



*International  
Virtual  
Observatory  
Alliance*

## IVOA DataLink

### Version 1.1

#### IVOA Working Draft 2019-07-21

Working group

DAL

This version

<http://www.ivoa.net/documents/DataLink/20190721>

Latest version

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

Previous versions

This is the first public release

Author(s)

Patrick Dowler, François Bonnarel, Laurent Michel, Markus Demleitner, Mark Taylor

Editor(s)

Patrick Dowler

## Abstract

This document describes the linking of data discovery metadata to access to the data itself, further detailed metadata, related resources, and to services that perform operations on the data. The web service capability supports a drill-down into the details of a specific dataset and provides a set of links to the dataset file(s) and related resources. This specification also includes a VOTable-specific method of providing descriptions of one or more services and their input(s), usually using parameter values from elsewhere in the VOTable document. Providers are able to describe services that are relevant to the records (usually datasets with identifiers) by including service descriptors in a result document.

## Status of this document

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

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

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	The Role in the IVOA Architecture . . . . .	5
1.2	Motivating Use Cases . . . . .	6
1.2.1	Multiple Files per Dataset . . . . .	6
1.2.2	Progenitor Dataset . . . . .	6
1.2.3	Alternate Representations . . . . .	6
1.2.4	Standard Services . . . . .	7
1.2.5	Free or Custom Services . . . . .	7
1.2.6	Access Data Services . . . . .	7
1.2.7	Recursive DataLink . . . . .	8
1.2.8	Datasets linked to an astronomical source . . . . .	8
1.2.9	Metadata and data related to provenance entities . . . . .	8
<b>2</b>	<b>Resources</b>	<b>8</b>
2.1	{links} endpoint . . . . .	9
2.1.1	ID . . . . .	9
2.1.2	RESPONSEFORMAT . . . . .	10
2.2	Availability: VOSI-availability . . . . .	10
2.3	Capabilities: VOSI-capabilities . . . . .	10
<b>3</b>	<b>{links} Response</b>	<b>11</b>
3.1	DataLink MIME Type . . . . .	11
3.2	List of Links . . . . .	12
3.2.1	ID . . . . .	13
3.2.2	access_url . . . . .	13
3.2.3	service_def . . . . .	13
3.2.4	error_message . . . . .	13
3.2.5	description . . . . .	14
3.2.6	semantics . . . . .	14
3.2.7	content_type . . . . .	15

3.2.8	content_length . . . . .	15
3.3	Successful Requests . . . . .	15
3.3.1	VOTable output . . . . .	15
3.3.2	Other Output Formats . . . . .	16
3.4	Errors . . . . .	16
<b>4</b>	<b>Service Descriptors</b>	<b>17</b>
4.1	Service Resources . . . . .	17
4.1.1	Input PARAMETERS . . . . .	18
4.1.2	Additional templated endpoints for RESTful services . . . . .	18
4.2	Example: Service Descriptor for the {links} Capability . . . . .	19
4.3	Example: Service Descriptor for an SIA-1.0 Service . . . . .	20
4.4	Example: Service Descriptor for VOSpace-2.0 . . . . .	21
4.5	Example: Custom Access Data Service . . . . .	22
4.6	Example: Self-Describing Service . . . . .	24
<b>5</b>	<b>New “datalink” content-type for the LINK element in VOTable</b>	<b>26</b>
<b>Appendices</b>		<b>26</b>
<b>Appendix A</b>	<b>Recognizing a {links}RESOURCE URL in various context</b>	<b>26</b>
A.1	DataLink and SIAP-2.0 or ObsTAP services . . . . .	26
A.1.1	DataLink discovery via format and reference columns . . . . .	27
A.1.2	SIAP-2.0, ObsTAP and service descriptors . . . . .	27
A.2	DataLink in the context of other dataset discovery methods . . . . .	27
A.3	DataLink outside Data discovery context . . . . .	28
A.4	Various recognition solutions for use cases introduced in section A.2 and A.3 . . . . .	29
<b>Appendix B</b>	<b>Changes</b>	<b>29</b>
B.1	WD-DataLink-1.1-20190721 . . . . .	30
B.2	PR-DataLink-1.0-20150413 . . . . .	30
B.3	PR-DataLink-1.0-20140930 . . . . .	31
B.4	PR-DataLink-20140530 . . . . .	31
B.5	WD-DataLink-20140505 . . . . .	31
B.6	WD-DataLink-20140212 . . . . .	32

## Acknowledgments

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

## Conformance-related definitions

The words “MUST”, “SHALL”, “SHOULD”, “MAY”, “RECOMMENDED”, and “OPTIONAL” (in upper or lower case) used in this document are to be interpreted as described in IETF standard RFC2119 (Bradner, 1997).

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

## 1 Introduction

This specification defines mechanisms for connecting metadata about discovered **VO items** to related data products, and web services

The *links* web service capability is a web service capability for drilling down from a discovered **item such as an identifier, a source in a catalog or any other data item. In the first case** (typically an IVOA publisher dataset identifier) **it allows** to find details about the data files that can be downloaded, alternate representations of the data that are available, and services that can act upon these data (usually without having to download the entire dataset). **In the latter cases it allows to associate metadata, images, cubes, spectra, timeseries or ancillary data to a source in the sky or other kind of time.** The expected usage **for datasets** is for DAL (Data Access Layer) data discovery services (e.g. a TAP service (Dowler and Rixon et al., 2010) with the ObsCore (Louys and Tody et al., 2017) data model or one of the simple DAL services) to provide an identifier that can be used to query the associated DataLink capability. The DataLink capability will respond with a list of links that can be used to access the data. Here we specify the calling interface for the capability and the response, which lists the links and provides both concrete metadata and a semantic vocabulary so clients can decide which links to use.

The *service descriptor resource* uses the metadata features of VOTable to embed service metadata along with tabular data, such as would be obtained by querying a simple DAL data discovery service or a TAP service. This service metadata tells the client how to invoke a service and, for those registered in an IVOA registry, how to lookup additional information about the service. The service provider can use this mechanism to tell clients about services that can be invoked to access the discovered **item** in some way: get additional metadata, download the data, or invoke services that act upon the data files. These services may be IVOA standard services or custom services from the data providers. The current version provides no way to describe

the output of a service, but this may be added in a future (minor) revision of this specification.

*(suppressed text : We expect ...responses.)*

## 1.1 The Role in the IVOA Architecture

DataLink is a data access protocol in the IVOA architecture whose purpose is to provide a mechanism to link resources found via one service to resources provided by other services.

