dataset"/>
    <FIELD name="c/gas_pressure_input"/>
    <FIELD name="c/radm_ini"/>
    ...
```

The lifetime of the page resources used for pagination is managed by the server so that the client need not worrying about it. The server must provide sufficient lifetime for interactive browsing of the pages in the user session.

A client knows that a collection resource supports pagination by looking in the resource representation (a VOTable) if there is a pagination special LINK element. Thus, a service may implement the pagination but decide not to use it for some collections while using it for others.

Finally, since we have interactive use cases in mind here, the pagination system only makes sense for sync resources. Async resources already deal with the time generation/data size through their *async* characteristic.

Pagination link inference One should not infer a way to access directly a particular page from the URI pattern of the pages link value. Indeed, nothing is standard here and it is up to the service publisher to define its own internal implementation.

3.7 VOSI-availability

A SimDAL service must have a VOSI-availability resource.

3.8 VOSI-capabilities

A standalone SimDAL service must have a VOSI-capabilities resource. The standardID for the resources are:

```
ivo://ivoa.net/std/SimDALRepository#{<resource-term>}-1.0
ivo://ivoa.net/std/SimDALSearch#{<resource-term>}-1.0
ivo://ivoa.net/std/SimDALDataAccess#{<resource-term>}-1.0
```

where <resource-term> is taken from the *Resource name* column in Tab. 1. Example:

```
ivo://ivoa.net/std/SimDALSearch#views-1.0
```

3.9 Time

All timestamps are returned in UTC date/time following the ISO-8601 standard (YYYY-MM-DDThh:mm:ss optionally followed by a decimal point and fractions of seconds).

In several parts of the standard, a VOTable FIELD named `created` is used. While often optional, it is a good idea to provide it, in particular for batch application which would update the database for a proxy application. The proxy application could use this timestamp information while periodically polling for updates to know if it has the last version or not.

3.10 Format of the examples in this document

In the examples below, we assume the following service URIs for the SimDAL components API implementation:
- SimDAL Repository at `http://<simrepo uri>`
- SimDAL Search at `http://<simsearch uri>`
- SimDAL Data Access at `http://<simdal uri>`

4 SimDAL Repository

A SimDAL Repository is a repository to store and discover the reference descriptions of codes and numerical simulation projects published in the VO. Publishers publish these descriptions in SimDAL Repositories using the SimDM XML serialization (protocol.xml, project.xml and parties.xml).

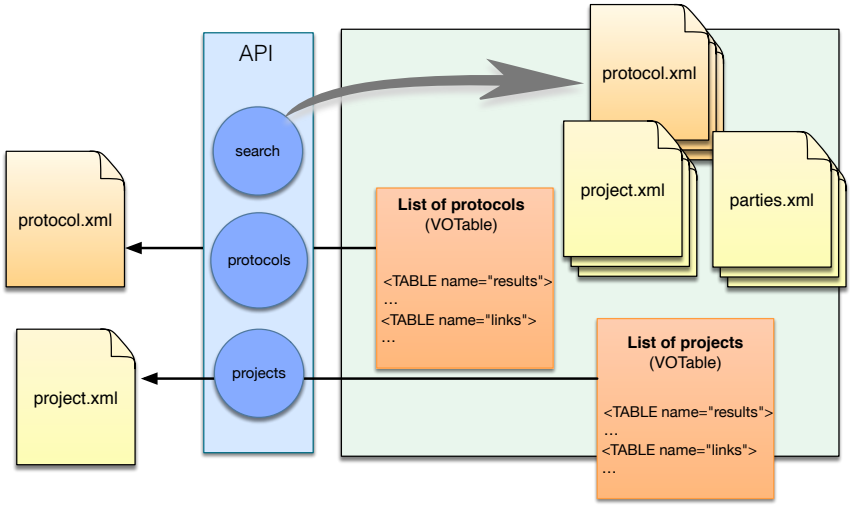

Figure 5: SimDAL Repository. The SimDAL Repository stores the descriptions of theoretical projects (codes and results of these codes) published in the VO. Its three resources allows to discover these projects and, eventually, retrieve their SimDM XML serializations.

The API exposed by a SimDAL Repository aims to:

- search in the repository for metadata on projects, protocols and parties. This search is done on concepts that are found in the text attributes of the SimDM classes (Examples: N-body, spectrum, proton density, star, ...).
- retrieve / download the SimDM XML serializations.

In a SimDAL Repository, resources (projects, protocols, parties) are identified by a repository local identifier, named a `ident`. The definition and management of this identifier is up to the service provider.

4.1 {search}

The *{search}* resource is used to perform fulltext search among all XML documents in the repository.

This resource should implement the pagination API.

Parameter

- `q`: the text pattern to search for. By default, the service handles plain strings with the optional `*` wildcard, meaning "any character".

  Any extra feature of the search logic is up to the publisher (auto wild card insertion, case sensitiveness, spellchecking etc...). In this case, the documentation of such features is the publisher responsability.

- `att`: the attribute whose value is to be matched by `q`. The default (if not provided) is to try to match `q` against all the available attributes.

Example searching for `n(h2)` in the attribute `name` only:

```
http://<simrepo uri>/search?q=n(h2)&att=name
```

Response schema The response schema of the *results* table is (values of FIELDs `name`s attributes):

- `publisher`: the IVOA identifier of the publisher of the project the document having the match belongs to.
- `project`: the project `ident` (see above) which identifies the project in the repository.
- `document`: the document containing the matching text.
- `match`: the matching value among the document attributes values.
- `attribute`: The attribute corresponding to the matching value
- `rank`: The relevance of the match, greater is better. This field is optional.

In case of no match, the returned VOTable `<tabledata>` element is empty.

In the example, the query returns:

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
       xmlns:stc="http://www.ivoa.net/xml/STC/v1.30"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RESOURCE type="results">
   <INFO name="QUERY_STATUS" value="OK"/>
   <TABLE name="results">
     <FIELD datatype="char" arraysize="*" ID="document" name="document"/>
     <FIELD datatype="char" arraysize="*" name="match"/>
     <FIELD datatype="char" arraysize="*" name="attribute"/>
     <FIELD datatype="char" arraysize="*" name="publisher"/>
     <FIELD datatype="char" arraysize="*" name="project"/>
     <FIELD datatype="char" arraysize="*" name="rank"/>
     <DATA>
       <TABLEDATA>
         <TR>
           <TD>cd_h2</TD>
           <TD>N(H2)</TD>
           <TD>name</TD>
           <TD>ivo://ism.obspm</TD>
```

```
        <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
        <TD>0.0607927</TD>
      </TR>
      <TR>
        <TD>n_h2</TD>
        <TD>n(H2)</TD>
        <TD>name</TD>
        <TD>ivo://ism.obspm</TD>
        <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
        <TD>0.0607927</TD>
      </TR>

      ...

    </TABLEDATA>
  </DATA>
</TABLE>
<TABLE name="documents">
  <FIELD datatype="char" arraysize="*" name="document"/>
  <FIELD datatype="char" arraysize="*" name="attribute"/>
  <FIELD datatype="char" arraysize="*" name="value"/>
  <LINK content-role="documents_schema" href="simdal-repo/documents/schema.xml">
  <GROUP name="foreign_key" ref="results">
    <GROUP>
      <PARAM name="local_field" value="document"/>
      <FIELDRef ref="document"/>
    </GROUP>
  </GROUP>
  <DATA>
    <TABLEDATA>
      <TR>
        <TD>cd_h2</TD>
        <TD>short_label</TD>
        <TD>ColumnDensityOfUFHFLCQGNIYNRP-UHFFFAOYSA-N</TD>
      </TR>
      <TR>
        <TD>cd_h2</TD>
        <TD>doc</TD>
        <TD>Column density</TD>
      </TR>

      ...

      <TR>
        <TD>n_h2</TD>
        <TD>datatype</TD>
        <TD>real</TD>
      </TR>
      <TR>
        <TD>n_h2</TD>
        <TD>publisherdid</TD>
        <TD>n_h2</TD>
```

```
        </TR>
        <TR>
          <TD>n_h2</TD>
          <TD>label</TD>
          <TD>
            http://purl.obspm.fr/vocab/PhysicalQuantities/AbundanceOfH2
          </TD>
        </TR>

      </TABLEDATA>
    </DATA>
</TABLE>
</RESOURCE>
</VOTABLE>
```

This describes the quantities involved in the simulation dataset.

Abstract Data Type In the *{search}* resource point of view, a project is a set of *documents*. Each *document* has a set of attributes and associated values.

For example, the project `PDR_152_1528_ch1509_iso_mathis` has only one document with the following (attribute, value) couples:

```
utype:          simdm://SimDM:/resource/protocol/Simulator
publisherdid:   PDR_152_1528
updated:        2015-12-03T15:09:31.148904
description:    The Meudon PDR code is an astrochemical code designed to
                simulate the interaction between neutral interstellar gas
                and radiation field under a broad range of conditions (Le
                Petit et al. 2006). The code considers a stationary 1D
                slab of dust and gas illuminated by an interstellar
                radiation field.
                [...]
```

This way of modeling the data allows for extensibility of future use cases in the search feature.

The *{search}* resource represents all the documents among all the available projects having attributes values matching the query `q`.

