International

Virtual

Observatory

Alliance

IVOA AccessData
Version 1.0
IVOA Working Draft 2014-03-12

Interest/Working Group:

http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/IvoaDAL

This version:

WD-AccessData-1.0-20140312

Latest version:

Not yet issued

Previous version(s):

Editors:

Patrick Dowler

Authors:

Patrick Dowler, ...

Abstract

This document describes the AccessData web service capability. AccessData is a low-level data access capability that can act upon the data files, performing various kinds of operations: filtering/subsection, transformations, pixel operations, and applying functions to the data.

Status of This Document

This is a working draft internal to the DAL-WG.

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

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

Acknowledgments

The authors would like to thank all the participants in DAL-WG discussions for their ideas, critical reviews, and contributions to this document.

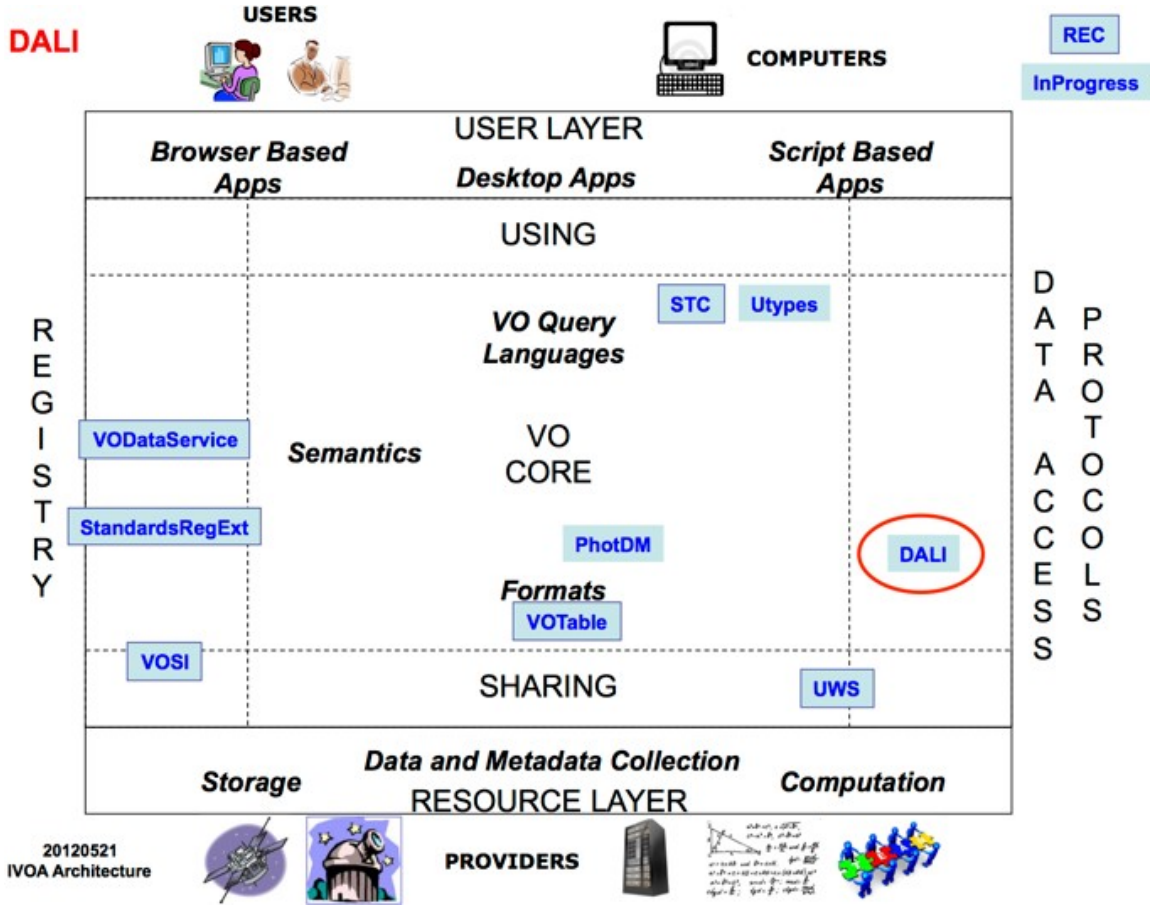
Contents

1 Introduction

The AccessData web service interface defines a RESTful web service for performing server-side operations on data before transfer.

1.1 The Role in the IVOA Architecture

TODO: new diagram from TCG



AccessData services conform to the Data Access layer Interface (DALI [1]) specification, including the Virtual Observatory Support Interfaces (VOSI [2]) resources.

