International
Virtual
Observatory
Alliance

IVOA DataLink

Version 1.1

IVOA Proposed Recommendation 2023-09-11

Working Group
DAL
This version
https://www.ivoa.net/documents/DataLink/20230911
Latest version
https://www.ivoa.net/documents/DataLink
Previous versions
WD-20211115
DataLink-1.0
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 Proposed Recommendation made available for public review. It is appropriate to reference this document only as a recommended standard that is under review and which may be changed before it is accepted as a full Recommendation.

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

Contents

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**", "**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 data items discovered via one service to related data products and web services.

The *links* web service capability is a web service capability for drilling down from a discovered data 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 clients to find ancillary resources like progenitors, derived data products, or alternate representations of the data, and services that can act upon the data (usually without having to download the entire dataset). The expected usage 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 data 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.

We expect that the *service descriptor resource* mechanism will be the primary way that clients will find and use the *links* capability from data discovery 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.

PDF fallback: A conversion from SVG to PDF failed. This is probably because inkscape is not installed. While SVG is not supported by the major TeX engines, it is recommended to commit built PDFs to the VCS.

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. (2017)), including the Virtual Observatory Support Interfaces resources (VOSI, Graham and Rixon et al. (2017)). DataLink services use VOTable (Ochsenbein and Taylor et al., 2019) as the default output format both for successful output and to return error documents.

DataLink specifies a standardID for itself which, as defined in VOResource (Plante and Demleitner et al., 2018), is used to identify compliant service capabilities in Registry and VOSI metadata. 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 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 be custom services with peculiar functionalities or IVOA standard services. The IVOA access data service standard is SODA (Bonnarel and Dowler et al., 2017). SODA services should be described in the same way as custom access data services.

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 standards such as ConeSearch (Plante and Williams et al., 2008) or TAP. 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 easy retrieval of 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 the data. This information can be stored and retrieved in several ways including in 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 the more complete data they represent using the {links} endpoint. Reversely full provenance records can be linked to standard discovery service rows using the same endpoint.

2 The {links} endpoint

Most commonly, DataLink link lists are retrieved from {links} endpoints. These are DALI-sync endpoints with implementor-defined names. As specified by DALI-sync, the parameters for a request are submitted using an HTTP GET (query string) or POST action. Any service may offer zero or more datalink endpoints.

2.1 Parameters on {links} endpoints

On {links} endpoints, the ID and RESPONSEFORMAT parameters as defined below are mandatory.

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. Submitting ID values in batches may be more efficient if the client is planning to submit many such values; clients can control the size of the output by limiting the number of ID values they submit in each request.

Services **may** place a limit on the number of ID values they will process in one request. 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 to denote that the result is truncated as described in DALI. The service **must not** truncate the output within the set of rows (links) for a single ID value.

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 **??**).

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 Registering {links} endpoints

Since normal datalink operations do not involve the Registry, this specification poses no requirements to register {links} endpoints. Datalink clients also generally have no reason to inspect VOSI capabilities endpoints, and hence there are no requirements on mentioning {links} endpoints in any VOSI capability documents.

Operators still wishing to declare {links} endpoints can do this by giving a capability with a standardID of

```
ivo://ivoa.net/std/DataLink#links-1.1
```

This specification does not constrain the capability type used in such declarations. The access URL of the {links} endpoint **must** be given in a *vs:ParamHTTP*-typed interface element.

Hence, a single datalink capability could be declared as follows within either a VOResource record or a VOSI capabilities element:

```
<capability standardID="ivo://ivoa.net/std/DataLink#links-1.1"
    xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.1">
    <interface xsi:type="vs:ParamHTTP" role="std">
      <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>
      <param std="true" use="optional">
        <name>RESPONSEFORMAT</name>
        <description>Return the links in this tabular format (defaults
          to VOTable).</description>
      </param>
    </interface>
</capability>
```

2.3 VOSI

Since DataLink services are not usually registered, the VOSI-capabilities endpoint is not required; the VOSI-availability endpoint is **optional**.

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 preferred 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).