Note: One can also set the parameter `att`, which is `*` by default (meaning any attribute), to restrict on which document attribute the query `q` applies.

documents schema Of course one must have a way to know what the available attributes to describe a document are (`utype`, `publisherdid`, etc... in the example above). This is done by providing a *documents schema* describing the available attributes describing documents. This schema is a VOTable having a table with the following FIELDs:

- `attribute`: the name of the attribute
- `doc`: a documentation of what the attribute refer to/is.

Example:

```
<RESOURCE type="results">
 <TABLE name="schema">
   <FIELD name="attribute" datatype="char" arraysize="*"></FIELD>
   <FIELD name="doc"    datatype="char" arraysize="*"></FIELD>
   <DATA>
     <TABLEDATA>
       <TR>
         <TD>created</TD>
         <TD>the creation date of the document</TD>
       </TR>
       <TR>
         <TD>utype</TD>
         <TD>the utype of the document</TD>
       </TR>
       <TR>
         <TD>description</TD>
         <TD>The description attribute in SimDM class having it</TD>
       </TR>
       ...
     </TABLEDATA>
   </DATA>
 </TABLE>
 ...
```

The link to reach the *documents schema* is given in the VOTable representation of the *{search}* resource as a *documents* TABLE LINK with the special `content-role` *documents_schema*.
Providing a document schema is not mandatory, in this case the metadata schema has to be communicated by non-standard means.

4.2 {projects}

The aim of this resource is to allow a client to:

- browse through all the theoretical projects published in the repository without knowing exactly what to search for (i.e. using the *{search}* resource)
- get the full standardized SimDM XML serialization of a project package to fulfill whatever specific use case a client may have.

The {projects} resource represents a list of the available projects and a set of associated links.

The *link*s can be in particular those toward the SimDM XML serialization of the project but more importanly the links towards associated services like SimDAL Search and/or SimDAL Data Access.

Parameter

- `project`: returns the metadata and links associated only with the project having as identifier `project`. If not provided, the collection resource is returned, with all the available projects.

Example:

```
http://<projects resource URI>?project=ism.obspm/DIFF51_n_ERLH_isot
```

link-rel	SimDAL Component
simdal/search	SimDAL Search
simdal/data_access	SimDAL Data Access
simdm/project/xml	SimDM XML serialization of project
simdm/parties/xml	SimDM XML serialization of parties

Table 2: Standard values for the link-rel attribute

Response schema The response schema of the *results* table is (FIELDs `name`s):

- `ident`: the identifier of the project inside the SimDAL Repository service.
- `created`: the timestamp of registration of the project in the SimDAL Repository service.
- `publisher`: the identifier of the publisher inside the SimDAL Repository service.
- `name`: the plain text name of the project
- `doc`: the plain text documentation/description of the project. Optional.

links The resources/services associated with every project are listed in the *links* table, with the following standardized `link-rel`:

Example of response:

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
       xmlns:stc="http://www.ivoa.net/xml/STC/v1.30"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RESOURCE type="results">
   <INFO name="QUERY_STATUS" value="ok"/>
   <TABLE name="results">
     <FIELD arraysize="*" datatype="char" ID="ident" name="ident"/>
     <FIELD arraysize="*" datatype="char" name="created"/>
     <FIELD arraysize="*" datatype="char" name="publisher"/>
     <FIELD arraysize="*" datatype="char" name="name"/>
     <FIELD arraysize="*" datatype="char" name="doc"/>
     <DATA>
       <TABLEDATA>
         <TR>
           <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
           <TD>2016-04-21 12:29:28.821136</TD>
           <TD>ivo://ism.obspm</TD>
           <TD>Grid of diffuse clouds at constant density</TD>
           <TD>
           This grid of isochoric PDR 1.5.2 models covers diffuse
             clouds conditions. Explored parameters are proton density,
             UV field intensity and size of the clouds. The full grid
             contains 924 models. Parameters range are: n from 10 to
             1000 cm-3, radiation field from 0.5 and 10 times the ISRF,
             and size (AV) from 0.2 to 3 mag. Both sides of the clouds
             are illuminated by the same radiation field .
              [...]
```

```
          </TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
  <TABLE name="links">
    <FIELD arraysize="*" datatype="char" name="project"/>
    <FIELD arraysize="*" datatype="char" name="link-rel"/>
    <FIELD arraysize="*" datatype="char" name="link-uri"/>
    <GROUP name="foreign_key" ref="results">
      <GROUP>
        <PARAM name="local_field" value="project"/>
        <FIELDRef ref="ident"/>
      </GROUP>
    </GROUP>
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
          <TD>simdm/project/xml</TD>
          <TD>
            http://127.0.0.1:3435/api/simdal/DIFF51_n_ERLH_isot/xml
          </TD>
        </TR>
        <TR>
          <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
          <TD>simdal/search</TD>
          <TD>
            http://localhost:3132/api/simdal/440d0ed4-b034
          </TD>
        </TR>
        <TR>
          <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
          <TD>simdal/data_access</TD>
          <TD>
            http://127.0.0.1:3435/api/simdal/DIFF51_n_ERLH_isot
          </TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
 </RESOURCE>
</VOTABLE>
```

4.3 {protocols}

The aim of this resource is to allow a client to:

- browse through all the protocols used by the published projects in the repository without knowing exactly what to search for (i.e. using the *{search}* resource)

link-rel	SimDAL Component
simdm/protocol/xml	SimDM XML serialization of protocol
simdm/parties/xml	SimDM XML serialization of parties

Table 3: Standard values for the link-rel attribute

- get the full standardized SimDM XML serialization of a protocol package to fulfill whatever specific use case a client may have.

The {protocols} resource represents a list of the available protocols and a set of associated links.

The *link*s can be in particular those toward the SimDM XML serialization of the protocol but more importanly the links towards associated services like SimDAL Search and/or SimDAL Data Access.

Parameters

- `project`: filters the protocols defined in the project having the specified `project` ident. This parameter is optional. If not provided returns all the protocols from all the projects.

```
http://<protocols resource URI>?project=ism.obspm/DIFF51_n_ERLH_isot
```

Response schema The response schema of the *results* table is:

- `ident`: the identifier of the project inside the SimDAL Repository service.
- `created`: the timestamp of registration of the project in the SimDAL Repository service.
- `publisher`: the identifier of the publisher inside the SimDAL Repository service.
- `name`: the plain text name of the project
- `doc`: the plain text documentation/description of the project. Optional.
- `project`: the identifier (inside the SimDAL Repository service) of the project in which the protocol is defined.

links The resources/services associated with every project are listed in the *links* table, with the following standardized `link-rel`:

Example of response:

```
...
<RESOURCE type="results">
  <INFO name="QUERY_STATUS" value="OK"/>
  <PARAMETER name="project" value="ism.obspm/DIFF51_n_ERLH_isot"/>
   ...
  <TABLE name="results">
    <FIELD ID="ident" name="ident" datatype="char" arraysize="*"></FIELD>
    <FIELD name="created" datatype="char" arraysize="*"></FIELD>
    <FIELD name="publisher" datatype="char" arraysize="*"></FIELD>
    <FIELD name="name"    datatype="char" arraysize="*"></FIELD>
```

```
  <FIELD name="doc"     datatype="char" arraysize="*"></FIELD>
  <FIELD name="project" datatype="char" arraysize="*"></FIELD>
  <DATA>
    <TABLEDATA>
      <TR>
        <TD>ism.obspm/pdr_152_r4533</TD>
        <TD>2016-04-21 12:29:28.821136</TD>
        <TD>ivo://ism.obspm</TD>
        <TD>PDR 1.5.2 rev 4533</TD>
        <TD>PDR Meudon code version 1.5.2</TD>
        <TD>ism.obspm/DIFF51_n_ERLH_isot</TD>
      </TR>
      ...
    </TABLEDATA>
  </DATA>
</TABLE>

<TABLE name="links">
  <FIELD name="protocol" datatype="char" arraysize="*"></FIELD>
  <FIELD name="link-rel" datatype="char" arraysize="*"></FIELD>
  <FIELD name="link-uri" datatype="char" arraysize="*"></FIELD>

  <GROUP name="foreign_key" ref="results">
    <GROUP>
      <PARAM name="local_field" value="protocol"/>
      <FIELDRef ref="ident"/>
    </GROUP>
  </GROUP>

  <DATA>
    <TABLEDATA>
      <TR>
        <TD>ism.obspm/pdr_152_r4533</TD>
        <TD>simdm/protocol/XML</TD>
        <TD>http://link_toward_protocol_xml_serialization</TD>
      </TR>
      ...
    </TABLEDATA>
  </DATA>
</TABLE>