1.2 Motivating Use Cases

Below are some of the more common use cases that have motivated the development of the AccessData specification. While this is not complete, it helps to understand the problem area covered by this specification.

1.2.1 Retrieve Subsection of a Datacube

Cutout a subsection using coordinate axis values. The input to the cutout operation will include one or more of the following:

- a region on the sky
- an energy value or range
- a time value or range
- one or more polarization states

The region on the sky should be something simple: a circle, a range of coordinate values, or maybe a polygon.

1.2.2 Retrieve subsection of a 2D Image

This is a special case of 1.2.1 where the cutout is only in the spatial axes.

1.2.3 Retrieve subsection of a Spectrum

This is a special case of 1.2.1 where the cutout is only in the spectral axis.

1.2.4 Flatten a Datacube into a 2D Image

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.5 Flatten a Datacube into a 1D Spectrum

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.6 Rebin Data by a Fixed Factor

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.7 Reproject Data onto a Specified Grid

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.8 Compute Aggregate Functions over the Data

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.9 Apply Standard Function to Data Values

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.10 Apply Arbitrary User-Specified Function to Data Values

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

1.2.11 Run Arbitrary User-Supplied Code on the Data

This use case will be developed and supported in the AccessData-1.1 (or later) specification.

2 Resources

AccessData services are implemented as HTTP REST [18] web services with a {sync} resource that conforms to the DALI-sync resource description.

resource type	resource name	required
{sync}	service specific	
{async}	service-specific	
DALI-examples	/examples	no
VOSI-availability	/availability	yes
VOSI-capabilities	/capabilities	yes

A stand-alone AccessData service may have one or both of the {sync} and {async} resources. For either type, it could have multiple resources (e.g. to support alternate authentication schemes). The AccessData service may also include other custom or supporting resources.

Either the {sync} or {async} AccessData capability may be included as part of other web services. For example, a single web service could contain the SIA-2.0 {query} capability, the DataLink-1.0 {links} capability, and the AccessData {sync} capability. Such a service must also have the VOSI-availability and VOSI-capabilities resources to report on and describe all the implemented capabilities.

2.1 {sync} resource

The {sync} resource is a synchronous web service resource that conforms to the DALI-sync description. The implementer is free to name (set the path) for this resource however they like; the client will find the resource path using the VOSI-capabilities resource.

The {sync} resource performs the data access as specified by the input parameters and returns the data directly in the output stream. Synchronous data access is suitable when the operations can be quickly performed and the data stream can be setup and written to (by the service) in a short period of time (e.g. before any timeouts).

2.2 {async} resource

The {async} resource is an asynchronous web service resource that conforms to the DALI-async description. The implementer is free to name (set the path) for this resource however they like; the client will find the resource path using the VOSI-capabilities resource.

The {async} resource performs the data access as specified by the input parameters and either (i) stores the results for later transfer or (ii) pushes the results to a specified destination (e.g. to a VOSpace location). Asynchronous data access usually introduces resource constraints on the service (which may be limited) and usually imposes a higher latency before any results can be seen because the location of results does not have to be valid until the data access job is complete. Asynchronous data access is intended for (but not limited to) use when the operations take considerable time and results must be staged (e.g. some multi-pass algorithms or operations that result in multiple outputs).

2.3 Examples: DALI-examples

AccessData services should provide a DALI-examples resource with one example invocation that shows the variety operations the service can perform. Example operations using the {sync} resource and that output a small data stream are preferred.

2.4 Availability: VOSI-availability

An AccessData web service must have a VOSI-availability resource [2] as described in DALI [1].

2.5 Capabilities: VOSI-capabilities

A web service that includes AccessData capabilities must have a VOSI-capabilities resource [2] as described in DALI [1]. The standardID for the {sync} resource is

```
ivo://ivoa.net/std/AccessData#sync
```

The standardID for the {async} resource is

```
ivo://ivoa.net/std/AccessData#async
```