3.1.1 DataLink recognition outside the context of ObsCore responses

In order to provide access URL to {links} endpoints outside the ObsCore-style service responses context there are several options.

- A DataLink service descriptor (4) could be added to the main table as long as the content of one of the table FIELDs may be used as the ID parameter in the {links} endpoint. An exemple is given in subsection 4.5

- In case the {links} endpoint URL is given in the main table for each row, a LINK element SHOULD be added to the FIELD containing this URL. The content-type of the LINK will be set to :

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

  In the example below the content of the "Datalink" FIELD is used as an URL to the {links} endpoint for each row.

  ```
  <FIELD name="DataLink" datatype="char" arraysize="*" ucd="meta.ref.url"  >
     <LINK content-type="application/x-votable+xml;content=datalink"  />
   </FIELD>
  ```

- In case the main table contains an URL not systematically pointing to {links} endpoint, but may also point to responses with other content types (e.g. for single file download), the ObsCore utype "Access.reference" SHOULD be added to the FIELD and an additional FIELD with ObsCore utype "Access.format" SHOULD contain the "application/x-votable+xml;content=datalink" value when appropriate. A ref to the ID of the FIELD containing the URL SHOULD be added to avoid ambiguities.

  In the example below, the Image-Link contains the URL pointing to a {links} response or the dataset istself depending on the value of the Image-Format FIELD.

```
<FIELD ID="ILink" name="Image-Link" datatype="char" arraysize="*"
ucd="meta.ref.url" utype="AccessReference" />
<FIELD name="Image-Format" datatype="char" arraysize="*"
 ucd="meta.code.mime" utype="AccessFormat" ref="ILink" />
```

3.2 List of Links

The list of links that is returned by the {links} endpoint can be represented as a table with the columns listed in Table 1.

name	**description**	**field required**	**value required**	**UCD**
ID	Input identifier	yes	yes	meta.id;meta.main
access_url	link to data or service	yes	one only	meta.ref.url
service_def	reference to a service descriptor resource	yes		meta.ref
error_message	error if an access_url cannot be created	yes		meta.code.error
description	human-readable text describing this link	yes	no	meta.note
semantics	Term from a controlled vocabulary describing the link	yes	yes	meta.code
content_type	mime-type of the content the link returns	yes	no	meta.code.mime
content_length	size of the download the link returns	yes	no	phys.size;meta.file
content_qualifier	nature of the content the link returns	no	no	
local_semantics	An identifier that allows clients to associate rows from different datalink documents on the same service with each other.	no	no	meta.id.assoc
link_auth	use of the link requires authentication	no	no	meta.code
link_authorized	caller is authorized to use the link	no	no	meta.code

Table 1: Fields for Links Output

Fields **must** be present and values 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

To facilitate consumption of large datalink results in streaming mode, all links for a single ID value **must** be served in consecutive rows in the output.

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).

Unless specified otherwise below, all fields are text values (datatype= "char" in the VOTable FIELD).

3.2.1 ID

The ID column contains the input identifier value.

3.2.2 access_url

The access_url column contains a URL to download a single resource. This URL can be a static link or a link to a dynamic resource (e.g. preview generation).

Access URLs may have fragment parts, which could, for instance, refer to id-ed elements within XML documents or extensions within FITS files. As in URIs in general, the interpretation of a fragment identifier depends on the media type. Apart from that no other client handling is expected.