</RESOURCE>
...
```

Note: The mechanism of `link-rel` here is intended to allow future updates of the standard at low cost by defining other standard values. Also, one is free to use any custom values for its own use (for example if one provides a client that understands that specific values)

5 SimDAL Search

SimDAL Search components are intended to search for datasets. URIs of SimDAL Search services are usually known after a search in a SimDAL Repository.

To allow searches for simulations, a SimDAL Search service presents the queryable data (input values, statistical values, property values, ...) to clients in one or several *views*. These *views* can be seen as relations or tables. They are called *views* because they are focused views on subsets of a SimDM instance.

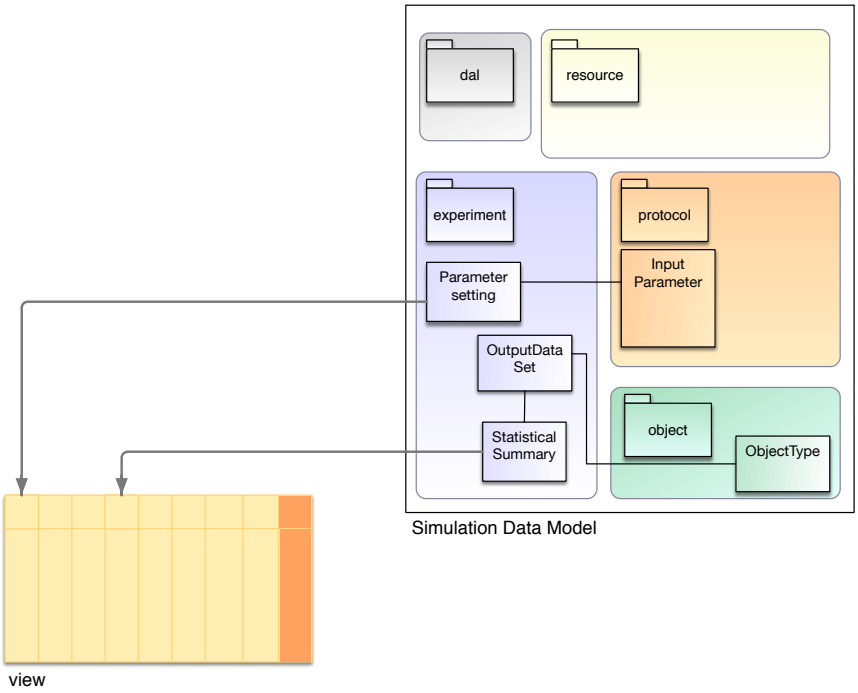

Figure 6: SimDAL Search and SimDM Experiments package. In SimDM, a run is characterized by its input parameters and its output (SimDM OutputDataSets) by statistics. To make datasets discoverable, publishers create views with columns representing the input parameters and the statistics, with rows representing the various datasets. Querying views gives access to datasets IDs, stored in a column of the *view* (orange column in the Figure).

A SimDAL Search service can contain one or several *views*. Since *views* are used to discover datasets, at least one *view* per ObjectType will be usually defined. For example, to publish results of a code that produces a 3D structure of a star (first ObjectType) plus a synthetic emission spectrum for this star (second ObjectType), two *views* could be defined, the first one to discover the IDs of datasets associated to the 3D structures, and the second one to discover the IDs of datasets associated to the spectra.

Publishers can add any other *views* they wish if this allows them to satisfy other use cases they are interested to provide. For instance, a single view can be defined including properties of several different ObjectType together. This freedom allows for good extensibility while future use cases are coming. As explained in the document, *views* fields are usually InputParameters, StatisticalValues and/or PropertyValues defined in SimDM. They are tagged in the view schema by SimDM utypes. Views fields that would not be associated to any SimDM classes, do not have SimDM utypes in their schema.

These *views* are often so big that they cannot be retrieved through the network and, therefore, they must be kept server-side. The only way to access them is to define subsets (cut-outs) first. A cut-out is defined by either filtering the columns (the fields) or the rows (by applying constraints on fields values) of the *view*. That is what would be done, for e.g., when performing a SQL query on a single

flat table, SELECT filters on columns, and WHERE filters on rows.

A schema must be provided by the service for each defined *view* (see below), so that the available fields (to ask cut-out on) and associated semantic can be known by the client.

SimDAL does not define how metadata in *views* are stored and managed server-side because, depending on the data (volume and heterogeneity), different publishers will have to implement different technological solutions (relational database, document database, graph database etc). The only standard thing is this abstract element, called *views*, (may be imagined as *giant* server-side VOTable). This problem of the potential large-size of *views* is a major element of the SimDAL Search design. It aims at untying the standard and the implementation details.

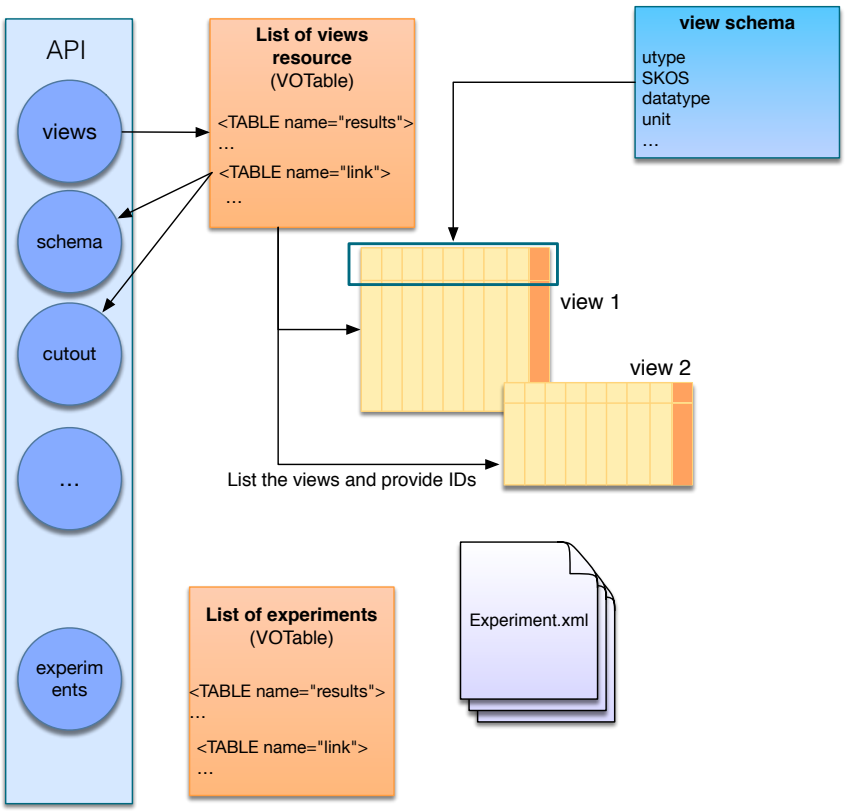

Figure 7: Architecture of the SimDAL Search component.

Typical SimDAL Search workflow

1. use the *{views}* resource to get the available *views* and choose one.
2. get the *view* schema that provides the list of fields that can be used to do a cut-out.
3. make a cut-out by submitting a query to the *{cutouts}* resource.
4. find a dataset ID and the URL of the associated SimDAL Data Access.

5. go to the SimDAL Data Access component and access data.

Some additional considerations about the SimDAL Search API design can be found in appendix B.

5.1 SimDAL Search API design

As stated above, the SimDAL Search service allows queries on *view*s of a numerical simulations project. Each *view* is described by a schema.

View schema In order to query a *view* of a SimDAL Search service, the *view* schema provides a description of the FIELDs of the view. That is, the FIELDs that can be used with the *{cutouts}* resource to make a search. In the *view* schema each column/field is described by an ID (mandatory), which is used in query's `select`, `where` and `orderby` parameters, and optional attributes. Interoperability requires that elements contained in *view*s be described in a standard way, i.e points towards classes of a standard data model (here we only describe the case of the Simulation Data Model). In this context, the optional attributes mentioned above are name, utype, unit, datatype and SKOS concept. In client-server exchanges, *view schema*s are serialized as VOTable headers where each FIELD has these attributes and a LINK element for the associated SKOS concept with `content-role="type"` as defined in the VOTable standard.

Note: It is very common to use the FIELD `name` attribute for the *human readable name* and `ID` for its *machine readable name* because the common use case of schemas is to provide the client with user friendly columns names in a UI form while allowing it to submit queries with machine friendly names (the `ID`s).

Note on semantic: Data model linking through utype allows to reach a certain level of interoperability. Nevertheless, SimDM is an abstract model (or a meta model) so utypes provide information on SimDM classes but not on the meaning of the concepts (For example, if a view column corresponds to an input parameter that is a speed, providing the utype will only allow the client to know that this column is an InputParameter but not that the related quantity is speed nor which type of speed it is). To enforce interoperability on the semantic aspect, we must refer to a standard vocabulary where shared meaning of given concepts are defined. This is done through the *label* attribute of SimDM elements that holds URI of SKOS concepts (see example below).
To fulfill the general use cases, it is recommended to have at least one *view* (more precisely one per ObjectType), with fields corresponding to ParameterSettings (i.e the input of a simulation) and/or fields corresponding to values or statistical values of properties of OutputData (PropertyValue and/or StatisticalSummary in SimDM) plus a field holding the OutputData IDs (i.e. datasets IDs).

Example of a schema for a basic *view*. Note the use of utypes and SKOS to identify the content of the *view* fields.