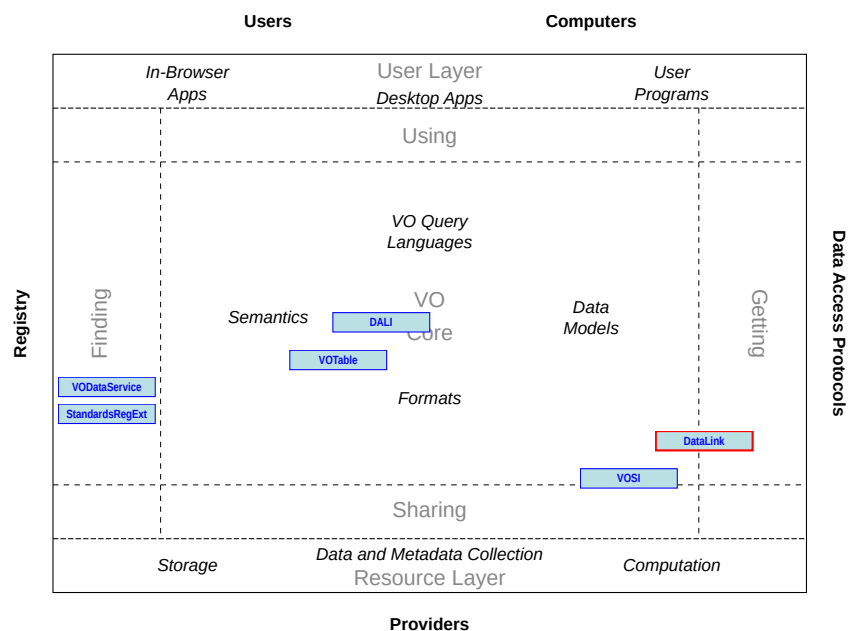


Figure 1: Architecture diagram for this document

Although not shown in Figure 1, any implementation of an access protocol could make use of DataLink to expose resources. DataLink services conform to the Data Access Layer Interface specification (DALI, Dowler and Demleitner et al. (2013)), including the Virtual Observatory Support Interfaces resources (VOSI, Graham and Rixon et al. (2011)). DataLink services use VOTable (Ochsenbein and Taylor et al., 2013) as the default output format both for successful output and to return error documents.

DataLink specifies a standardID for itself, as defined in VODataService (Plante and Stébé et al., 2010), to be used in a StandardsRegExt record (Harrison and Burke et al., 2012). It also specifies how to include standardID values in the response to describe links to services.

DataLink includes a description of how data discovery services can include the link to the associated DataLink service in VOTable. VOTable is

also the default output format for the DataLink web service capability.

## 1.2 Motivating Use Cases

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

### 1.2.1 Multiple Files per Dataset

It is very common for a single dataset to be physically manifest as multiple files of various types. With a DataLink web service, the client can drill down using a discovered dataset identifier and obtain links to download one or more data files. For static data files, the DataLink service will be able to provide a URL as well as the content-type and content-length (file size) for each download.

### 1.2.2 Progenitor Dataset

In some cases, the data provider may wish to provide one or more links to progenitor (input) datasets; this would enable the users to drill down to input data in order to better understand the content of the product dataset, possibly reproduce the product to evaluate the processing, or reprocess it with different parameters or software.

### 1.2.3 Alternate Representations

For some datasets (large ones) it is useful to be able to access preview data (either precomputed or generated on-the-fly) and use it to determine if the entire dataset should be downloaded (e.g. in an interactive session). A DataLink service can provide links to previews as a URL with a specific relationship to the dataset and include other metadata like content-type (e.g. image/png) and content-length to assist the client in selecting a preview; multiple previews with different sizes (content-length) could be returned in the list of links. Plots derived from the dataset could also be linked as previews. Some previews may be of the same content-type as the complete dataset, but reduced content in some fashion (e.g. a representative image or spectrum derived from a large data cube).

Links to alternate representations may be to pre-generated resources or may be computed on the fly, using either an opaque URL or a custom parameterised service (see 1.2.5 below).

Other alternate representations that are not previews could also be included in the list of links. For example, one could provide an alternate

download format for a data file with different content-type (e.g. FITS and HDF).

#### 1.2.4 Standard Services

Data providers often implement services that can access a dataset or its files using standard service interfaces or provide alternate representations of the dataset. For example, the links for a dataset discovered via a TAP service could be to an SSA (Tody and Dolensky et al., 2012) service, allowing the caller to get an SSA query response that describes the same dataset with metadata specific to the SSA service.

Providers should be able to link to current and future data access services that perform filtering and transformations as these services are defined and implemented (without requiring a new DataLink specification). For IVOA standard services, the DataLink response would use the VODataService standardID as the service type to tell the client which standard (and version) the linked service complies to. The client can select services they understand and use the link to invoke the service (with additional service parameters added by the client).

#### 1.2.5 Free or Custom Services

Data providers often implement custom services that can access a dataset or its files or provide alternate representations of the dataset. The availability of such services should be conveyed to clients/users in the same fashion as for standard services. This allows services defined within the VO to be used in conjunction with services defined outside the VO to deliver features to users.

#### 1.2.6 Access Data Services

In many access scenarios, server-side processing of data is highly desirable, typically to reduce the amount of data to be transferred. Examples for such operations are cutouts, slicing of cubes, and re-binning to a coarser grid. Other examples for server-side operations include on-the-fly format conversion or recalibration. For the purpose of this specification, we call such services *access data services*. DataLink should let providers declare such access data services in a way that a generic client can discover what operations are supported, their semantics, and the domains of the operations' parameters. This lets clients operate multiple independent access services behind a common user interface, allowing scenarios like “give me all voxels around positions X in wavelength range Y of all spectral cubes from services Z\_1, Z\_2, and Z\_9”.

Access data services may also be standard services or custom services; at the time of writing, the definition of standard access service capabilities is in progress in separate specifications.

### 1.2.7 Recursive DataLink

In some cases, a dataset may contain many files (as in 1.2.1 above) and the provider may wish to make some files directly accessible and other (less important) files only accessible via additional calls. Such organisation of links could be accomplished by including a link to another DataLink service in the initial DataLink response (e.g. recursive DataLink). This service link would be described with both a service type (as in 1.2.4) and content type.

### 1.2.8 Datasets linked to an astronomical source

There are a lot of catalogs of astronomical sources made available using VO services such as ConeSearch (Plante and Williams et al., 2008) or TAP services. For some catalogs “associated data” are available. These data include images from which sources have been extracted, or imaging the object in case of extended objects, as well as additional observations such as Spectra or Time Series of the source and even spectral cubes and Time Series of images for extended or varying objects. The {links}response obtained for the source id can allow to easily retrieve all these associated data in one shot

### 1.2.9 Metadata and data related to provenance entities

The IVOA Provenance datamodel (Servillat and Riebe et al., 2019) represents metadata tracing the history of data. This information can be retrieved through ProvTAP (Bonnarel, 2019) or ProvSAP (Riebe, 2018) DAL services. The Entity instances represent the state of the data items between various steps of the data processing flow. “Entities” can be hooked to lists of real datasets retrieval and previews, or Obscore, SimDM and SSA metadata describing them using the {links}endpoint.

## 2 Resources

The DataLink web service capability is implemented as an HTTP REST (Richardson and Ruby, 2007) web service capability that conforms to the DALI-sync resource description. The {links} endpoint is described fully below (see 2.1). The values for the identifiers are typically found using a data discovery service (e.g. TAP with ObsCore). Data discovery services that work with an associated DataLink service should include an endpoint



in the discovery response to describe the DataLink service. The mechanism for doing this is described below (see 4).

The requirements for a standalone DataLink service are given in the table below.

endpoint type	endpoint name	required
{links}	service specific	yes
VOSI-availability	/availability	yes
VOSI-capabilities	/capabilities	yes

A standalone DataLink service must have at least one {links} endpoint; it could have multiple {links} endpoints (e.g. to support alternate authentication schemes). Alternatively, the {links} endpoint may be embedded in another web service, in which case the VOSI-capabilities resource of that service would also describe the {links} capability.