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 4.1. For example, if the response document includes this resource to describe a service:

```
<RESOURCE type="meta" utype="adhoc: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 access_url or service_def can be generated for an input identifier. If an error_message is included in the output, the ID and semantics values **must** be provided as usual; in particular, the value in the semantics column should reflect the semantics of the link that could not be produced. From version 1.1 of this standard, services **may** provide values in other fields or leave them null (as was required in 1.0).

For example, if an ID value is unrecognized by the service, it would normally provide the minimum output: the input value for the ID, `#this` for semantics, and an error message. If a service did recognise the input ID and would normally create a download link, but generating the access_url failed, the service could include the usual content_type, content_length, and description along with the ID, semantics, and error_message.

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 URI for a concept that describes the meaning of the linked item relative to what ID references. The semantics column is intended to be machine-readable and to assist automated link selection, presentation, and usage.

The value is always interpreted as a URI; relative URIs (Berners-Lee and Fielding et al., 2005) are completed using the base URI of the core DataLink vocabulary, `http://www.ivoa.net/rdf/datalink/core`. Terms from this vocabulary **must** always be written as relative URIs. This means that for concepts from the core vocabulary, the value in the semantics column always starts with a hash.

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 of the preview, and the semantics column will be `#preview`.

The core DataLink vocabulary defines a special term for the concept of *this*; this term is used to describe links available for the retrieval of the file(s) making up what ID references.

For concepts outside the core DataLink vocabulary, the full concept URI **must** be given. It **should** resolve to a human-readable document describing what the concept means and what clients are expected to do with links annotated with it.

As per Vocabularies in the VO 2 (Demleitner and Gray et al., 2021), at `http://www.ivoa.net/rdf/datalink/core` the datalink core vocabulary can be retrieved in various formats including HTML (in a way that the concept URI is usable in a web browser), various RDF serialisations, and the VO-specific Desise optimised for simple machine consumption; this should be used by clients to present the user with labels (and perhaps definitions) rather than the URI parts given in the semantics column.

In RDF terms, the concepts in datalink core are properties. A datalink row can be interpreted as an RDF triple

$$(\langle access\_url\rangle, \textit{is-a-}\langle semantics\rangle\textit{-for}, \langle \mathrm{ID}\rangle).$$

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.

3.2.9 content_qualifier

The content_qualifier column is **optional**. If it is present, it tells the client the nature of the thing or service they will receive or access if they use the link.

If the access_url references a data product, the content_qualifier field **should** define its product type. In that case, the considerations for the semantics column (Sect. 3.2.6) apply, except that the basic vocabulary is `http://www.ivoa.net/rdf/product-type`, and the interpretation as an RDF triple would be

$$(\langle access\_url\rangle, \textit{is-a}, \langle content\_qualifier\rangle)$$

For rows not linking to data products, content_qualifier's interpretation will be different, and the default vocabulary will be inappropriate. Full

concept URIs will have to be used in this case, and their translations to RDF triples is not covered by this version of DataLink.

3.2.10 local_semantics

The local_semantics column allows for identification of corresponding rows for different IDs in the same DataLink service where the combination of semantics, content_type and content_qualifier is not sufficient to identify them. It contains a service specific vocabulary. It aids clients in presenting to the user the same sort of link as they go from one dataset to another within a service. For instance, suppose a service serves both continuum and line cubes. Using local_semantics, users can configure their clients such that, as they change to a new data set, they always see the line cube even when the semantics, content_qualifier and content_type columns agree for both types of data. The vocabulary can be a simple list of terms defined for the service (eg : local_semantics="line-cube") or can be described in an ad hoc external resource accessible via an URI (eg : local_semantics="http://our-service-adhoc-vocab/terms#continuum-cube").

3.2.11 link_auth

The link_auth column tells the client whether or not authentication is required to use the link. Valid values are:

`false` : the link allows anonymous access only

`optional` : the link supports both anonymous and authenticated access

`true` : authentication is required

This field **may** be null (blank) if the value is unknown.

3.2.12 link_authorized

The link_authorized column tells the client whether the currently authenticated identity is authorized to use the link. For VOTable, the FIELD **must** be datatype="boolean". This is generally a prediction to save clients from trying to use a link and getting a permission denied response. Valid values are:

`false` : current user not authorized

`true` : current user is authorized

If the value is `false` and the caller tries to use the link anyway, it may be challenged for credentials (e.g. HTTP 401 response with WWW-Authenticate headers) or denied (e.g. HTTP 403 “permission denied”).

If the value is `true`, the caller should proceed with the same authentication and should expect to succeed.

This field **may** be null (blank) if the value is unknown.

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 one of the values allowed by the VOTable specification, which at the time of this writing includes "application/x-votable+xml" and "text/xml". The former value is preferred and SHOULD be augmented 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 and UCD attributes for FIELD elements in the VOTable (and the units in one case) are specified above (see 3.2).

The DALI specification states that VOTable output main "results" RESOURCE should include an INFO element with name="standardID" and the standardID string as a value.