```
<VOTABLE XMLns="http://www.ivoa.net/XML/VOTable/v1.3">
 <RESOURCE type="results">
  <TABLE name="my view schema">
   <FIELD name="Proton density"
          ID="c:densh"
          utype="simdm:/resource/experiment/ParameterSetting.numericValue.value"
          datatype="float" unit="cm-3">
    <LINK content-role="type"
          href="http://purl.obspm.fr/vocab/PhysicalQuantities/ProtonDensity"/>
   </FIELD>
```

```
    <FIELD name="Max Grain temperature for silicates"
          ID="c:max:temperature_grain_silicates"
          utype="simdm:/resource/experiment/StatisticalSummary.numericValue.value"
          datatype="float" unit="K">
      <VALUES>
        <MIN value="16.9"/>
        <MAX value="77.2" inclusive="yes"/>
      </VALUES>

      <LINK content-role="type"
           href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
    </FIELD>

    <FIELD name="Grains dataset id"
          ID="c:grains_outputdataset_pubdid"
          utype="simdm:/resource/experiment/OutputDataset.publisherdid"
          datatype="string"/>

  </TABLE>
 </RESOURCE>
</VOTABLE>
```

Link between views It can be very useful to allow links between *view*s, in particular to enforce predictability about how an object found in one *view* v1 can be referenced in another *view* v2 where it is also present.

This can be seen as a *foreign key* in relational database vocabulary.

The mechanism is simple: one must provide the URI towards the (SimDAL Search) resources providing the target *view* along with the mapping between local field and target *view* fields.

The information is provided as a VOTable GROUP element in the source table definition with a PARAM having the special value `foreign_key` with name `view:elem`.
The special target resource URI `"."` is used in the case the target *view* is handled by the same resource than the source *view*.

```
 <GROUP>
   <PARAM name="view:elem" value="foreign_key"/>
   <PARAM name="resource" value="."/>
   <PARAM name="view" value="v1"

   <PARAM name="local_field" value="inputdataset"/>
   <PARAM name="dest_field" value="dataset"/>

 </GROUP>
```

Query Language A basic query language is defined to query a *view*. This language is a tiny subset of SQL for flat tables without any referential integrity. Thus a query is composed of 3 parameters: `select`, `where`, and `orderby`.

- `select` is a list of the fields to return the value of
- `where` is a list of `constraints` on fields to filter the objects of the *view*.
- `orderby` is a list of 2 elements (field to order on, order direction). *Order direction* is either `asc` or `desc`.

The `select`, `where` and `orderby` parameters are JSON elements passed to the service as a POST http query.

`constraint` is a JSON object with the following keys:

- `:att`: the field to constrain
- `:op`: the operator defining the nature of the constraint. Possible values are: `>, <, =, >=, <=, !=`
- `:val`: the value constraining the field

This approach has the following advantages:

- Not limited by a potential maximum GET URL length for some browsers/servers
- Not limited by the constraints about the data structures / allowed characters in a GET query string
- Allows JSON document as service input, i.e nice and clean data structures
- Enable future extension, for example to support basic arithmetic operations on fields

5.2 {views}

The *{views}* resource provides the list of the available *views*, with their metadata and a set of associated links (e.g. link towards the *view* schema). This resource is a collection and its representation is a VOTable listing.

view metadata The metadata of the *view* are the FIELDs of the table named `results`. Only one field carrying metadata is mandatory, the identifier of the *view*, called `ident`. Along with this mandatory metadata, the publisher can add as many additional metadata as he wants as additional FIELDs elements. Optional metadata can be for example: `created`, `objecttype`, `protocol`, ...

Example of *{views}* resource for one view called `v2`:

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RESOURCE type="results">
   <INFO name="QUERY_STATUS" value="OK"/>
   <TABLE name="results">
     <FIELD arraysize="*" datatype="char" name="ident" ID="ident"/>
     <FIELD arraysize="*" datatype="char" name="created"/>
     <FIELD arraysize="*" datatype="char" name="objecttype"/>
     <FIELD arraysize="*" datatype="char" name="protocol"/>
     <DATA>
       <TABLEDATA>
         <TR>
```

```
          <TD>v2</TD>
          <TD>2016-03-21 12:30:39.324347</TD>
          <TD>cloud</TD>
          <TD>PDR_152_1598_ch1512_iso_mathis</TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
  <TABLE name="links">
    <FIELD arraysize="*" datatype="char" name="view"/>
    <FIELD arraysize="*" datatype="char" name="link-rel"/>
    <FIELD arraysize="*" datatype="char" name="link-uri"/>
    <GROUP name="foreign_key" ref="results">
      <GROUP>
        <PARAM name="local_field" value="view"/>
        <FIELDRef ref="ident"/>
      </GROUP>
    </GROUP>
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>v2</TD>
          <TD>view/schema</TD>
          <TD>
            http://localhost:3132/api/simdal/440d0ed4-b034/views/v2/schema
          </TD>
        </TR>
        <TR>
          <TD>v2</TD>
          <TD>view/cutouts</TD>
          <TD>
            http://localhost:3132/api/simdal/440d0ed4-b034/views/v2/cutouts
          </TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
 </RESOURCE>
</VOTABLE>
```

Standard links associated with a view A set of standard links can be associated with a *view*.

Link towards SimDAL Data Access component The link towards the SimDAL Data Access component associated with the datasets found in a given *view* is defined as a LINK VOTable element child of the FIELDs elements carrying the dataset ID (with utype `SimDM:/resource/experiment/OutputDataset.publisherDID`) in the *{cutouts}* result VOTable.

This is done this way so it is possible for a publisher to have several different SimDAL Data Access services for, for example, each ObjectType of the datasets, or for each partitions of datasets if the publisher has decided to segment his datasets internally in sub-datasets (for example if a dataset content

link-rel	description
`view/schema`	URI of the VOTable representation of the view schema
`view/cutouts`	URI of the *cutouts* resource
`view/async/cutouts`	URI of the *async/cutouts* resource
`view/fields`	URI of the *fields* resource
`view/cutouts-preview`	URI of the *cutouts-preview* resource
`simdm/protocol/xml`	URI of the associated SimDM protocol XML serializations
`simdm/project/xml`	URI of the associated SimDM project XML serialization
`simdm/parties/xml`	URI of the associated SimDM XML serializations of parties
`view/reference`	URI of a reference paper, website, or material related to the research work present in the view

Table 4: Standard values for the link-rel FIELD. The links below the horizontal line are optional, at least one of *view/cutouts* and *view/async/cutouts* is required. The special link-rel *view/reference* allows citable references to be explicitly linked to the view.

is shared across several sites and he sets up a SimDAL Data Access service on each one).

The LINK element has a special `content-role` equals to `view/data_access`.

The location of a SimDAL Data Access can thus be found both in the *links* table in the *{views}* VOTable representation and as a LINK element child of a FIELD element in a *{cutouts}* results VOTable. In case both are present, the most specific, i.e the LINK element in *{cutouts}* results VOTable, takes priority.

Note: One knows which FIELD carries the dataset ID before designing a query by searching the view schema for the FIELD having `utype=SimDM:/resource/experiment/OutputDataset.publisherDID`

Link towards XML serializations In order to link the *view* with a standard simulation project description (i.e an IVOA SimDM serialization instance), a link towards the project.xml, protocol.xml and parties.xml documents describing the project in which the *view* is defined should be added in the `links` table.

5.3 {experiments}

The goal of this resource is to provide the full, standardized metadata, describing an experiment. It allows clients to fulfill all the use cases they could have that are not addressed by the available *view*s provided by the publisher.

The {experiments} resource represents a list of the available experiments and a set of associated links towards the SimDM XML serializations associated with an experiment (experiment, protocol, project and/or parties serializations).

Example of {experiments} resource:

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
       xmlns:stc="http://www.ivoa.net/xml/STC/v1.30"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RESOURCE type="results">
```

```
<INFO name="QUERY_STATUS" value="ok"/>
<TABLE name="results">
  <FIELD arraysize="*" datatype="char" ID="ident" name="ident"/>
  <FIELD arraysize="*" datatype="char" name="created"/>
  <FIELD arraysize="*" datatype="char" name="publisher"/>
  <FIELD arraysize="*" datatype="char" name="name"/>
  <FIELD arraysize="*" datatype="char" name="doc"/>
  <DATA>
    <TABLEDATA>
      <TR>
        <TD>ism.obspm/DIFF51_r3e0A3e0d7e2_s_20</TD>
        <TD>2016−04−21 12:29:28.821136</TD>
        <TD>ivo://ism.obspm</TD>
        <TD>DIFF51_r3e0A3e0d7e2_s_20</TD>
        <TD></TD>
      </TR>
    </TABLEDATA>
  </DATA>
</TABLE>

<TABLE name="links">
  <FIELD name="experiment" datatype="char" arraysize="*"></FIELD>
  <FIELD name="link-rel" datatype="char" arraysize="*"></FIELD>
  <FIELD name="link-uri" datatype="char" arraysize="*"></FIELD>

  <GROUP name="foreign_key" ref="results">
    <GROUP>
      <PARAM name="local_field" value="experiment"/>
      <FIELDRef ref="ident"/>
    </GROUP>
  </GROUP>

  <DATA>
    <TABLEDATA>
      <TR>
        <TD>ism.obspm/DIFF51_r3e0A3e0d7e2_s_20</TD>
        <TD>simdm/experiment/xml</TD>
        <TD>link_toward_experiment_xml_serialization</TD>
      </TR>
      <TR>
        <TD>ism.obspm/DIFF51_r3e0A3e0d7e2_s_20</TD>
        <TD>simdm/protocol/xml</TD>
        <TD>link_toward_protocol_xml_serialization</TD>
      </TR>
      <TR>
        <TD>ism.obspm/DIFF51_r3e0A3e0d7e2_s_20</TD>
        <TD>simdm/project/xml</TD>
        <TD>http://link_toward_project_xml_serialization</TD>
      </TR>
    </TABLEDATA>
  </DATA>
</TABLE>
```

link-rel	SimDAL Component
`simdm/experiment/xml`	SimDM XML serialization of the experiment
`simdm/project/xml`	SimDM XML serialization of project
`simdm/protocol/xml`	SimDM XML serialization of protocol
`simdm/parties/xml`	SimDM XML serialization of parties

Table 5: Standard values for the link-rel attribute