## 2.1 {links} endpoint

The {links} endpoint is a synchronous web service resource that conforms to the DALI-sync description. The implementer is free to name this resource however they like as long as the {links} endpoint is a sibling of the VOSI resources; this restriction allows a client to construct the URL to VOSI resources from any {links} URL and thus discover other capabilities or check the availability if there is a failure. For example, a DataLink service could have:

```

http://example.com/datalink/links — anonymous access
http://example.com/datalink/auth-links — HTTP authentication
https://example.com/datalink/links — IVOA SSO authentication

```

As a DALI-sync resource, the parameters for a request may be submitted using an HTTP GET (query string) or POST action.

### 2.1.1 ID

The ID parameter is used by the client to specify one or more identifiers. The service will return at least one link for each of the specified values. The ID values are found in data discovery services and may be readable URIs or opaque strings.

If an ID value specified by the client is not understandable by the DataLink service, the service must include a single link in the output with the ID and an error message (see below).

If the client submits more ID values than a service is prepared to process, the service should process ID values up to the limit and **must** include an

overflow indicator in the output as described in DALI. The service **must not** truncate the output within the set of rows (links) for a single ID value if the request exceeds such an input limit.

If the client submits no ID values, the service must respond with a normal response (e.g. an empty results table for VOTable output). The service may include service descriptors (see 4) for related services and a service descriptor describing itself (see 4.6).

### 2.1.2 RESPONSEFORMAT

The RESPONSEFORMAT parameter is described in DALI; support for RESPONSEFORMAT is mandatory.

The only output format required by this specification is VOTable with TABLEDATA serialization; services must support this format. Clients that want to get the standard (VOTable) output format should simply ignore this parameter.

To comply with this standard, a {links} **endpoint** only needs to strip off MIME type parameters and understand the following:

- no RESPONSEFORMAT
- RESPONSEFORMAT=votable
- RESPONSEFORMAT=application/x-votable+xml

All of these result in the standard output format.

Service implementers may support additional output formats but must follow the DALI specification if they chose any formats described there.

## 2.2 Availability: VOSI-availability

A DataLink web service must have a VOSI-availability **endpoint**, as defined by DALI and VOSI.

## 2.3 Capabilities: VOSI-capabilities

A standalone DataLink web service must have a VOSI-capabilities **endpoint** as defined by DALI and VOSI. The standardID for the {links} **endpoint** is

```
ivo://ivoa.net/std/DataLink#links-1.0
```

The following capabilities document shows the minimal metadata and does not require a registry extension schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<vos:capabilities
  xmlns:vos="http://www.ivoa.net/xml/VOSICapabilities/v1.0"
```

```

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:vs="http://www.ivoa.net/xml/VODDataService/v1.1">
...
<capability standardID="ivo://ivoa.net/std/DataLink#links-1.0">
  <interface xsi:type="vs:ParamHTTP" role="std" version="1.0">
    <accessURL use="base">
      http://example.com/datalink/mylinks
    </accessURL>
    <queryType>GET</queryType>
    <queryType>POST</queryType>
    <resultType>
      application/x-votable+xml;content=datalink
    </resultType>
    <param std="true" use="required">
      <name>ID</name>
      <description>publisher dataset identifier</description>
      <ucd>meta.id;meta.main</ucd>
      <dataType>string</dataType>
    </param>
  </interface>
</capability>
</vosi:capabilities>

```

Multiple capability elements for the {links} **endpoints** may be included; this is typically used if they differ in protocol (http vs. https) and/or authentication requirements.

The {links} capability may also be included as a resource in another service, in which case the VOSI-Capabilities for that service would describe all the capabilities of that service, including {links}.

### 3 {links} Response

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

#### 3.1 DataLink MIME Type

In some data discovery responses (e.g. ObsCore, Louys and Tody et al. (2017)), there are columns with a URL (access\_url in ObsCore) and a content type (access\_format in ObsCore). If the implementation uses a DataLink service to implement this data access, it should include a complete (including the ID parameter) DataLink URL and a parameterised VOTable MIME type:

```
application/x-votable+xml;content=datalink
```

to denote that the response from that URL is a DataLink response. This is also the MIME type for the {links} response (see 3.3) unless the caller has explicitly requested a specific value via the RESPONSEFORMAT parameter (see 2.1.2). Services may include other MIME type parameters in the response.

### 3.2 List of Links

The list of links that is returned by the {links} endpoint can be represented as a table with the following columns:

name	description	required	UCD
ID	Input identifier	yes	meta.id;meta.main
access_url	link to data or service	one only	meta.ref.url
service_def	reference to a service descriptor resource		meta.ref
error_message	error if an access_url cannot be created		meta.code.error
description	human-readable text describing this link	yes	meta.note
semantics	Term from a controlled vocabulary describing the link	yes	meta.code
content_type	mime-type of the content the link returns	no	meta.code.mime
content_length	size of the download the link returns	no	phys.size;meta.file

Table 1: Required Fields for Links

All fields must be present in the output table; values must be provided (or null) as described in Table 1. Each row in the table represents one link and must have exactly one of:

- an access\_url
- a service\_def
- an error\_message

If an error occurs while processing an ID value, there should be at least one row for that ID value and an error\_message. For example, if an input ID value is not recognised or found, one row with an error\_message to that effect is sufficient. If some links can be created (e.g. download links) but others

cannot due to some temporary failure (e.g. service outage), then one could have one or more rows with the same ID and different error\_message(s).

Services may include additional columns; this can be used to include values that can be referenced from service descriptor input parameters (see 4.1).

### 3.2.1 ID

The ID column contains the input identifier value.

### 3.2.2 access\_url

The access\_url column **may contain** a URL to download a single resource. *(suppressed text)* This URL can be a link to a dynamic resource (e.g. preview generation).

Beside dereferencable URL, it is allowed to use URI-fragments to link the intial resource to a specific part of the retrievable resource, with its specific semantics and description. Exemples of this are section in an html page or pathes in an archive file or extensions in a MEF. The interpretation of the fragment will depend from the content type of the retrievable resource. No other additional parameters or client handling are allowed.

### 3.2.3 service\_def

The service\_def column contains a reference from the result row to a separate resource. This resource describes a service as specified in section 4. For example, if the response document includes this resource to describe a service:

```
<RESOURCE type="meta" utype="ad hoc:service" ID="srv1">
...
</RESOURCE>
```

then the service\_def column would contain *srv1* to indicate that a resource with XML ID *srv1* in the same document describes the service. Note that service descriptors do not always require an XML ID value; it is only the reference from service\_def that warrants adding an ID to the descriptor.

### 3.2.4 error\_message

The error\_message column is used when no accessURL can be generated for an input identifier. If an error\_message is included in the output, the only other columns with values should be the ID column and the semantics column; all others should be null.

### 3.2.5 description

The description column **SHOULD** contain a human-readable description of the link; it is intended for display by interactive applications **and very important to help user distinguish links with same semantics (see below)**.

### 3.2.6 semantics

The semantics column contains a single term from an external RDF vocabulary that describes the meaning of this linked resource relative to the identified dataset. The semantics column is intended to be machine-readable and assist automating data retrieval and processing.

The core DataLink vocabulary defines a special term for the concept of *this*; this term is used to describe links used for retrieval of the dataset file(s). Since null values are not permitted, the semantics value in cases where only an `error_message` is supplied should be the most appropriate for the link the service was trying to generate.

The value is always interpreted as a URI; if it is a relative URI, it is resolved (Berners-Lee and Fielding et al., 2005) against the base URI of the core DataLink vocabulary:

```
http://www.ivoa.net/rdf/datalink/core
```

The value used in the semantics column is normally the URI of the vocabulary, followed by a fragment (`#`), followed by a predicate from the specified vocabulary. For example, if the `{links}` table contains a link to a preview of a dataset, the `ID` column will contain the dataset identifier, the `access_url` column will contain the URL to the preview, and the semantics column could contain this predicate:

```
http://www.ivoa.net/rdf/datalink/core#preview
```

or this relative URI:

```
#preview
```

For predicates outside the core DataLink vocabulary, the full URI is required.

The core DataLink vocabulary is published at:

```
http://www.ivoa.net/rdf/datalink/core
```

and the latest version is available as a human readable document and an RDF XML document. Services are encouraged to use the core vocabulary as much as possible, but may use a custom vocabulary as long as they use a custom vocabulary namespace (base URI); the base URI should be resolvable to a human-readable document describing the terms.

### 3.2.7 content\_type

The content\_type column tells the client the general file format (mime-type) they will receive if they use the link (access\_url or invoking a service). For recursive DataLink links, the content\_type value should be as specified in section 3.1. This field may be null (blank) if the value is unknown.

### 3.2.8 content\_length

The content\_length column tells the client the size of the download if they use the link, in bytes. For VOTable, the FIELD must be datatype="long" with unit="byte". The value may be null (blank) if unknown and will typically be null for links to services.

## 3.3 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. The content of the response (for tabular formats) is described above, with some additional details below.

Unless the incoming request included a RESPONSEFORMAT parameter requesting a different format, the content-type header of the response MUST be "application/x-votable+xml" with the "content" parameter set to "datalink", with the canonical form given in 3.1 strongly recommended. Contrary to all other uses of the string given in 3.1, clients wishing to evaluate the content type of the response must, however, perform a full parse of header value. This specification cannot and does not outlaw content types with additional parameters (e.g. "application/x-votable+xml; content=datalink;charset=iso-8859-1") or with extra spaces or quotes (as allowed for MIME types, Freed and Borenstein (1996)).

If the incoming request includes a DALI RESPONSEFORMAT parameter, content-type follows the DALI rules.

### 3.3.1 VOTable output

The table of links **must** be returned in a RESOURCE with type="results". The table **must** be in TABLEDATA serialization unless another serialization is specifically requested (see 2.1.2) and supported by the implementation. The name attribute for FIELD elements in the VOTable (and the units in one case) are specified above (see 3.2).

**StandardID INFO element** : DALI specification states (reference) that such an INFO element with name standardID and the actual standardID string as a value SHOULD be provided. It is recommended to include such

an element to help users and applications to identify VOTables as results of DataLink services this way :

```
<RESOURCE type="results"
...
<INFO name="standardID"
value="ivo://ivoa.net/std/DataLink#links-1.?"/>
...
</RESOURCE>
```

### 3.3.2 Other Output Formats

This specification does not describe any other output formats, but allows (via the RESPONSEFORMAT in section 2.1.2) implementations to provide output in other formats.

## 3.4 Errors

The error handling specified for DALI-sync resources applies to service failure (where no links can be generated) and to the usage error where no ID parameter is specified. Services should return the document format requested by the client (see 2.1.2). For the standard output format (VOTable) the error document **must** also be VOTable.

For errors that occur while generating individual links, each identifier may result in a link with only an error\_message as described above. In either case (error document or per-link error\_message), the error message must start with one of the strings in Table 2, in order of specificity.

Error	Meaning
NotFoundFault	Unknown ID value
UsageFault	Invalid input (e.g. invalid ID value)
TransientFault	Service is not currently able to function
FatalFault	Service cannot perform requested action
DefaultFault	Default failure (not covered above)

Table 2: Error Messages

In all cases, the service may append additional useful information to the error strings above. If there is additional text, it must be separated from the error string with a colon (:) character, for example:



NotFoundFault: ivo://example.com/data?foo cannot be found

UsageFault: foo:bar is invalid, expected an ivo URI

## 4 Service Descriptors

The DataLink service interface is designed to add functionality to data discovery services by providing the connection between the discovered datasets and the download of data files and access to services that act on the data. When the {links} capability returns links to services, the response document also needs to describe the services so that clients can figure out how to invoke them. This is done by including an additional metadata resource in the response document to describe each type of service that can be used.

The same mechanism can also be used in any VOTable document, such as a data discovery response from a TAP query or one of the simple DAL query protocols, to enable clients to find and use the {links} capability itself.

Here we describe how to construct a resource that describes a service and add it to a VOTable document. The mechanism is general and can be used wherever a VOTable document is created.

### 4.1 Service Resources

In a data discovery response, one RESOURCE element (usually the first) will have an attribute type="results" and tabular data; this resource contains the query result.

To describe an associated service, the VOTable would also contain one or more resources with attribute type="meta" and utype="ad hoc:service". **A free name SHOULD be added to each descriptor RESOURCE to help identifying the meaning.**

**In case a response contains several "descriptor" RESOURCES and several "results" RESOURCES these RESOURCES should be nested in order to allow correct association.** A resource of this type has no tabular data, but may include a rich set of metadata. The utype attribute makes it easy for clients to find the RESOURCE elements that describe services. **A service RESOURCE SHOULD contain a DESCRIPTION element to explain the purpose of the service.** A service resource contains PARAM elements to describe the service and GROUP elements with additional PARAM elements to describe the input parameters **or additional RESTful endpoints templates.** The standard PARAM elements for a *service* resource are described in Table 3.

For services that implement an IVOA standard, the standardID is specified as the value attribute of the PARAM with name="standardID". For free or custom services, this PARAM is not included.

name	value	required
accessURL	URL to invoke the capability	yes
standardID	URI for the capability	no
resourceIdentifier	IVOA registry identifier	no
contentType	Media type of the service response	no

Table 3: Service Resource Parameters

For registered services, the resourceIdentifier PARAM allows the client to query an IVOA registry for complete resource metadata. This could be used to find documentation, contact info, etc. Although they need not be, free or custom services could be registered in an IVOA registry and thus have a resourceIdentifier to enable lookup of the record.

For standard services, the value of the accessURL PARAM must be the accessURL for the capability specified by the standardID. The accessURL is not generally usable as-is; the client must include extra parameters as described below. If a standardID indicates a capability that supports multiple HTTP verbs (GET, POST, etc.), the client may use any supported verbs. Otherwise, there is no way in this version to specify that POST (for example) is supported so clients should assume that only HTTP GET may be used. Since the accessURL may contain parameters; clients must parse the URL to decide how to append additional parameters when invoking the service.

#### 4.1.1 Input PARAMETERS

A GROUP with name="inputParams" contains PARAM elements describing how to invoke the service. For services where the parameter values come from columns in the results resource, we use the ref attribute of the PARAM to indicate the FIELD (column) with the values. Other PARAM elements (without a ref attribute) are also allowed; these would describe additional service parameters, the type of value that must be specified, the meaning (UCD) of the value they apply to, etc.

#### 4.1.2 Additional templated endpoints for RESTful services

In order to describe RESTful services endpoints (eg VOSPACE endpoints) with variable path segments the value of which can be read in FIELDS of the main "results" TABLE, a GROUP with name = "endpoints" contains PARAMS utype="template" defining the relative templated URLs. The templating scheme adopted in the value attribute of this PARAM is following the appropriate IETF RFC (Gregorio and Fielding et al., 2012)

## 4.2 Example: Service Descriptor for the {links} Capability

The {links} capability can be used with a result table when one of the columns contains identifier values that can be used with the ID parameter (see 2.1.1). In order for the service resource to refer to this FIELD, the FIELD element describing this column of the table **must** include an XML ID attribute that uniquely identifies the FIELD (column). For example, a response following the ObsCore-1.0 data model would use the following:

```
<FIELD name="obs_publisher_did" ID="primaryID"
      utype="obscure:Curation.PublisherDID"
      ucd="meta.ref.url;meta.curation"
      xtype="adql:VARCHAR" datatype="char" arraysize="256*" />
```

where the ID value *primaryID* is arbitrary. This FIELD would typically be found within the RESOURCE of type="results". The same VOTable document would have a second RESOURCE with type="meta" to describe the associated DataLink {links} capability.

The {links} capability described in section 2.1 is described by the following resource:

```
<RESOURCE type="meta" utype="adhoc:service" name="{links} for
↳ Obscore">
  <DESCRIPTION>Links resources to datasets</DESCRIPTION>
  <PARAM name="standardID" datatype="char" arraysize="*"
        value="ivo://ivoa.net/std/DataLink#links-1.0" />
  <PARAM name="accessURL" datatype="char" arraysize="*"
        value="http://example.com/mylinks" />
  <PARAM name="contentType" datatype='char' arraysize='*'
↳ value="application/x-votable+xml;content=datalink" >
  <GROUP name="inputParams">
    <PARAM name="ID" datatype="char" arraysize="*"
          value="" ref="primaryID"/>
  </GROUP>
</RESOURCE>
```

Clients that want to find services to operate on the results would look for resources with type="meta" and utype="adhoc:service". They would find a DataLink service specifically via the PARAM with name="standardID". To call the service, the GROUP contains a PARAM with the service parameter name and a ref attribute whose value is the XML ID attribute on a FIELD. In the example above, the ref="primaryID" refers to the FIELD with ID="primaryID" in the same document (usually the result table). The URL to call the service would be:

```
http://example.com/datalink/mylinks?ID=<obs_publisher_did value>
```

Although this version of DataLink only has one parameter (ID), using a GROUP and providing the service parameter name allows this recipe to be used with any service and (with the GROUP) with multi-parameter services.

In the above example, the {links} capability is not registered in an IVOA registry so there is no resourceIdentifier PARAM included in the descriptor.

### 4.3 Example: Service Descriptor for an SIA-1.0 Service

Suppose you have an SIA-1.0 (Harrison and Tody et al., 2009) service and you want users to be able to call it to get SIA-1.0 specific metadata. This VOTable RESOURCE describes the basic query interface of SIA-1.0:

```
<RESOURCE type="meta" utype="ad hoc:service" name="My SIA
↪ service">
  <DESCRIPTION> This is service SIA for so and
  ↪ so</DESCRIPTION>
  <PARAM name="resourceIdentifier" datatype="char"
  ↪ arraysize="*"
    value="ivo://example.com/mySIA" />
  <PARAM name="standardID" datatype="char" arraysize="*"
    value="ivo://ivoa.net/std/SIA#1.0" />
  <PARAM name="accessURL" datatype="char" arraysize="*"
    value="http://example.com/sia/query" />
  <PARAM name="contentType" datatype="char" arraysize="*"
  ↪ value="application/x-votable+xml" />
  <GROUP name="inputParams">
    <PARAM name="POS" datatype="char" arraysize="*"
      value="" />
    <PARAM name="SIZE" datatype="char" arraysize="*"
      value="0.5" />
    <PARAM name="VERB" datatype="int" value="0" />
    <PARAM name="FORMAT" datatype="char" arraysize="*"
      value="ALL">
      <VALUES>
        <OPTION value="ALL" />
        <OPTION value="image/fits" />
        <OPTION value="METADATA" />
      </VALUES>
    </PARAM>
  </GROUP>
</RESOURCE>
```

If this SIA service supported querying specific data collections via a custom parameter named COLLECTION, the following PARAM describes the

custom parameter, including the possible values:

```
<PARAM name="COLLECTION" datatype="char" arraysize="*"
      value="ALL">
  <VALUES>
    <OPTION value="ALL" />
    <OPTION value="FOO" />
    <OPTION value="BAR" />
  </VALUES>
</PARAM>
```

This PARAM would be added to the GROUP name="inputParams" of the service description.

#### 4.4 Example: Service Descriptor for VOspace-2.0

VOspace-2.0 is a RESTful web service with several capabilities. Each of these capabilities can be described with a service descriptor; this would save the client having to perform a registry lookup to find and use the service. The descriptors cannot describe the path usage and XML document based input to the service, but they can describe the optional parameters:

```
<RESOURCE type="meta" utype="ad hoc:service" ID="vnodes"
  ↪ "name='My VOSPACE'>
<DESCRIPTION>A nice container to share space</DESCRIPTION>
  <PARAM name="resourceIdentifier"
        value="ivo://example.com/vospace" />
  <PARAM name="standardID"
        value="ivo://ivoa.net/std/VOspace/v2.0#nodes" />
  <PARAM name="accessURL"
        value="http://example.com/vospace/nodes" />
  <GROUP name="inputParams">
    <PARAM name="detail" datatype="char" arraysize="*"
          value="min"/>
    <PARAM name="limit" datatype="integer"
          value="1000"/>
    <PARAM name="uri" datatype="char" arraysize="*"
          value=""/>
  </GROUP>
</RESOURCE>
<RESOURCE type="meta" utype="ad hoc:service" ID="vtrans">
  <PARAM name="resourceIdentifier"
        value="ivo://example.com/vospace" />
  <PARAM name="standardID"
        value="ivo://ivoa.net/std/VOspace/v2.0#transfers"
  ↪ />
```

```

    <PARAM name="accessURL"
          value="http://example.com/vospace/transfers" />
  </RESOURCE>

```

Since the capability being described is RESTful and inputs are *[?fixed?]*, the caller must recognise the standardID values and use a VOSpace-aware client to call the service.

## 4.5 Example: Custom Access Data Service

Parameters for custom access data services can be described such that clients can figure out how to call the service and even create a basic form-based user interface. The following Rotatable resource describes a custom spectral cutout service:

```

<RESOURCE type="meta" utype="ad hoc:service" ID="apoadimo"
  ↪ name='Custom Recalibration'>
<DESCRIPTION>Custom APOADIMO spectrum recalibration
  ↪ service</DESCRIPTION>
  <PARAM arraysize="*" datatype="char" name="accessURL"
    ↪ ucd="meta.ref.url"
    ↪ value="http://dc.zah.uni-heidelberg.de/flashheros/q/sdl/dlget"/>
  <PARAM name='contentType' datatype='char' arraysize='*'
    ↪ value='application/fits' />
  <GROUP name="inputParams">
    <PARAM arraysize="*" datatype="char" name="ID" ref="xjc7ra"
      ucd="meta.id;meta.main" value="">
      <DESCRIPTION>The publisher DID of the dataset of interest
      </DESCRIPTION>
    </PARAM>