```
<RESOURCE type="results">
  ...
  <INFO name="standardID" value="ivo://ivoa.net/std/DataLink#links-1.1"/>
  ...
  <TABLE>
  ...
  </TABLE>
  ...
</RESOURCE>
```

From version 1.1 of this standard, the {links} response main "results" RESOURCE **must** include this INFO element so that a table of links is easily identified by users and applications when initially received from the service and if saved for later use.

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). 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.

Here we describe how to construct a resource that describes a service and add it to a VOTable document. This "service descriptor" mechanism can be used in any VOTable document, such as a data discovery response from a TAP query or one of the simple DAL query protocols or the {links} endpoint described above. The linked services can be any HTTP service, including but not limited to the {links} endpoint described above, other IVOA services (e.g. SODA), custom services, or other kinds of internet resources like web pages (e.g. interactive applications, DOI landing pages, or documentation).

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 document would also contain one or more resources with attribute type="meta" and utype="adhoc:service" (or utype="adhoc:this" in case of a self-describing service — see **??**). 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 short name attribute, and a more verbose DESCRIPTION subelement, MAY be added to the service descriptor RESOURCE to provide the user with information about the service's purpose or semantics. This SHOULD be done if the semantics are not obvious, and especially in the case of multiple sibling service descriptors, or non-standard services.

In cases where a response document contains several "service descriptor" RESOURCEs and several "results" RESOURCEs, these RESOURCEs MAY be nested in order to better display correct association.

4.2 Descriptive PARAMs

A service resource contains PARAM elements to describe the service. The standard PARAM elements for a *service* resource are described in Table 3.

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
exampleURL	example invocation of the service	no

Table 3: Parameters Describing the Service

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.

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.

In case the contentType is "text/html", the client SHOULD send the result of the service query to a web browser. This is appropriate for both HTML documents and web interactive interfaces.

A service descriptor **may** contain multiple exampleURL PARAMs. In exampleURL PARAMs, operators can give valid service calls as GET-able URLs in the PARAMs' value attribute. They are intended as an aid for debugging, in particular to aid users and developers in making sure a service is still operating as expected. The PARAM's description **should** give an indication of what the call will result in. End-user clients might indicate exampleURLs to the user after unexpected service failures.

4.3 Input PARAMs

A service descriptor **must** contain a GROUP element with name="inputParams" to describe user-specified input parameters of the service. There are three types of input params: params with a fixed value, params where the values come from the "results", and params where the value is variable and chosen/specified by the user.

For params with a fixed value (e.g. fly="true"), the client **must** treat it as a required parameter and include it in the service invocation; this allows a service implementor to include constant params explicitly (and describe them via a DESCRIPTION element) rather than just include them in the "accessURL" without the possibility to explain them.

For services where the parameter value(s) come from the "results" resource, the value attribute is empty (value="") and the PARAM includes a ref attribute to indicate the FIELD (column) that contains the values. For

example, a TAP query result may contain identifiers that can be used to invoke the links service; the FIELD with the identifiers **must** have an XML ID attribute (e.g. ID="abc") and the input PARAM would include the attribute ref="abc"). When this mechanism is used, the client **must** treat it as a required parameter and the parameter and value **must** be included in the service invocation.

For user-specified input PARAMs the value attribute is empty (value="") and the user supplies the value(s). The PARAM specifies the type of value required via the datatype, arraysize, and xtype attributes; this may be augmented further by the ucd, units and utypes[1] attributes and a child DESCRIPTION element. To allow for expressive, usable user interfaces, operators SHOULD indicate useful ranges of parameters in MIN and MAX children or, for enumerated parameters, indicate the valid values in OPTIONS in case these values cannot be inferred from relevant metadata retrieved before the service descriptor discovery. In general, services may have parameters of this type that are optional or required and this distinction is not currently described; services **should** use a child DESCRIPTION element to document any requirements. Clients should assume that these user-specified parameters are optional, but that specifying some of them may be necessary to have the service do something useful. Services **should** respond with an informative error message if the input is not adequate to perform the operations(s).

4.4 Service self-description

A service may include a service descriptor that describes itself with its normal output. In that case the utype "adhoc:this" indicates the self-describing nature of the service descriptor. This convention makes finding the self-description unambiguous in cases where the output also contains other service descriptors. This usage is comparable to prototype work on S3 (see Rodrigo and Cerviño et al. (2008)) and when combined with calling a service with no input parameters (e.g., as allowed in 2.1.1), and/or with the DALI `MAXREC=0` convention, will make it easy for clients to obtain a description of both standard and custom features.

For backward compatibility with DataLink 1.0 and SIA 2.0, client software conforming to the present recommendation should also treat elements of the form `<RESOURCE type="meta" utype="adhoc:service" name="this"/>` as self-descriptions, equivalent to `<RESOURCE type="meta" utype="adhoc:this" name=""/>`. A conforming client should treat the provision of more than one self-description `<RESOURCE>` element as an error, except that if a service provides exactly one of each of the present (DataLink 1.1 and beyond) and DataLink

[1] An example of utype usage for service parameters is described in section 3.4 of the SODA specification

1.0 styles, the client may silently ignore the DataLink 1.0 style instance.

The output of a {links} endpoint with no input ID would include the self-describing service descriptor and an empty results table.

4.5 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-1.1 data model would use the following:

```
<FIELD name="obs_publisher_did" ID="primaryID"
       utype="obscore:Curation.PublisherDID"
       ucd="meta.ref.ivoid"
       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 is described by the following resource:

```
<RESOURCE type="meta" utype="adhoc:service" name="RawAndCatalogDataLinks">
  <DESCRIPTION>
    This datalink service gives access to the raw data for the
    discovered datasets as well as to catalogues of extracted sources
  </DESCRIPTION>
  <PARAM name="standardID" datatype="char" arraysize="*"
         value="ivo://ivoa.net/std/DataLink#links-1.1" />
  <PARAM name="accessURL" datatype="char" arraysize="*"
         value="http://example.com/mylinks" />
  <PARAM name="contentType" datatype="char" arraysize="*"
         value="application/x-votable+xml;content=datalink" />
  <PARAM name="exampleURL" datatype="char" arraysize="*"
         value="http://example.com/mylinks?ID=NGC%206946" />
  <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>
```

The exampleURL value in the example above provides an example of a URL that use of this service descriptor could produce; it **should** resolve to produce an actual result.

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.

4.6 Example: Service Descriptor for an SIA-1.0 Service

Suppose you have an SIA-1.0 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="adhoc:service"
          name="RadioCubeDiscoveryService">
  <DESCRIPTION>
   This parameter based HTTP service allows discovery of Radio Cubes
   obtained by LOFAR observations processing
  </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 would describe 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.

4.7 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="adhoc:service" ID="vnodes" name="CADC-Store">
  <DESCRIPTION>
   Datasets discovered here are automatically available in
   CADC's VOSpace under the URI produced here
  </DESCRIPTION>
  <PARAM name="resourceIdentifier" datatype="char" arraysize="*"
         value="ivo://example.com/vospace" />
  <PARAM name="standardID" datatype="char" arraysize="*"
         value="ivo://ivoa.net/std/VOSpace/v2.0#nodes" />
  <PARAM name="accessURL" datatype="char" arraysize="*"
         value="http://example.com/vospace/nodes" />
  <GROUP name="inputParams">
    <PARAM name="detail" datatype="char" arraysize="*"
           value="min"/>
    <PARAM name="limit" datatype="int"
           value="1000"/>
    <PARAM name="uri" datatype="char" arraysize="*"
           value=""/>
  </GROUP>
</RESOURCE>
```