```
  </RESOURCE>
</VOTABLE>
```

The resources associated with every experiment are listed in the *links* table, with the following standardized `link-rel`:

5.4 {cutouts}

The {cutouts} resource allows a client to define a cut-out in a view and retrieve the corresponding data.

This resource represents a collection of *cutout*s. A *cutout* is represented as a VOTable encoded list of the values that fulfill the associated *cutout* request.

Conforming to traditional REST practices, a *cutout* resource is created by POSTing a query to the *{cutouts}* collection resource.
Here, *cutout* is really a cut-out in an hypercube (in this case the axis of the hypercube are the fields of the *view*). Since the *view*s in SimDAL Search components are mainly about metadata, it is then more a metadata cut-out than a traditional data cut-out (to the contrary, SimDAL Data Access components have also a *cutout* resource but for data).

The *{cutouts}* resource can be either (or both) a sync or an async resource conforming to DALI sync, async respectively.

Queries conform to the query language defined above.

Example of a cut-out in which we search for datasets having maximum silicate temperatures (c:max:temperature_grain_silicates) above 30 K and minimum charge (c:min:charge_grain_silicates) below -1. In this query we ask to retrieve fields, datasets IDs (c:outputdataset_pubdid) that could be used in a SimDAL Data Access service to retrieve data and other fields corresponding to gas density, silicate maximum temperature, gas temperature and minimum charge of silicates.

```
curl -H "Content-Type: application/JSON" -d

'{
  "where": [
    {":att": "c:max:temperature_grain_silicates", ":val": 30, ":op": ">"},
    {":att": "c:min:charge_grain_silicates", ":val": -1, ":op": "<"}
  ],

  "select": [
    "c:outputdataset_pubdid",
    "c:gas_density",
```

```
    "c:max:temperature_grain_silicates",
    "c:gas_temperature",
    "c:min:charge_grain_silicates"
  ],

  "orderby": ["c:gas_density", "asc"]
}'

http://<cutouts_resource>
```

Sync Response format The *cutout* resource representation (i.e *{cutouts}* query response format) is a VOTable. The complete metadata header (composed GROUPS etc...), describing the complete schema of the *view* is not required, with the exception of the FIELD elements.

Example of the newly created *cutout* resource :

```
<VOTABLE XMLns="http://www.ivoa.net/XML/VOTable/v1.2">
  <RESOURCE type="results">
    <INFO name="REQUEST_STATUS" value="OK"/>
    <TABLE name="my_view_pubid">
      <LINK content-role="next_page"
           href="http://<simsearch uri>/cutouts/1?view=my_view_pubid&page_size=5&page=2"/>
      <PARAM name="page_size" datatype="int" value="5"/>

      <FIELD name="outputdataset publisher did"
            ID="c:outputdataset_pubdid"
            datatype="float"
            utype="SimDM:/resource/experiment/OutputDataset.publisherDID">
            <LINK content-role="view/data_access"
                 href="http://localhost:3435/api/simdal/DIFF51_n_ERLH_isot"/>
      </FIELD>

      <FIELD name="Max Grain temperature silicates"
            ID="c:max:temperature_grain_silicates"
            utype="simdm:/resource/experiment/statistical_summary.numeric_value.value"
            datatype="float" unit="K">
        <!-- SKOS concept (SimDM 'label' attribute) -->
        <LINK content-role="type"
             href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
      </FIELD>

      <FIELD name="Min Charge Grain silicates"
            ID="c:min:charge_grain_silicates"
            utype="simdm:/resource/experiment/statistical_summary.numeric_value.value"
            datatype="float" unit="K">
        <!-- SKOS concept (SimDM 'label' attribute) -->
        <LINK content-role="type"
             href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
      </FIELD>

      <FIELD name="Gas density"
            ID="c:gas_density"
```

```
          utype="simdm:/resource/experiment/parameter_setting.numeric_value.value"
          datatype="float" unit="cm-3">
    </FIELD>
    <FIELD name="Gas temperature"
          ID="c:gas_temperature"
          utype="simdm:/resource/experiment/parameter_setting.numeric_value.value"
          datatype="float" unit="K">
    </FIELD>

    <DATA>
      <TABLEDATA>
        <TR>
          <TD>rad_dset12</TD>
          <TD>50.57</TD>
          <TD>−2.31</TD>
          <TD>1e5</TD>
          <TD>456.2</TD>
        </TR>
        <TR>
          <TD>rad_dset17</TD>
          <TD>52.34</TD>
          <TD>−1.11</TD>
          <TD>1e5</TD>
          <TD>402.3</TD>
        </TR>
        <TR>
          <TD>rad_dset23</TD>
          <TD>31.01</TD>
          <TD>−2.81</TD>
          <TD>1e5</TD>
          <TD>103.2</TD>
        </TR>
        <TR>
          <TD>rad_dset24</TD>
          <TD>37.09</TD>
          <TD>−1.49</TD>
          <TD>1e4</TD>
          <TD>110.2</TD>
        </TR>
        ...
      </TABLEDATA>
    </DATA>
  </TABLE>
 </RESOURCE>
</VOTABLE>
```

5.5 {fields}

The goal of the *{fields}* resource is to facilitate the handling of very large *view*s schemas. It permits to get the description of individual FIELDS. It adresses two use cases that become relevant when a client wants interactive access to a schema that is very large:

- to search for a specific field without downloading the whole schema
- to get the schema subset relative to a specific field (once it has been identified through a previous search).

The *{fields}* resource location is provided by the *{views}* resource, through the `link-rel` `view/fields`.

The resource itself does not provide a representation, because the collection may be very large and the data is not really relevant since it can be obtained through the *{views}* schema.

Query The search is done by setting the parameter `q` of the resource *search* in a GET query. The search is done on the field name based on the text pattern provided by the parameter `q`.

Example:

```
http://<fields_resource uri>/search?q="Grain temperature"
```

Response schema The response VOTable schema is very simple:

- `match`: the matching value among the document attributes values.
- `ident`: the ID of the schema field

Example:

```
...
<RESOURCE type="results">
  <INFO name="QUERY_STATUS" value="OK"/>
  <PARAMETER name="q" value="Grain temperature for silicates"/>
   ...
  <TABLE>
    <FIELD name="match" datatype="char" arraysize="*"></FIELD>
    <FIELD name="ident" datatype="char" arraysize="*"></FIELD>

    <DATA>
      <TABLEDATA>
        <TR>
          <TD>Max Grain temperature for amorphous carbons</TD>
          <TD>c:max:temperature_grain_amorphous_carbons</TD>
        </TR>
         ...
        <TR>
          <TD>Max Grain temperature for silicates</TD>
          <TD>c:max:temperature_grain_silicates</TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
</RESOURCE>
...
```

Get field schema Getting the schema of a specific field is done by providing the `id` parameter to the *fields* resource in a GET query. For example, for the field that has ID `c:max:temperature_grain_silicates` (ID found in *field search* described above as FIELD named `ident`.)

http://<field_resource uri>/fields?id=c:max:temperature_grain_silicates

returns:

```
<VOTABLE XMLns="http://www.ivoa.net/XML/VOTable/v1.2">
  <RESOURCE type="results">
    <TABLE name="my_view_pubid">
      <FIELD name="Max Grain temperature for silicates"
            ID="c:max:temperature_grain_silicates"
            utype="simdm:/resource/experiment/StatisticalSummary.numericValue.value"
            datatype="float" unit="K">
        <VALUES>
          <MIN value="16.9"/>
          <MAX value="77.2" inclusive="yes"/>
        </VALUES>
        <LINK content-role="type"
             href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
      </FIELD>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

5.6 {cutouts-preview}

As explained above, when a cut-out is requested, the service returns, essentially, a list of the values that fulfill the cut-out request. The cutout-preview optional feature provides a summary of the information that would be obtained in the cut-out: minimum and maximum values for each of the requested fields and the total number of results that would be obtained. This is done the same way as the one for the "cutout" but adding the VALUES element for each FIELD and no DATA. This is useful so that final users can get a hint about if they are asking for too many results or not and, thus, if they should refine the query.

Depending on how the service is implemented, in some cases it is easier and faster to generate this preview than the actual list of results, but in other cases it would be the opposite. Thus, the feature is optional.