    <PARAM arraysize="*" datatype="char" name="FLUXCALIB"
      ucd="phot.calib" utype="ssa:Char.FluxAxis.Calibration"
      value="">
      <DESCRIPTION>Recalibrate the spectrum. Right now, the only
      recalibration supported is max(flux)=1 ('RELATIVE').
      </DESCRIPTION>
      <VALUES>
        <OPTION name="RELATIVE" value="RELATIVE"/>
        <OPTION name="UNCALIBRATED" value="UNCALIBRATED"/>
      </VALUES>
    </PARAM>

    <PARAM ID="axi5fg" datatype="float" name="LAMBDA_MIN"
      ucd="par.min;em.wl" unit="m" value="">
      <DESCRIPTION>Spectral cutout interval, lower limit
      </DESCRIPTION>
      <VALUES>
        <MIN value="3.4211e-07"/>

```

```

        <MAX value="5.5927e-07"/>
    </VALUES>
</PARAM>

<PARAM ID="k4dfpe" datatype="float" name="LAMBDA_MAX"
    ucd="par.max;em.wl" unit="m" value="">
    <DESCRIPTION>Spectral cutout interval, upper limit
    </DESCRIPTION>
    <VALUES>
        <MIN value="3.4211e-07"/>
        <MAX value="5.5927e-07"/>
    </VALUES>
</PARAM>

<PARAM arraysize="*" datatype="char" name="FORMAT"
    ucd="meta.code.mime" utype="ssa:Access.Format"
    value="">
    <DESCRIPTION>MIME type of the output format</DESCRIPTION>
    <VALUES>
        <OPTION name="FITS binary table"
            value="application/fits"/>
        <OPTION name="Original format" value="image/fits"/>
        <OPTION name="Comma separated values" value="text/csv"/>
        <OPTION name="VOTable, tabledata encoding"
            ↪ value="application/x-votable+xml;serialization=tabledata"/>
        <OPTION name="VOTable, binary encoding"
            value="application/x-votable+xml"/>
        <OPTION name="Tab separated values"
            value="text/tab-separated-values"/>
    </VALUES>
</PARAM>
</GROUP>
</RESOURCE>

```

The custom service described above supports 5 input parameters: ID, FLUXCALIB, LAMBDA\_MIN, LAMBDA\_MAX, and FORMAT.

The PARAM describing the ID parameter has a ref attribute; the value is the XML ID of a FIELD element in a results table in the same document (the value *xjc7ra* is arbitrary; it is an opaque string that matches an ID value elsewhere in the document). The specified column contains values for the ID parameter. The client (user) will pick rows (presumably spectra) from the results table and then can invoke the service via the ID parameter and value from that row.

The FLUXCALIB parameter allows the client to specify one of two values: UNCALIBRATED or RELATIVE (listed as OPTIONS along with a description of the meaning). The UCD (Derriere and Preite Martinez et al., 2005) value of phot.calib conveys the basic meaning of this parameter (it is

related to photometric or flux calibration).

The LAMBDA\_MIN and LAMBDA\_MAX parameters allow the user to specify a spectral interval to extract from the spectrum. The PARAM(s) specify that the values are wavelengths: ucd="par.min;em.wl" and ucd="par.max;em.wl" say they are minimum (par.min) and maximum (par.max) wavelength (em.wl) values. The VALUES child elements convey a range of valid wavelength values from which a subset could be extracted.

The FORMAT parameter allows the client to select from a list of output formats for the extracted spectrum. Here, the name of the PARAM is suitable to display (e.g. in a user interface) while the value would be used to call the service.

## 4.6 Example: Self-Describing Service

A service may include a service descriptor that describes itself with its normal output. This usage is comparable to prototype work on S3 (see Rodrigo and Cerviño et al. (2008)) and (+ *moved text*):1 will make it easy for clients to obtain a description of both standard and custom features (eg PARAMETER Options or specific endpoints for the resource).

When combined with calling a service with no input parameters (e.g. as allowed in 2.1.1) it gives ranges or options of all accepted parameters. When combined with calling a service with some of the required parameters only (eg an ID) the response may give defaults specific to the resource identified or selected by these parameters.

*(suppressed text)*

*(suppressed example)*



```

<RESOURCE type='meta' utype='ad hoc:service' ID='PwL'
  name='Power Law fitting'>
  <DESCRIPTION>Apply a power law model on a XMM-Newton
  EPIC spectrum </DESCRIPTION>

  <PARAM name="accessURL" datatype="char" arraysize="*"
  value="http://obs-he-lm:8888/3XMM/fitmodelonspectrum&
  model=powlaw" />
  <GROUP name='inputParams'>
    <PARAM name='oid' datatype='char' arraysize='*'
    value='1160803203386703876' unit ='none' >
      <DESCRIPTION>Spectrum internal ID in the database
    </DESCRIPTION>
    </PARAM>
    <PARAM name='binSize' ucd='spect.binSize'
  datatype='int' unit='none' value='10' >
      <DESCRIPTION>Number of counts per
  bin</DESCRIPTION>
      <VALUES>
        <OPTION value='1' />
        <OPTION value='5' />
        <OPTION value='10' />
        <OPTION value='20' />
        <OPTION value='50' />
      </VALUES>
    </PARAM>
    <PARAM name='nh' ucd='phys.abund.X' datatype='float'
  unit='1e22cm-2' value='0.01' >
      <DESCRIPTION>Galactical NH</DESCRIPTION>
      <VALUES>
        <MIN value='0' />
        <MAX value='1' />
      </VALUES>
    </PARAM>
    <PARAM name='alpha' ucd='meta.code;spect.index'
  datatype='float' unit='none' value='1.7' >
      <DESCRIPTION>Photon index of power
  law</DESCRIPTION>
      <VALUES>
        <MIN value='1' />
        <MAX value='9' />
      </VALUES>
    </PARAM>
  </GROUP>
</RESOURCE>

```

In the above example the oid parameter is repeating the value given in the call of the service. The OPTIONS and ranges given for the three other parameters are restricted and adapted to the spectrum so identified. If the service were called without any parameter the oid would have been free and the ranges and options of the three other parameters would have been valid for the whole service

*(suppressed text : we give .... changes)*

## 5 New “datalink” content-type for the LINK element in VOTable

Beside the cases where {links}URLs are retrieved from standard DAL services responses or service descriptors, if a FIELD contains an URL retrieving a {links}response at each row it is easy to tell this to the client using the content-type attribute of an included LINK element, as long as the content-type “application/x-votable+xml;content=datalink” is provided.i (see Appendix)