All DAL services must implement the */capabilities* resource. The following capabilities document shows the minimal metadata for a stand-alone AccessData service and does not require a registry extension schema:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<vosi:capabilities
   xmlns:vosi="http://www.ivoa.net/xml/VOSICapabilities/v1.0"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:vod="http://www.ivoa.net/xml/VODataService/v1.1">
  <capability standardID="ivo://ivoa.net/std/VOSI#capabilities">
    <interface xsi:type="vod:ParamHTTP" version="1.0">
      <accessURL use="full">
         http://example.com/data/capabilities
      </accessURL>
    </interface>
  </capability>
  <capability standardID="ivo://ivoa.net/std/VOSI#availability">
    <interface xsi:type="vod:ParamHTTP" version="1.0">
      <accessURL use="full">
         http://example.com/data/availability
      </accessURL>
    </interface>
  </capability>
  <capability standardID="ivo://ivoa.net/std/AccessData#sync">
    <interface xsi:type="vod:ParamHTTP" role="std" version="1.0">
      <accessURL use="full">
           http://example.com/data/sync
      </accessURL>
    </interface>
    <!-- service details from extension schema could go here -->
  </capability>
  <capability standardID="ivo://ivoa.net/std/AccessData#async">
    <interface xsi:type="vod:ParamHTTP" role="std" version="1.0">
      <accessURL use="full">
           http://example.com/data/async
      </accessURL>
    </interface>
    <!-- service details from extension schema could go here -->
  </capability>
</vosi:capabilities>
```

Note that the {sync} and {async} resources do not have to be named as shown in the accessURL(s) above. Multiple capability elements for the {sync} and the {async} resources may be included; this is typically used if the differ in protocol

(http vs. https) and/or authentication requirements.

3 Parameters for {sync} and {async}

The {sync} and {async} resources accept the same set of parameters.

3.1 Common Parameters

3.1.1 REQUEST

The REQUEST parameter is a single-valued parameter. Specified value: getData.

3.1.2 ID

The ID parameter is used to specify the dataset or file to be accessed. The values for the ID parameter are generally discovered from data discovery or DataLink requests. The values must be treated as opaque identifiers that are used as-is. The DataLink specification [8] describes mechanisms for conveying opaque parameters and values in service descriptor resources that can be used by clients to set the ID parameter.

> *It would be feasible in many cases to do away with this parameter and leave it up to services to either generate URLs with the identifier already included; this would work for {sync} but would not really work for {async} since there is a common UWS joblist resource and no good place to embed custom parameters. The DataLink service descriptor resource could convey a custom name for the identifying parameter, which would work for both {sync} and {async} AccessData. However, it seems to be a common issue to require an identifier in such services so standardising the ID parameter seems like an obvious thing to do.*

3.2 Filtering Parameters

Filtering parameters are used to extract subsets of larger datasets or data files.

3.2.1 POS

The POS parameter defines the positional region(s) to be extracted from the data. The value is made up of a shape keyword followed by coordinate values; the coordinate values must be specified in the coordinate system and units of the data. The allowed shapes are:

Shape	Coordinate values
CIRCLE	<longitude> <latitude> <radius>
RANGE	<longitude1>/<longitude2> <latitude1>/<latitude2>
POLYGON	<longitude1> <latitude1> ... (at least 3 pairs)

Table 1: POS Values in Spherical Coordinates

A circle at (12,34) with radius 0.5:

```
POS=CIRCLE 12 34 0.5
```

A range of [12,14] in longitude and [34,36] in latitude:

```
POS=RANGE 12/14 34/36
```

A polygon from (12,34) to (14,34) to (14,36) to (12,36) and (implicitly) back to (12,34):

```
POS=POLYGON 12 34 14 34 14 36 12 36
```

The inside is always assumed to be the smaller of the region to the left and the region to the right so only polygons smaller than half the sphere can be specified.

A band around the equator:

```
POS=RANGE 0/360 -2/2
```

The north pole:

```
POS=RANGE 0/360 89/
```

This syntax is in the same style as STC-S, but with no reference positions, coordinate systems, units, or geometric operators like union, intersection, not, etc.

All longitude and latitude values (plus the radius of the CIRCLE) are expressed in degrees in the ICRS. A future version of this specification may allow the use of other reference systems (specifically the native system of ther data).

> *TODO: put an explicit and suitable reference system and STC ref here... Arnold?*

The POS parameter is multi-valued; multiple shapes can be included in a single request and all will be extracted.

> *Allowing multi-valued is not required for the base use cases from CSP and is technically challenging to implement (e.g. in a {sync} operation, the service may have to generate a multi-extension FITS*

file or tar file on-the-fly if faced with multiple distinct cutouts – in {async} it could just generate multiple output files). However, allowing multi-valued cutout parameters later would require a version number increment to 2.0... TBD. Also applies to BAND and TIME; the case for POL is quite different.

Option: we could restrict to single-value for {sync} and allow multi-valued for {async} since the latter easily handles the creation and delivery of multiple output files.

3.2.2 BAND

The BAND parameter defines the energy interval(s) to be extracted from the data. The value is an open or closed numeric interval of values in the native spectral axis coordinate system and units of the data. The intervals always include the bounding values.

If the range separator (/) is not present, the interval is assumed to be infinitely small (a scalar value).

The closed interval [500,550]:

```
BAND=500/550
```

The open interval (-inf,300]

```
BAND=/300
```

The open interval [750,inf)

```
BAND=750/
```

The scalar value 550, equivalent to [550,550]:

```
BAND=550
```

Extracting using a scalar value should normally extract a single pixel along the energy axis of the data; extracting using an interval should extract one or more pixels.

All energy values are expressed as barycentric wavelength in meters. A future version of this specification may allow the use of other reference systems (specifically the native system of ther data).

TODO: put an explicit and suitable reference system and STC ref here... Arnold?

The BAND parameter is multi-valued; multiple intervals can be included in a single request and all will be extracted.

See note above.

3.2.3 TIME

The TIME parameter defines the time interval(s) to be extracted from the data. The value is an open or closed interval with either numeric values (interpreted as Modified Julian Dates) or timestamp values as specified in DALI [1].

If the range separator (/) is not present, a numeric interval is assumed to be infinitely small (a scalar value) and a timestamp value is assumed to be an interval

For timestamp strings with only the date portion present, the time portion is interpreted to be 00:00:00 if the range separator is absent (scalar) or the value is at the lower bound of the interval and 23:59:59.999 if the value is at the upper bound of the interval.

An open interval from the beginning of 2012 and all later times:

```
TIME=2012-01-01/
```

All of 2012 (using the implied time portions above):

```
TIME=2012-01-01/2012-12-31
```

The following time intervals are equivalent (using the implied time portions above):

```
TIME=2012-01-01/2012-01-10
TIME=2012-01-01T00:00:00/2012-01-10T23:59:59.999
```

A range of MJD values:

```
TIME=55123.456/55123.466
```

An instant in time:

```
TIME=2012-01-02T12:34:56.789
```

An instant in time using Modified Julian Date:

```
TIME=55678.123456
```

The following scalar time instants are equivalent:

```
TIME=2012-01-02
TIME=2012-01-02T00:00:00
```

Time values are always UTC.

> *TODO: put an explicit and suitable reference system and STC ref here... Arnold?*

The TIME parameter is multi-valued; multiple intervals can be included in a single

request and all will be extracted.

See note above.

3.2.4 POL

The POL parameter defines the polarization state(s) (Stokes) to be extracted from the data.

Extract the unpolarized intensity:

```
POL=I
```

Extract the standard circular polarization:

```
POL=V
```

The POL parameter is multi-valued; multiple values can be included in a single request and all will be extracted. Extract only the IQU components:

```
POL=I
POL=Q
POL=U
```

Multi-valued POL parameter seems like it would be simpler to deal with for normal cubes with a FITS WCS Stokes axis... maybe it isn't though and POL really needs to support a range of states, eg POL=I/U would mean extract from I to U inclusive. Unfortunately, the ObsCore pol_states column also uses slashes to separate the values in the list of states... TBD.

3.2.5 COORD

IMPORTANT NOTE: These parameters (COORD and SELECT) have been added to support AccessData use cases from the TheoryIG in their development of SimDAL. Here we define two parameters that would perform the required filtering operations in the arbitrary n-dimensional datasets typical of theory. The syntax is consistent with the way ranges are handled above.

TBD: Keeping this means that we need a useful/rigid policy on what do do if a specified access parameter is not applicable in a specific service or for the identified data (e.g. the age-old silently-ignore policy). Maybe this pair of params is delayed to AccessData-1.1 or maybe SimDAL defines their own custom capability.

The COORD parameter is used to extract a range of values from an arbitrary coordinate axis. The value is made up of an axis name and a numeric interval. The axis names must be obtained from the detailed metadata for the dataset.

Extract from 20 to 40 along the foo axis:

```
COORD=foo 20/40
```

Extract from 20 to 40 in foo, bar larger than 50, and two slices in baz:

```
COORD=foo 20/40
COORD=bar 50/
COORD=baz 1/2
COORD=baz 8/9
```

The COORD parameter as defined here is limited to ranges of scalar values. It is intended for use with highly processed datasets that do not have normal physical axes that can be interpreted using the POS, BAND, TIME, and POL parameters, such as simulations (and it could be applied to tables or catalogues, although those might be better served via TAP [6]).

The scalar range parameters for observational data (BAND, TIME, POL) could be expressed using the COORD parameter. For example, the following could be equivalent:

```
BAND=500/550
COORD=WAVE 500/550
```

3.2.6 SELECT

The SELECT parameter is used to select a subset of the *observable* values (properties or attributes). Normally, observational data has a single value (typically an intensity) at each sampled coordinate (pixel) and the SELECT parameter is not used. However, for datasets produced from arbitrary computations, including theoretical simulations, there can be many properties for every sample (grid cell, particle for n-body, etc.); in this case the SELECT parameter lets the client extract a subset of the properties.

The value for the SELECT parameter is the name of the property to be extracted. The parameter is multi-valued, so extracting multiple properties requires the use of multiple parameters.

Extract the luminosity:

```
SELECT=luminosity
```

Extract several properties:

```
SELECT=temperature
SELECT=density
SELECT=pressure
SELECT=Fe_H
```

```
SELECT=HeIII
```

3.3 Transformation Parameters

Transformations will be defined in a future version of AccessData.

4 {sync} Responses

All responses from the {links} resource follow the rules for DALI-sync resources, except that the {links} response allows for error messages for individual input identifier values.

4.1 Successful Requests

Successfully executed requests should result in a response with HTTP status code 200 (OK) and a response in the format requested by the client or in the default format for the service.

If the values specified for cutout parameters do not include any pixels from the target datset/file, the service must respond with HTTP status code 204 (No Content) and no response body.

The service should set the following HTTP headers to the correct values where possible.

Content-Type	mime-type of the response
Content-Encoding	encoding/compression of the response (if applicable)

Table 2: Recommended HTTP Response Headers

Since the response is usually dynamically generated, the Content-Length and Last-Modified headers cannot usually be set.

4.2 Errors

The error handling specified for DALI-sync resources applies to service failure. Error documents should be text using the text/plain content-type and the text must begin with one of the following strings:

Error	General error (not covered below)
AuthenticationError	Not authenticated
AuthorizationError	Not authorized to access the resource
ServiceUnavailable	Transient error (could succeed with retry)
UsageError	Permanent error (retry pointless)

5 {async} Responses

The {async} resource conforms to the DALI-async resource description, which means the job is a UWS job with all the job control features available. All result files are to be listed as children of the UWS results resource. The service provider is free to name each result.

6 Changes

6.1 WD-AccessData-1.0-20140312

This is the initial document.

7 References

[1] P. Dowler, M. Demleitner, M. Taylor, D. Tody *Data Access Layer Interface* 1.0, IVOA Recommendation 29 November 2013.
http://www.ivoa.net/std/DALI/

[2] M. Graham & G. Rixon (ed.), GWS-WG, *IVOA Support Interfaces Version 1.0*, IVOA Recommendation, 31 May 2011.
http://www.ivoa.net/Documents/VOSI/

[3] Y. Shafranovich, *Common Format and MIME Type for Comma-Separated Values (CSV) Files*, IETF RFC 4180.
http://www.ietf.org/rfc/rfc4180.txt

[4] IANA, *MIME Media Types*,
http://www.iana.org/assignments/media-types/text/tab-separated-values

[5] F. Ochsenbein, M. Taylor (ed.), R. Williams, C. Davenhall, M. Demleitner, D. Durand, p. Fernique, D. Giaretta, R. Hanisch, T. McGlynn, A. Szalay, A. Wicenec *VOTable Format Definition Version 1.3*, IVOA Recommendation 20 September 2-13.
http://www.ivoa.net/Documents/VOTable/

[6] P. Dowler, G. Rixon, D. Tody, DAL-WG, *Table Access Protocol Version 1.0,* IVOA Recommendation 27 March 2010.
http://www.ivoa.net/Documents/TAP/1.0

[7] M. Louys, F. Bonnarel, D. Schade, P. Dowler, A. Micol, D. Durand, D. Tody, L. Michel, J. Salgado, I. Chilingarian, B. Rino, J. De Dios Santander, P. Skoda, Observation Data Model Core Components and its Implementation in the *Table Access Protocol Version 1.0,* IVOA Recommendation 28 October 2011.
http://www.ivoa.net/Documents/ObsCore/

[8] P. Dowler, F. Bonnarel, L. Michel, T. Donaldson, D. Languignon, M. Demleitner, *DataLink Version 1.0*, IVOA Working Draft 29 February 2014.
http://www.ivoa.net/Documents/DataLink/