Example:

```
curl -H "Content-Type: application/JSON" -d

"{
  where: [
    {':att': 'c:max:temperature_grain_silicates', ':val': 30, ':op': '>'},
    {':att': 'c:min:charge_grain_silicates', ':val': -1, ':op': '<'}
  ],

  select: [
    'c:proton_density',
    'c:max:temperature_grain_silicates',
```

```
    'c:gas_temperature',
    'c:min:charge_grain_silicates'
  ]
}"

http://<cutout_preview_resource>
```

returns:

```
<VOTABLE xmlns="http://www.ivoa.net/xml/VOTable/v1.2">
  <RESOURCE type="results">
    <TABLE name="my_view_pubid">
      <PARAM name="nresults" datatype="int" value="5000"/>
      <FIELD name="Max Grain temperature silicates"
            ID="c:max:temperature_grain_silicates"
            utype="simdm:/resource/experiment/statistical_summary.numeric_value.value"
            datatype="float" unit="K">
        <!-- SKOS concept (SimDM 'label' attribute) -->
        <LINK content-role="type"
             href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
        <VALUES>
          <MIN value="30.1"/>
          <MAX value="60.2"/>
        </VALUES>
      </FIELD>

      <FIELD name="Min Charge Grain silicates"
            ID="c:min:charge_grain_silicates"
            utype="simdm:/resource/experiment/statistical_summary.numeric_value.value"
            datatype="float" unit="K">
        <!-- SKOS concept (SimDM 'label' attribute) -->
        <LINK content-role="type"
             href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
        <VALUES>
          <MIN value="-4.1"/>
          <MAX value="-1.1"/>
        </VALUES>
      </FIELD>

      <FIELD name="Proton density"
            ID="c:proton_density"
            utype="simdm:/resource/experiment/parameter_setting.numeric_value.value"
            datatype="float" unit="cm-3">
        <VALUES>
          <MIN value="1.0e5"/>
          <MAX value="1.0e4"/>
        </VALUES>
      </FIELD>
      <FIELD name="Gas temperature"
            ID="c:gas_temperature"
            utype="simdm:/resource/experiment/parameter_setting.numeric_value.value"
            datatype="float" unit="K">
        <VALUES>
```

```
          <MIN value="2.7"/>
          <MAX value="11000.0"/>
        </VALUES>
      </FIELD>

    </TABLE>
  </RESOURCE>
</VOTABLE>
```

6 SimDAL Data Access

The goal of the SimDAL Data Access component is to retrieve data. Usually this is done after a discovery phase in a SimDAL Search component; a dataset has been identified and its *dataset* ID is known by the client. In fact, it is also possible to access a SimDAL Data Access component directly without the knowledge of a *dataset* ID as explained below.

Two main operations can be done on *dataset*s at the level of SimDAL Data Access components: download the whole raw data ({rawdata}) or extract a part of these data ({cutouts}).

Simulation outputs are stored in files in various heterogenous formats: ASCII, HDF5, VTK, FITS, proprietary format, ... SimDAL does not define a standard format in which the data should be sent to clients. That is a limit to interoperability. Obviously, whenever possible publishers should provide their data in VO standard formats like VOTable and FITS.

The advantages of providing a cut-out service as part of SimDAL Data Access are:

- manage large volume of data and avoid to download the whole data
- knowledge (even basic) about the raw data format is no longer needed
- installing, configuring, using specific extraction tools to read specific data format is no longer needed for the end user
- standard, interoperable, web accessible, VO access interface

6.1 SimDAL Data Access API design

The SimDAL Data Access component has a design similar to the one of the SimDAL Search component. Indeed, the *dataset*s (SimDM OutputDatasets) it deals with are hyper-cubes (from 1 to N dimensions) that can also be seen as tables where rows are the objects and columns (VOTable FIELDs) are the object properties. So, cut-outs in *dataset*s are similar to cut-outs in *view*s.

In the common use case, a *dataset* ID is found in the SimDAL Search component. If the client reaches a SimDAL Data Access service without this first discovery phase, it can access the *{datasets}* resource which will list all the available *dataset*s (plus some associated metadata), and it is up to users/clients to identify which one interests them.

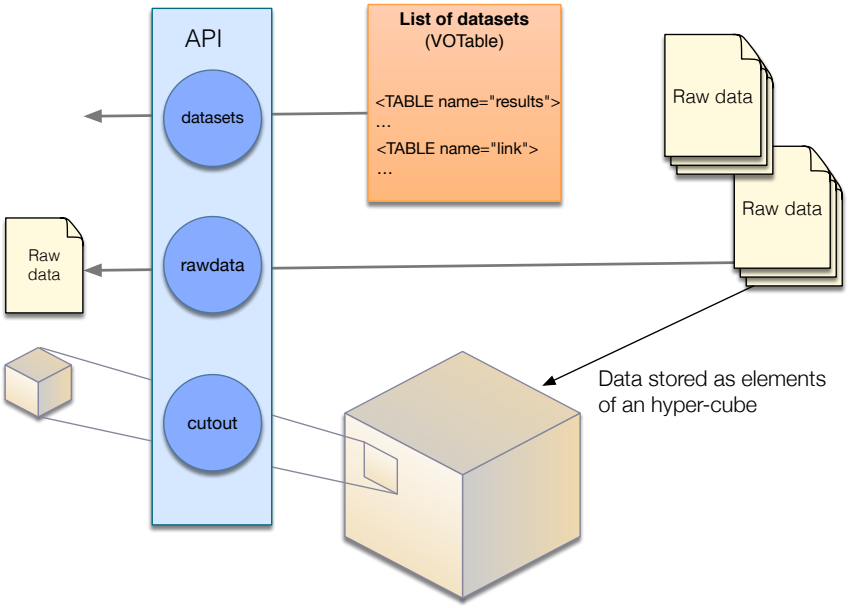

Figure 8: Architecture of the SimDAL Data Access. The SimDAL Data Access gives access to data. The raw data themselves may be directly downloaded with the {rawdata} resource. If required a *{cutout}* resource may be implemented. In that case, the axis on which the cut-out can be done, are considered axis of an hyper-cube. The cut-out is done in this hyper-cube.

Dataset schema In order to make SimDAL Data Access components as independent as possible from the SimDAL Search components, a *dataset* schema is provided by the service, even if the information could theoretically be grabbed from the XML serialization of the protocol related to the *dataset* (and the relevant ObjectType). So it is possible for a publisher to provide its own implementation of this SimDAL component, without having to implement a SimDAL Search component. It also allows for tuning which object properties to make available, or not (i.e not necessarily the same that those in the XML serialization of the protocol, as long as they are described in the *dataset* schema). For example, a publisher publishes output results of a code that computes many properties for each object (all of them properly described in the XML serializations of the protocol and experiments) but the publisher decides that only a few of them are accessible in the SimDAL Data Access component.

Datasets schemas are similar to *views* schemas.

6.2 {datasets}

The goal of the *{datasets}* resource is to provide the list of the avalable *datasets* and a set of associated links (ex: link towards the *dataset* schema).

It is a collection and its representation is a VOTable listing the metadata about the available *datasets* in the SimDAL Data Access service.

Parameters

- `dataset`: obtained either from the full *{datasets}* listing if available, or from a SimDAL Search service (*{cutouts}* resource). This is optional.

The value of the `dataset` parameter is obtained either from the *{datasets}* resource listing or from the *{cutouts}* resource in a SimDAL Search service. In the later case, it corresponds to a *view* FIELD having utype `simdm:/resource/experiment/OutputDataset.publisherDID`

Note: The `dataset` parameter must be specified to access a specific dataset resource representation. If not provided (that is the *{datasets}* resource is accessed withtout any parameter), and if the publisher implemented it, the representation of the collection resource will list all the available *dataset*s. This is optional because their may be cases where the list would be too large. If the collection representation is not available, the representation is an empty list (TABLE element with name *results*).

Example:

http://<simdata_access uri>/datasets?dataset=DM51NoPAH_20_cloud

The example above returns the metadata of the *dataset* having the ID `DM51NoPAH_20_cloud` in a VOTable. This VOTable must have a PARAM element that recalls parameter values. It also provides the links towards the associated resources, like the schema of the *dataset* and the associated *{fields}* resource.

The resulting VOTable is presented below, following the same design than the SimDAL Search *{views}* resource regarding the metadata FIELDs. Thus, the only one required FIELD is `ident`

```
 ...
<RESOURCE type="results">
  <INFO name="QUERY_STATUS" value="ok"/>
  <PARAM name="dataset" value="DM51NoPAH_20_cloud"/>
  <TABLE name="datasets">
    <FIELD arraysize="*" datatype="char" name="ident" ID="ident"/>
    <FIELD arraysize="*" datatype="char" name="created"/>
    <DATA>
      <TABLEDATA>
        <TR>
          <TD>DM51NoPAH_20_cloud</TD>
          <TD>2016-03-29 15:03:03</TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
  <TABLE name="links">
    <FIELD arraysize="*" datatype="char" name="dataset"/>
    <FIELD arraysize="*" datatype="char" name="link-rel"/>
    <FIELD arraysize="*" datatype="char" name="link-uri"/>
    <GROUP name="foreign_key" ref="datasets">
      <GROUP>
        <PARAM name="local_field" value="dataset"/>
        <FIELDRef ref="ident"/>
      </GROUP>
    </GROUP>
    <DATA>
      <TABLEDATA>
```

```
        <TR>
          <TD>DM51NoPAH_20_cloud</TD>
          <TD>dataset/schema</TD>
          <TD>
            http://<da-server>/api/simdal/p1/datasets/DM51NoPAH_20_cloud/schema
          </TD>
        </TR>
        <TR>
          <TD>DM51NoPAH_20_cloud</TD>
          <TD>dataset/cutouts</TD>
          <TD>
            http://<da-server>/api/simdal/p1/datasets/DM51NoPAH_20_cloud/cutouts
          </TD>
        </TR>
        <TR>
          <TD>DM51NoPAH_20_cloud</TD>
          <TD>dataset/async/cutouts</TD>
          <TD>
            http://<da-server>/api/simdal/p1/datasets/DM51NoPAH_20_cloud/async/cutouts
          </TD>
        </TR>
        <TR>
          <TD>DM51NoPAH_20_cloud</TD>
          <TD>dataset/rawdata</TD>
          <TD>
            http://<da-server>/api/simdal/p1/datasets/DM51NoPAH_20_cloud/rawdata
          </TD>
        </TR>
      </TABLEDATA>
    </DATA>
  </TABLE>
 </RESOURCE>
...
```

Standalone SimDAL Data Access component In the case where a publisher wants to publish only some *dataset*s from a simulation project without setting up a SimDAL Search service, the *{datasets}* resource can be accessed without the *dataset* parameter, returning the list of the metadata for all the published *dataset*s of the project. In this case, the number of *dataset*s is generally small, so that the list size would be manageable.