```
<FIELD name="bla" datatype="char" arraysize="*"
[utype="Access.reference" ucd="meta.url" ]>
<LINK
content-type="application/x-votable+xml;content=datalink"
title="DataLink" />
<DESCRIPTION> bla bla bla </DESCRIPTION>
</FIELD>
```

## Appendix A Recognizing a {links}RESOURCE URL in various context

This section is an implementation note for {links}RESOURCE recognition and service descriptors After reviewing the two methods by which DataLink documents can be associated to resource entries in these responses we show the limitations of these methods in a more generic context and propose a new solution.

### A.1 DataLink and SIAP-2.0 or ObsTAP services

SIAP-2.0 (Dowler and Bonnarel et al., 2015) services are queryable by setting constraints on the four "axes" of the data (2D-space, spectral, time and polarization) and their properties. Other archive metadata or data details are also queryable (target, collection, facilities, etc...). ObsTAP services are

TAP services delivering the `ivoa.Observe` table queryable via ADQL. The successful response is a `VOTable` containing a mandatory result `TABLE` and optional service descriptor resource(s).

### **A.1.1 DataLink discovery via format and reference columns**

The result table describes the dataset characterization on the different axes and give additional operative features. The various columns are mapped from the `ObsCore-1.1` data model (reference). The `accessReference` and `AccessFormat` fields give different solutions to reach the dataset for download or access. This can be done in two ways:

- The data discovery step has yielded a result with the `AccessFormat` value (media type) : `application/x-votable+xml;content=datalink` To the client, this indicates that what is given in the access reference (e.g., the `access_url` column in `ObsTAP` or `SIAP-2.0`) is a datalink document (see below for `DataLink` functionalities).
- In case the media type is different, the access reference is a direct link to the dataset download.

### **A.1.2 SIAP-2.0, ObsTAP and service descriptors**

To enable additional `Datalink` functionalities, `SIAP-2.0` or `ObsTAP` services can then add a service descriptor in the query response that indicates the availability of a `Datalink` service accompanying the `DAL` service. It references one (or more) field(s) from the `DAL` response.

This is explained in detail in section 4.2 above. The net result is that `DataLink`-enabled clients can find ancillary data and use associated services for data access or processing by virtue of being able to retrieve `Datalink` documents.

The service descriptor embedded in an `ObsTAP` query response is allowed because `ObsTAP` is a peculiar `TAP` service and `TAP` allows the `TAP` response to contain additional resources in addition to the "results" one (See section 2.9 of `TAP-1.0` specification). Hence the inclusion of a service descriptor service is valid.

## **A.2 DataLink in the context of other dataset discovery methods**

Datasets can also be discovered in various other ways. Dataset "discovery" is defined as the discovery of the `ivoa.publisher_id` of a dataset in standard services or by the ad hoc discovery of "obsids" in combination with the a priori knowledge of a `DataLink` service "aware" of these obsids.

Beside the most standard and modern discovery path via ObsTAP/SIAP-2.0, the following alternative scenarii are possible:

- The discovery can be done via an SIAP-1.0 or an SSA service. A DataLink service descriptor should have been added to the Service query response to allow further guiding to additional resources
- The discovery can be done via a paper in a journal implementing publication of datasets associated with articles. The editor should describe or implement a DataLink service (eg via a service descriptor valid for the journal) for further usage and access to download, DataAcces, metadata resources.
- Logs of observations are dataset descriptions belonging to the catalog category. They can be exposed in the VO using ConeSearch or TAP. They are in general not consistent with the ObsCore model. However they allow some kind of "dataset discovery". A Service descriptor can be added to the VOTable containing the log file.
- HiPS is an IVOA standard for a global and hierarchical allsky access to pixel and catalogue data (reference HIPS). HiPS cells contain ids for original dataset which have been processed to build the Healpix maps. HiPS metadata file allows to generate a specific VOTable record for each cell of the HiPS collection including a "progenitor" url. This can be a new way to discover datasets and start a DataLink session to find out additional associated resources.

### A.3 DataLink outside Data discovery context

DataLink 1.0 provides description of resources to be attached to a dataset. In the current specification it is actually indifferent to DataLink output what is the real content of the VOTable entry it is attached to. That's why it is tempting to use DataLink mechanisms in other VO contexts than Dataset discovery.

For example it could be useful to attach to a source in a catalog table such resources as

- data records for the same object contained in another catalog/database
- a service interface to an external database prepared to query around the object position.
- Datasets associated to the source (such as images, spectra, etc..)

In addition we could attach links to each measurement in a measurement list for an individual source formatted in VOTable. Light curves or radial

velocity TimeSeries are good examples of such measurement lists. These links can attach:

- original dataset from which the measurement is extracted
- metadata describing the extraction method
- calibration information

We could also attach links to entities in an IVOA provenance metadata service response.

#### **A.4 Various recognition solutions for use cases introduced in section A.2 and A.3**

In such contexts, the links resource URL attached to a record may be given in various ways:

- It may be deduced from an "identifier" given by one of the main table FIELD. In that case it is easy to attach a service descriptor to the main VOTable containing the measurements.
- It may be given directly in one of the FIELD of the table (let's say the name of this FIELD is "bla"). In that case we may still have two different situations.
  - If the FIELD "bla" contains URL driving different contents from row to row, it is a good idea to use an "Obscore"-like solution, consisting in adding the utype "Access.reference" to FIELD bla, and to add a "foo" column with utype "Access.format" to tell the client what is the nature of the retrieved resource ({links} resource or whatever other media type).
  - If the FIELD "bla" contains always an URL to the {links} resource (or whatever media type) without any change from row to row, it is recommended to use a LINK element inside the FIELD "bla" to qualify the url as described in section 5.

## **Appendix B Changes**

This is the initial version of this document.

## B.1 WD-DataLink-1.1-20190721

- Introduction modified to extend the scope of the specifications
- Added use cases consistently with changes in introduction
- Added a standardID INFO section in {links}response
- Modified access URL field to include the possibility of fragments to point to parts of retrievable resources
- Insistance on the need to fill the description field in the {links}response
- Miscelaneous changes for SERVICE descriptor
- New entry for service DESCRIPTOR for restful services
- Complete rewriting subsection for self-described services
- New section to describe new LINK contentType value for tagging FIELD containig {links}URLs
- Added Appendix for implementation guide lines

## B.2 PR-DataLink-1.0-20150413

- Restricted the {links} resource path so that it must be a sibling of the VOSI resources in order to allow discovery of VOSI resources from a {links} URL.
- Changed ID parameter to allow caller to invoke service with no ID values and get an empty result table; this is actually easier to implement than a special error case. Added reference to previous work on S3 and an example section where an empty links response has a self-describing service descriptor and an empty result.