```
<RESOURCE type="meta" utype="adhoc:service" ID="vtrans">
  <PARAM name="resourceIdentifier" datatype="char" arraysize="*"
         value="ivo://example.com/vospace" />
  <PARAM name="standardID" datatype="char" arraysize="*"
         value="ivo://ivoa.net/std/VOSpace/v2.0#transfers" />
  <PARAM name="accessURL" datatype="char" arraysize="*"
         value="http://example.com/vospace/transfers" />
</RESOURCE>
```

Since the capability being described is RESTful, the caller must recognise the standardID values and use a VOSpace-aware client to call the service.

4.8 Example: SODA Spectral Cutout with Custom Parameters

The following service descriptor conforms to the requirements of SODA (Bonnarel and Dowler et al., 2017) and offers a cutout service for a spectrum. It also offers further, non-standard parameters for format conversion and basic re-calibration. It gives enough metadata to enable informative user interfaces.

```
<RESOURCE ID="procsvc" name="proc_svc" type="meta"
    utype="adhoc:service">
  <GROUP name="inputParams">
    <PARAM arraysize="*" datatype="char" name="ID"
           ucd="meta.id;meta.main"
           value="ivo://org.gavo.dc/~?feros/data/f08751.fits">
      <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 arraysize="2" datatype="double" name="BAND"
           ucd="em.wl" unit="m" value=""
           xtype="interval">
      <DESCRIPTION>Spectral cutout interval</DESCRIPTION>
      <VALUES>
        <MIN value="3.52631986e-07"/>
        <MAX value="9.21500998e-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="VOTable, binary encoding"
                value="application/x-votable+xml"/>
        <OPTION name="VOTable, tabledata encoding"
                value="application/x-votable+xml;serialization=tabledata"/>
        <OPTION name="Tab separated values" value="text/plain"/>
        <OPTION name="Comma separated values" value="text/csv"/>
        <OPTION name="FITS binary table" value="application/fits"/>
      </VALUES>
    </PARAM>
  </GROUP>
  <PARAM arraysize="*" datatype="char" name="accessURL"
         ucd="meta.ref.url"
         value="http://dc.zah.uni-heidelberg.de/feros/q/sdl/dlget"/>
  <PARAM arraysize="*" datatype="char" name="standardID"
         value="ivo://ivoa.net/std/SODA#sync-1.0"/>
</RESOURCE>
```

The PARAM describing the ID parameter has a non-empty value attribute, meaning that a client will always call the service with the dataset ID of a specific dataset. This is typical for service descriptors in datalink documents.

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 BAND parameter allows the user to specify a spectral interval to extract from the spectrum and follows SODA's regulations. Its VALUES child declares the range of wavelengths in the dataset; services **should** always try to give information on the sensible ranges of input parameters, and clients should strive to make them easily accessible to users, if possible in the users' preferred units. Given that at this point users do not have access to the full dataset, it is otherwise hard for them to guess what could be entered.

Finally, note the standardID PARAM outside of the GROUP of input parameters. It is a promise that the service conforms to SODA's guarantees (e.g., that BAND actually works as specified there). Clients must compare its value case-insensitively (because it is an IVOA identifier) and should for robustness ignore everything after the dot in the fragment identifier when determining whether or not to treat a service as a SODA version 1 service, as the minor version is guaranteed to be operationally insignificant.

4.9 Example: Self-Describing Service

The following service descriptor self-describes an ad hoc (non standard) service to apply a power law model on XMM-Newton spectra. It gives enough metadata to enable informative user interfaces to this service appropriate for the specific spectrum identified by the specified oid.

```
<RESOURCE type="meta" utype="adhoc:this" 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&amp;model=powlaw"/>
  <GROUP name="inputParams">
    <PARAM name="oid" datatype="char" arraysize="*"
           value="1160803203386703876">
      <DESCRIPTION>Spectrum internal ID in the database </DESCRIPTION>
    </PARAM>
    <PARAM name="binSize" ucd="spect.binSize" datatype="int" 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"
           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 we give the self-describing service descriptor a name attribute with the value “adhoc:this” to indicate the self-describing

nature. This convention would make finding the self-description unambiguous in cases where (i) the output also contained other service descriptors and (ii) the caller could not infer which descriptor was the self-describing one from the standardID (because it is optional and not present for custom services and because they might just have a URL). Even trying to match the URL that was used with the accessURL in the descriptors is likely to be unreliable (e.g. if providers use HTTP redirects to make old URLs work when service deployment changes).