One must keep in mind that, in this case, the only way to know what the *dataset*s are about are the basic metadata provided by the *{datasets}* resource (ex: objecttype, protocol).

Standard links associated with a dataset A set of standard links can be associated with a *dataset*.

The links below the horizontal line are optional but at least one among `dataset/cutouts`, `dataset/async/cutouts` and `dataset/rawdata` is required.

The special link-rel `view/reference` allows citable references to be explictly linked to the view.

link-rel	description
`dataset/schema`	URI of the dataset schema
`dataset/cutouts`	URI of the *cutouts* resource
`dataset/async/cutouts`	URI of the *async/cutouts* resource
`dataset/rawdata`	URI of the *rawdata* resource
`dataset/reference`	URI of a reference paper, website, or material related to the research work present in the dataset

Table 6: Standard values for the link-rel FIELD

6.3 {cutouts}

The goal of this resource is to retrieve values of a subset of a dataset's raw data easily.
To query an async *{cutouts}* resource a JSON document is posted, for instance:

```
curl −H "Content−Type: application/JSON" −d

"{
  where: [
    {":axis": "n_oh", ":op": ">", ":val": 2e−07},
    {":axis": "n_oh", ":op": "<", ":val": 5e-07}
  ],
  select: ["av", "n_oh"]
}"

http://<simdal-data-uri>/DM51NoPAH_20_cloud/async/cutouts/?format=votable&RUNID=j1
```

It eventually returns a UWS resource (see section 2.2.2.2 in UWS 1.1, Harrison and Rixon (2016)):

```
<uws:job
    xmlns:uws="http://www.ivoa.net/xml/UWS/v1.0"
    xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:uwsc="urn:uwscustom" version="1.1"
    xsi:schemaLocation="http://www.ivoa.net/xml/UWS/v1.0 UWS.xsd">
  <uws:jobId>c2aff943−f608−4e63</uws:jobId>
  <uws:ownerId xsi:nil="true"/>
  <uws:runId>j1</uws:runId>
  <uws:phase>COMPLETED</uws:phase>
  <uws:startTime>2016−04−08 10:03:13.550386</uws:startTime>
  <uws:endTime>2016−04−08 10:03:13.829351</uws:endTime>
  <uws:executionDuration>n/a</uws:executionDuration>
  <uws:destruction/>
  <uws:parameters>
    <uws:parameter id="where">
      [{":axis": "n_oh", ":op": ">", ":val": 2e−07}, {":axis": "n_oh", ":op": "<", ":val": 5e−07}]
    </uws:parameter>
    <uws:parameter id="select">["av", "n_oh"]</uws:parameter>
    <uws:parameter id="project">"p1"</uws:parameter>
    <uws:parameter id="dataset">"DM51NoPAH_20_cloud"</uws:parameter>
  </uws:parameters>
  <uws:results>
```

```
    <uws:result id="c2aff943-f608-4e63_1"
                name="cutout"
                xlink:href="http://<da-server>/api/simdal/results/c2aff943-f608-4e63_1"
                size="1"
                mime-type="application/xml"/>
  </uws:results>
</uws:job>
```

Thus, the client can access `http://<da-server>/api/simdal/results/c2aff943-f608-4e63_1` to retrieve the result of the cut-out:

```
<VOTABLE XMLns="http://www.ivoa.net/XML/VOTable/v1.3">
  <RESOURCE type="results">
    <INFO name="REQUEST_STATUS" value="OK"/>
    <TABLE name="dataset">
      <FIELD name="av" datatype="float"/>
      <FIELD name="n_oh" datatype="float"/>
      <DATA>
        <TABLEDATA>
          <TR>
            <TD>1e5</TD>
            <TD>456.2</TD>
          </TR>
          ...
        </TABLEDATA>
      </DATA>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

Query Unlike in SimDAL Search, the `orderby` query parameter is not supported in Data Access because it is not so relevant and can bring significant implementation problems due to the nature of the underlying data and the very specific associated extraction pipelines.

Note about UWS Because of well know security concerns about UWS (see the standard document), the `{cutouts}` resource (mapped to `{jobs}` in UWS document) does not return a list of the jobs.

sync Unlike the SimDAL Search component handling *view*s of metadata, a SimDAL Data Access service deals with output data from the code (OutputDataset). These data can be very large, the subset extraction process may be CPU intensive and output files may be large.

As a consequence, the synchronous collection pagination approach used in the SimDAL Search API may not be very relevant and the async approach would be more convenient.
When the sync *{cutouts}* resource is implemented by the SimDAL Data Access service it will be queried in the same way explained above, posting a JSON document to the `sync/cutouts` resource, and the answer will be directly the corresponding VOTable (as in the example above) with the optional pagination approach, as explained in SimDAL Search.

6.4 {rawdata}

Some publishers may just want to offer a direct download of their data files (possibly archived). The *{rawdata}* resource provides a list of links, with their type as a mime/type, where the files can be downloaded.

Resource interface The *{rawdata}* resource is very simple, it is a list (in a VOTable representation) of links towards the data files associated with a given *dataset*.
Example:

http://<simdata_access uri>/datasets/DM51NoPAH_20_cloud/rawdata

returns the following VOTable