- Fixed URL to DALI document in the references section.
- Fixed namespace prefix in example capabilities document to use recommended value.

### B.3 PR-DataLink-1.0-20140930

- Re-organised introduction to introduce the links capability and distinguish it from the service descriptor more clearly. Explicitly noted that service descriptors do not describe the output of a service.
- Fixed various small typos mentioned on the RFC page.
- Clarified the use of the DataLink vocabulary in the semantics column of the links table.
- Changed the links table output constraints to allow only one of: `access_url`, `service_def`, or `error_message`. This removes the possible inconsistency of `access_url` in the table being different from `accessURL` in a service descriptor referenced by use of `service_def` and reduces service use by clients to a single supported approach.
- Added specific `datatype="long"` to the `content_length` field in the links table.
- Moved VOSpace-2.0 service descriptor to be a separate example and made it explicit that all the necessary details to invoke such a RESTful service is not supported in this version of the specification; clients must recognise the `standardID` to use RESTful web services.

### B.4 PR-DataLink-20140530

- Changed document status to proposed recommendation.
- Removed `REQUEST` parameter
- Added custom service example.
- Removed standard authentication and authorization error messages since these are difficult to implement consistently in different web service platforms. Changed the error message strings to use the word `Fault` (following GWS-WG usage, e.g. VOSpace-2.0) since `Error` has specific meaning in some platforms.

### B.5 WD-DataLink-20140505

- Changed the `standardID` for the `{links}` resource to include version as will be described in the `StandardsRegExt` record.
- Changed service descriptor resource to use `type="meta" utype="adhoc:service"` so `VOTable` documents pass schema validation and this resource type can still be easily found.

- Improved the VOSI-capabilities example so it describes all parameters of the example DataLink service.
- Removed unnecessary HTTP header advice and clarified the strict DataLink mimetype usage.
- Removed mention of DALI-examples since it is an optional feature for all services.
- Changed name of the input parameters group element in a service descriptor to inputParams.
- Fixed reference to DALI document.
- Added SIA-1.0 resource descriptor example.
- Tried to clarify the relationship of the two aspects of DataLink in the introduction.
- Specifically allow `access_url` in the list of links to be different from `accessURL` in the service descriptor, with VOSpace example.

## B.6 WD-DataLink-20140212

- Clarified that one can implement a standalone DataLink service or include `{links}` resources in other services.
- Re-ordered sections 2–5 so all the sections describing the `{links}` capability are together.
- Changed from `GROUP` with `PARAM` and `FIELDref` siblings to `PARAM` with `ref` attribute when defining a parameter-column-with-values in section 4.1.
- Clarified the introduction so it is clear we intend to support linking of any services via `RESOURCE(s)` in any responses.
- Changed the output of `{links}` resource to clearly differentiate between links with usable `accessURL` and links where the `accessURL` is a service that requires more parameters. Changed the naming style for fields in the list of links to use lower case with underscore separator so that direct potential implementations don't run into case issues.

## References

- Berners-Lee, T., Fielding, R. and Masinter, L. (2005), 'Uniform Resource Identifier (URI): Generic syntax', RFC 3986.  
<http://www.ietf.org/rfc/rfc3986.txt>



- Bonnarel, F. (2019), 'Ivoa provenance table access protocol (provtap)', IVOA internal working draft.  
<https://wiki.ivoa.net/internal/IVOA/ObservationProvenanceDataModel/ProvTAP.pdf>
- Bradner, S. (1997), 'Key words for use in RFCs to indicate requirement levels', RFC 2119.  
<http://www.ietf.org/rfc/rfc2119.txt>
- Derriere, S., Preite Martinez, A., Williams, R., Gray, N., Mann, R., McDowell, J., Mc Glynn, T., Ochsenbein, F., Osuna, P. and Rixon, G. (2005), 'An IVOA Standard for Unified Content Descriptors Version 1.10', IVOA Recommendation 19 August 2005, arXiv:1110.0525.  
<http://dx.doi.org/10.5479/ADS/bib/2005ivoa.spec.0819D>
- Dowler, P., Bonnarel, F. and Tody, D. (2015), 'IVOA Simple Image Access Version 2.0', IVOA Recommendation 23 December 2015.  
<http://dx.doi.org/10.5479/ADS/bib/2015ivoa.spec.1223D>
- Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2013), 'Data Access Layer Interface Version 1.0', IVOA Recommendation 29 November 2013, arXiv:1402.4750.  
<http://dx.doi.org/10.5479/ADS/bib/2013ivoa.spec.1129D>
- Dowler, P., Rixon, G. and Tody, D. (2010), 'Table Access Protocol Version 1.0', IVOA Recommendation 27 March 2010, arXiv:1110.0497.  
<http://dx.doi.org/10.5479/ADS/bib/2010ivoa.spec.0327D>
- Freed, N. and Borenstein, N. (1996), 'MIME part one: Format of internet message bodies', RFC 2045.  
<http://www.ietf.org/rfc/rfc2045.txt>
- Graham, M., Rixon, G. and Grid and Web Services Working Group (2011), 'IVOA Support Interfaces Version 1.0', IVOA Recommendation 31 May 2011, arXiv:1110.5825.  
<http://dx.doi.org/10.5479/ADS/bib/2011ivoa.spec.0531G>
- Gregorio, J., Fielding, R., Hadley, M., Nottingham, M. and Orchard, D. (2012), 'Uri template', RFC 6570.  
<https://tools.ietf.org/html/rfc6570>
- Harrison, P., Burke, D., Plante, R., Rixon, G., Morris, D. and IVOA Registry Working Group (2012), 'StandardsRegExt: a VOResource Schema Extension for Describing IVOA Standards Version 1.0', IVOA Recommendation 08 May 2012, arXiv:1402.4745.  
<http://dx.doi.org/10.5479/ADS/bib/2012ivoa.spec.0508H>

- Harrison, P., Tody, D. and Plante, R. (2009), ‘Simple Image Access Specification Version 1.0’, IVOA Recommendation 11 November 2009, arXiv:1110.0499.  
<http://dx.doi.org/10.5479/ADS/bib/2009ivoa.spec.1111H>
- Louys, M., Tody, D., Dowler, P., Durand, D., Michel, L., Bonnarel, F., Micol, A. and IVOA DataModel Working Group (2017), ‘Observation Data Model Core Components, its Implementation in the Table Access Protocol Version 1.1’, IVOA Recommendation 09 May 2017.  
<http://dx.doi.org/10.5479/ADS/bib/2017ivoa.spec.0509L>
- Ochsenbein, F., Taylor, M., Williams, R., Davenhall, C., Demleitner, M., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A. and Wicenec, A. (2013), ‘VOTable Format Definition Version 1.3’, IVOA Recommendation 20 September 2013.  
<http://dx.doi.org/10.5479/ADS/bib/2013ivoa.spec.09200>
- Plante, R., Stébé, A., Benson, K., Dowler, P., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T. and Rixon, G. (2010), ‘VODataService: a VOResource Schema Extension for Describing Collections, Services Version 1.1’, IVOA Recommendation 02 December 2010, arXiv:1110.0516.  
<http://dx.doi.org/10.5479/ADS/bib/2010ivoa.spec.1202P>
- Plante, R., Williams, R., Hanisch, R. and Szalay, A. (2008), ‘Simple Cone Search Version 1.03’, IVOA Recommendation 22 February 2008, arXiv:1110.0498.  
<https://ui.adsabs.harvard.edu/abs/2008ivoa.specQ0222P>
- Richardson, L. and Ruby, S. (2007), *RESTful Web Services*, O’Reilly.
- Riebe, K. (2018), ‘Ivoa provenance simple access protocol (provsap)’, IVOA internal working draft.  
<http://volute.g-vo.org/svn/trunk/projects/dm/provenance/ProvSAP/>
- Rodrigo, C., Cerviño, M., Solano, E. and Manzato, P. (2008), ‘S3: Proposal for a simple protocol to handle theoretical data (microsimulations)’, IVOA Note.  
<http://www.ivoa.net/documents/latest/S3TheoreticalData.html>
- Servillat, M., Riebe, K., Boisson, C., Bonnarel, F., Galkin, A., Louys, M., Nullmeier, M., Sanguillon, M. and Streicher, O. (2019), ‘Ivoa provenance data model’, IVOA Proposed Recommendation.  
<http://www.ivoa.net/documents/ProvenanceDM/index.html>
- Tody, D., Dolensky, M., McDowell, J., Bonnarel, F., Budavari, T., Busko, I., Micol, A., Osuna, P., Salgado, J., Skoda, P., Thompson, R., Valdes,

F. and Data Access Layer Working Group (2012), 'Simple Spectral Access Protocol Version 1.1', IVOA Recommendation 10 February 2012, arXiv:1203.5725.  
<http://dx.doi.org/10.5479/ADS/bib/2012ivoa.spec.0210T>