5 Changes

5.1 DataLink-1.1

- allow optional columns to contain values when the row (link) has an error_message (see Section 3.2.4)
- added optional local_semantics to identify corresponding rows for different IDs in the same service (see Section 3.2.10)
- relax content-type usage to allow any valid VOTable MIME type (see Section 3.3)
- INFO element with standardID mandatory in {links} response (see Section 3.3.1)
- added optional content_qualifier to describe link target content with terms from the product-type vocabulary (see Section 3.2.9)
- added optional link_auth and link_authorized to signal whether authentication is necessary to use the link (see Sections 3.2.11 and 3.2.12)
- clarified use of multiple ID values and possible OVERFLOW (see Section 2.1.1)
- clarified use of utype for self-describing service descriptors(see Section **??**)
- clarified use of semantics (see Section 3.2.6)
- generalize by adding use cases for links to content other than data files (see Sections 1.2.8 and 1.2.9)
- added using LINK to convey when datalink request URL is in a table column (see Section 3.1.1)
- service descriptors can include a contentType param to describe service output and should include a name and description (see Sections 4.1 and 4.2)

- service descriptors can include exampleURL param(s) with working example and description (see Section 4.2)
- VOSI-availability and VOSI-capabilities endpoints are now optional (see Section 2.3)
- Capability standardID updated to `ivo://ivoa.net/std/DataLink#links-1.1` (see Section 2.2)

5.2 DataLink-1.0

Detailed evolution up to version 1.0 described below.

5.3 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.

5.4 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.

5.5 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.

5.6 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 desciptor 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.

5.7 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., Dowler, P., Demleitner, M., Tody, D. and Dempsey, J. (2017), 'IVOA Server-side Operations for Data Access Version 1.0', IVOA Recommendation 17 May 2017, arXiv:1710.08791.
http://doi.org/10.5479/ADS/bib/2017ivoa.spec.0517B

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

Demleitner, M., Gray, N. and Taylor, M. (2021), 'Vocabularies in the VO Version 2.0', IVOA Recommendation 25 May 2021.
https://ui.adsabs.harvard.edu/abs/2021ivoa.spec.0525D

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://doi.org/10.5479/ADS/bib/2005ivoa.spec.0819D

Dowler, P., Demleitner, M., Taylor, M. and Tody, D. (2017), 'Data Access Layer Interface Version 1.1', IVOA Recommendation 17 May 2017.
http://doi.org/10.5479/ADS/bib/2017ivoa.spec.0517D

Dowler, P., Rixon, G. and Tody, D. (2010), 'Table Access Protocol Version 1.0', IVOA Recommendation 27 March 2010, arXiv:1110.0497.
http://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., Dowler, P., Major, B., Grid and Web Services Working Group (2017), 'IVOA Support Interfaces Version 1.1', IVOA Recommendation 24 May 2017.
http://doi.org/10.5479/ADS/bib/2017ivoa.spec.0524G

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://doi.org/10.5479/ADS/bib/2017ivoa.spec.0509L

Ochsenbein, F., Taylor, M., Donaldson, T., Williams, R., Davenhall, C., Demleitner, M., Durand, D., Fernique, P., Giaretta, D., Hanisch, R., McGlynn, T., Szalay, A. and Wicenec, A. (2019), 'VOTable Format Definition Version 1.4', IVOA Recommendation 21 October 2019.
https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.1021O

Plante, R., Demleitner, M., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T. and Rixon, G. (2018), 'VOResource: an XML Encoding Schema for Resource Metadata Version 1.1', IVOA Recommendation 25 June 2018.
http://doi.org/10.5479/ADS/bib/2018ivoa.spec.0625P

Plante, R., Williams, R., Hanisch, R. and Szalay, A. (2008), 'Simple Cone Search Version 1.03', IVOA Recommendation 22 February 2008, arXiv:1110.0498.
http://doi.org/10.5479/ADS/bib/2008ivoa.specQ0222P

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