```
<VOTABLE
   xmlns="http://www.ivoa.net/xml/VOTable/v1.3"
   xmlns:stc="http://www.ivoa.net/xml/STC/v1.30"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <RESOURCE type="results">
   <INFO name="QUERY_STATUS" value="OK"/>
   <FIELD arraysize="*" datatype="char" name="link-uri"/>
   <FIELD arraysize="*" datatype="char" name="link-mimetype"/>
   <TABLE name="results">
     <DATA>
       <TABLEDATA>
         <TR>
           <TD>
             http://localhost:3435/p1/rawdata/p1/data/cloud/DM51NoPAH_A2e0_20.hdf5
           </TD>
           <TD>application/x-hdf</TD>
         </TR>
       </TABLEDATA>
     </DATA>
   </TABLE>
 </RESOURCE>
</VOTABLE>
```

6.5 {fields}

The API is the same as the one of the SimDAL Search component. Thus, the main use case is the same: make it possible to search for / access fields without having to retrieve a potentially big schema file. This is particularly useful in interactive user interfaces.

A Scientific use cases

As explained in the head of the document, the diffusion of simulations is a more complex topic than the diffusion of astronomical images or spectra. Most of the difficulty comes from the heterogeneity of simulations and the fact that, a code, can compute many things. For example, a single code simulating an astrophysical object can compute many physical quantities in each cell of a multi-dimensional grid as a function of time, but can also produce images, spectra, catalogs or anything else the scientist thinks meaningful. Code outputs can even be post-processed to compute other quantities and produce other outputs. The Simulation Data Model describes comprehensively all these possibilities. Concerning the access protocol, SimDAL version 1.0, is designed to answer to the most important use cases but not all of them, that are left for future versions of the standard.

The main use cases to which SimDAL version 1.0 must answer have been defined by the IVOA Theory I.G. A first list was defined in 2005 at the ESAC InterOp. They have been specified in 2013 when the works on SimDAL started.

Case 1 - Search for stellar spectra Scientists want to get a theoretical spectrum of star with an effective temperature of 10 000 K and log(g) = 4.0.

1. They browse the VO to discover services modeling stars and producing spectra or using a specific numerical code. To do so, they may do a text search: *stellar models* and *spectra*.
2. They select a service from the list of proposed services which are associated with the search criteria. The system sends back description of the services. Here, this description should contain the description of the service, the description of the code computing stellar models (physics), and the quantities computed as well as the type of results (example : spectra)
3. Once they have selected the service, they have to provide some specific parameters as effective temperature and the log(g). They may also have to provide other parameters required by the service. The service asks users for input parameters that are going to be used to select a model in the database.
4. The service proposes the download of the spectrum.

Case 2 - Large simulations - search for catalogs and datacubes Scientists are interested in large structures in cosmological context. They want to find a list of properties of halos (i.e. their masses, positions, velocities) in a LCDM model at z about 3. They are interested in halos with masses between M1 and M2. Apart the list of halos with their properties, they also want to download data cubes of these halos to have the positions of particles and their velocities.

1. They browse the VO to discover all theoretical services publishing cosmological simulations They may search in a registry-like system for keywoards equals to: *N-body simulations*, *Large structures*, *galaxy halos* etc...
2. From the responses, they select one of the service they are interested in and browse it to discover an experiment as well as redshifts / time step fulfilling their requirements. To browse in all available simulations, they have to select some input parameters.
3. The service may provide access to halos detected by different FriendOfFriend (FOF) methods. Users have to choose the FOF methods and eventually its parameter. From the previous step, the users / clients have a list of experiments. They select one of them. The system tells them which FOF algorithms have been used to post-process these experiments. They select one these FOF programs and fill-in values for the input parameters. The system answers with a catalog of halos.

4. They then ask a download of a catalog of properties of halos (eventually reducing the range and specifying the properties attached to halos). Users then specify the physical quantities associated to each halos they are interested in and download them.

5. Then they ask for a part of a simulated cube with some properties attached to each point / particles. They select a position in the experiment and a size for the cube, give the list of the physical quantities they wish and the system allows them to download the resulting file.

Case 3 - Get density profiles in MHD simulations Scientists want to get the density - velocity distribution in an interstellar clump simulated by MHD simulations.

1. They browse the Virtual Observatory to discover services publishing such simulations
2. They select a project gathering one or several simulations producing clumps
3. They select an experiment and a timestamp
4. They browse through this experiment to discover interesting clumps on various criteria (for example the mean pressure)
5. They select one clump
6. They extract the data corresponding to this clump

Case 4 - Preparation of spectroscopic observations Scientists want to ask observation time on a telescope to observe CO in a foreign galaxy. They need to get some estimations of line intensities of CO 1-0, 2-1 and 15-14 for interstellar clouds with a density of about 10 000 cm^{-3} and different UV illuminations.

1. They browse the VO to discover theoretical services computing CO line intensities.
2. They select the service and choose some models from input parameters corresponding to the object they want to observe.
3. They select the CO lines they are interested in.
4. Then they extract those CO line intensities.

Case 5 - Interpretation of spectroscopic observations Scientists want to interpret observations in interstellar clouds. They did data analysis and bibliography and they know that, in an object these rules apply: 10^{20} cm^{-2} < $N(H_2)$ < $3{\times}10^{20}$ cm^{-2} and $N(CO) > 2{\times}10^{14}$ cm^{-2}. They also guess that the proton density should be lower than 500 cm^{-3}. They want to get maps of these observables in space parameters, find models reproducing these observations and then download quantities from these models as densities and temperature profiles.

1. First, they search through the VO to discover theoretical services computing column density of $N(H_2)$ and N(CO) and have density as parameter.
2. They select a list of services and provide its constrains.
3. Each services send back the list of models matching the constrains
4. Then, scientists download the full models or a selection of properties for each model.

B SimDAL APIs design

In SimDAL, most of the complexity is server-side so that it is as easy as possible to develop clients. The resulting VO eco-system for numerical simulations will therefore be more something like multiple small but highly specific clients rather than one big generic multi purpose client (which is close to the current VO status with tools like TOPCAT that is widely used).

The goal is to allow any users (not only developers) to develop a tiny client layer that fulfills his specific needs. This is particularly relevant in the case of numerical simulations since SimDM is actually a meta model, very abstract, and is instantiated in a new data model for each new numerical simulation protocol. As a comparison, we could say that other IVOA standards for observations represent only one such data model, fixed in the standard.

Using TAP in this specific context (direct mapping of SimDM in a relational database) would result in very complex SQL queries, i.e not accessible for a mere scientist but almost only to SQL experts and some developers. In addition, to have relevant and thorough metadata in `TAP_SCHEMA` for quantities that are encoded following the EAV pattern in SimDM, we need to pivot the data to shift EAV attributes in table columns (and put the metadata in `TAP_SCHEMA.columns`). However, it's common to have many more properties (i.e table columns) than the current rdbms can handle (more than 100 000) (ex: 1024 for SQL Server, 4096 for MySQL, ...)

But, RDBMS still is the implementation of choice (the basic view query language described below is a subset of SQL) and therefore, can be plugged on top of any TAP very easily.

In addition, we achieve loose coupling with implementation/technological solution (TAP is tightly coupled with relational database).

Finally, in a scientist work added value perspective, the publishers pre-format their data the way they think will be the most useful/easy to process/analyse/exploit, because they are the most capable/relevant actor for doing that.

We made the APIs with the intent of putting users first, by considering that users should be anyone, from developers to mere scientists. This comes with the tradeoff that the publisher should know about the technical aspects.

In fact, the complexity of URI handling, very error prone, is kept server side, i.e handled only once, by only one actor (that is supposed to be technic expert). This allows the end user to not worry about this error prone processing.

C Complex schema example

It may be necessary to add more information in a *view* or *dataset* schema, for example some sort of relationship between FIELDs. VOTables provide GROUP that are useful.
Example of a schema for a basic *view*. Note the use of utypes and SKOS to identify the content of the *view* fields.

```
<VOTABLE XMLns="http://www.ivoa.net/XML/VOTable/v1.3">
  <RESOURCE type="results">
    <TABLE name="my view schema">
      <GROUP>
        <PARAM name="instance" value="simdm:resource/experiment/Experiment"/>
```

```
      <PARAM name="protocol"
            utype="simdm:/resource/experiment/Experiment.protocol"
            value="pdr_152_r34"/>

      <GROUP utype="simdm:resource/experiment/Experiment.parameter">
        <PARAM name="instance" value="simdm:resource/protocol/InputParameter"/>
        <PARAM name="input_parameter"
              utype="simdm:/resource/experiment/ParameterSetting.inputParameter"
              value="densh"/>
        <FIELDRef ref="c:densh"
                  utype="simdm:/resource/experiment/ParameterSetting.numericValue.value"/>
      </GROUP>

      <GROUP utype="simdm:resource/experiment/Experiment.outputData">
        <PARAM name="instance"
              value="simdm:resource/experiment/OutputDataset"/>
        <PARAM name="object"
              utype="simdm:/resource/experiment/OutputDataset.objectType"
              value="grain"/>
        <FIELDRef ref="c:grains_outputdataset_pubdid"
                  utype="simdm:/resource/experiment/OutputDataset.publisherdid"/>

        <GROUP utype="simdm:resource/experiment/OutputDataset.characterisation">
          <GROUP>
            <PARAM name="instance" value="simdm:resource/experiment/StatisticalSummary"/>

            <PARAM name="axis"
                   utype="simdm:/resource/experiment/StatisticalSummary.axis"
                   value="temperature_grain_silicates"/>
            <PARAM name="statistic"
                   utype="simdm:/resource/experiment/StatisticalSummary.statistic"
                   value="max"/>
            <FIELDRef ref="c:max:temperature_grain_silicates"/>
          </GROUP>
        </GROUP>
      </GROUP>

    </GROUP>

    <FIELD name="Proton density"
           ID="c:densh"
           utype="simdm:/resource/experiment/ParameterSetting.numericValue.value"
           datatype="float" unit="cm-3">
      <LINK content-role="type"
            href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
    </FIELD>

    <FIELD name="Max Grain temperature for silicates"
           ID="c:max:temperature_grain_silicates"
           utype="simdm:/resource/experiment/StatisticalSummary.numericValue.value"
           datatype="float" unit="K">
      <VALUES>
```

```
        <MIN value="16.9"/>
        <MAX value="77.2" inclusive="yes"/>
      </VALUES>

      <LINK content-role="type"
           href="http://purl.obspm.fr/vocab/PhysicalQuantities/TemperatureOfGrains"/>
    </FIELD>

    <FIELD name="Grains dataset id"
          ID="c:grains_outputdataset_pubdid"
          utype="simdm:/resource/experiment/OutputDataset.publisherdid"
          datatype="string"/>

  </TABLE>
 </RESOURCE>
</VOTABLE>
```

References

Cassisi, S., Cervino, M., Lemson, G., Osuna, P., Schaye, J., Walton, N. and Wozniak, H. (2008), 'D11 – teg report: Framework for the inclusion of theory data and services in the vobs'.
URL: *http://wiki.ivoa.net/internal/IVOA/IvoaTheory/EuroVO-DCA-D11-MPG-Final.pdf*

Dowler, P., Bonnarel, F. and al. (2015), 'Datalink', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/DataLink/*

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

ECMA (2013), 'Ecma-404 - the json data interchange format'.
URL: *http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf*

Fowler, M. (2010), 'Richardson maturity model'.
URL: *http://martinfowler.com/articles/richardsonMaturityModel.html*

Harrison, P. and Rixon, G. (2016), 'Universal worker service pattern, version 1.1', IVOA Recommendation.
URL: *http://www.ivoa.net/Documents/UWS*

Hunter, T. (2013), 'Principles of good restful api design'.
URL: *http://codeplanet.io/principles-good-restful-api-design/*

Lemson and al. (2012), 'Simulation data model'.
URL: *http://www.ivoa.net/documents/SimDM/20120503/html/SimDM.html*

Lemson, G., Bourges, L., Cervino, M., Gheller, C., Gray, N., LePetit, F., Louys, M., Ooghe, B., Wagner, R. and Wozniak, H. (2012), 'Simulation data model, version 1.0', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/SimDM*

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', IVOA Recommendation.
URL: *http://www.ivoa.net/documents/VOTable/*