International
Virtual
Observatory
Alliance

VODataService: A VOResource Schema Extension for Describing Collections and Services

Version 1.2

IVOA Proposed Recommendation 2021-02-23

Working group
Registry

This version
https://www.ivoa.net/documents/VODataService/20210223

Latest version
https://www.ivoa.net/documents/VODataService

Previous versions
PR-2019-07-15
WD-2018-10-26
REC 1.1

Author(s)
Raymond Plante, Markus Demleitner, Aurélien Stébé, Kevin Benson, Patrick Dowler, Matthew Graham, Gretchen Greene, Paul Harrison, Gerard Lemson, Tony Linde, Guy Rixon

Editor(s)
Markus Demleitner, Ray Plante

Abstract

VODataService refers to an XML encoding standard for a specialized extension of the IVOA Resource Metadata that is useful for describing data collections and the services that access them. It is defined as an extension of the core resource metadata encoding standard known as VOResource (Plante and Demleitner et al., 2018) using XML Schema. The specialized resource types defined by the VODataService schema allow one to describe how the data underlying the resource cover the sky and the frequency and time axes. VODataService also enables detailed descriptions of tables that includes information useful to the discovery of tabular data. It is intended that the VODataService data types will be particularly useful in describing services that support standard IVOA service protocols.

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

Versions 1.0 and 1.1 of this document have been developed with support from the National Science Foundation's Information Technology Research Program under Cooperative Agreement AST0122449 with The Johns Hopkins University, from the UK Particle Physics and Astronomy Research Council (PPARC), from the European Commission's (EC) Sixth Framework Programme via the Optical Infrared Coordination Network (OPTICON), and from EC's Seventh Framework Programme via its eInfrastructure Science Repositories initiative.

Version 1.2 of this document was developed in part with support from the German federal ministry for research and education's e-inf-astro project (BMBF FKZ 05A17VH2).

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.

Syntax Notation Using XML Schema

The eXtensible Markup Language, or XML, is a document syntax for marking textual information with named tags and is defined by Bray and Paoli et al. (2008). The set of XML tag names and the syntax rules for their use is referred to as the document schema. One way to formally define a schema for XML documents is using the W3C standard known as XML Schema (Thompson and Beech et al., 2004).

The XML Schemas of VODataService as well as VOResource and its other extensions are available from the IVOA schema repository[1] at any time. Parts of the schema appear within the main sections of this document; however, documentation nodes have been left out for the sake of brevity. Where the content of the pieces of schema embedded in this text diverges from the schema document in the IVOA document repository, the version in the schema repository is authoritative.

References to specific elements and types defined in the VOResource schema include the namespace prefix *vr* as in *vr:Resource* (a type defined in the VOResource schema).

1 Introduction

The VOResource standard (Plante and Demleitner et al., 2018) provides a means of encoding resource metadata as defined by DataCite (DataCite Metadata Working Group, 2016) and the VO-specific IVOA Resource Metadata (Hanisch and IVOA Resource Registry Working Group et al., 2007) in XML. VOResource uses XML Schema (Thompson and Beech et al., 2004) to define most of the XML syntax rules (while a few of the syntax rules are outside the scope of Schema). VOResource also describes mechanisms for creating extensions to the core VOResource metadata. This allows for the standardization of new metadata for describing specialized kinds of resources in a modular way without deprecating the core schema or other extensions.

[1] http://www.ivoa.net/xml

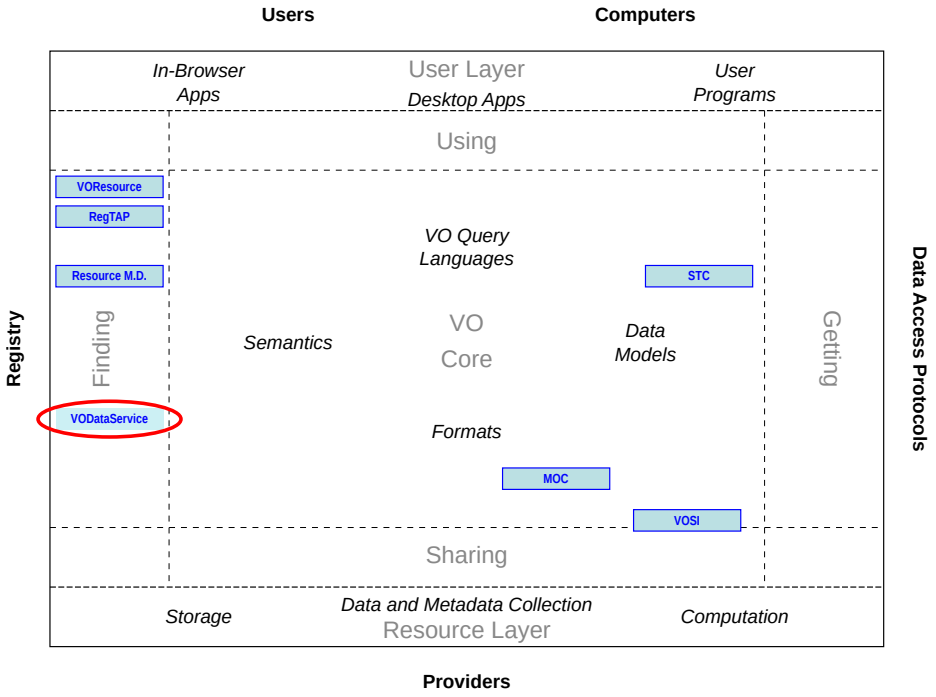

Figure 1: Architecture diagram for VODataService

This document defines one such extension referred to as VODataService. It provides types to define data services, their underlying tabular structures, their service interfaces, and the location of the data served in space, time, and energy.

The remainder of the document introduces the use cases addressed by this specification, then provides a high-level overview over the concepts and usage patterns in sect. 2 and finally discusses the concrete classes in sect. 3.

1.1 The Role in the IVOA Architecture

Fig. 1 shows the role VODataService plays within the IVOA Architecture (Arviset and Gaudet et al., 2010).

VODataService directly depends on the following other VO standards (unless specified otherwise, the dependency is on the major version of the cited standard rather than on the exact version):

VOResource, v1.1 (Plante and Demleitner et al., 2018) VOResource gives the fundamental types and structures extended here.

STC, v1.33 (Rots, 2007) The deprecated mechanism for declaring coverage through STCResourceProfile still uses concepts from version 1 of the IVOA data model

for Space-Time Coordinates. The updated mechanism has no such dependence any more.

VODataService is closely related to the following other VO standards:

VOSI, v1.1 (Graham and Rixon et al., 2017) VODataService defines the schema for the responses on the table metadata endpoint. It also defines the ParamHTTP interface type currently used in the capabilities of most standard protocols.

RegTAP, v1.1 (Demleitner and Harrison et al., 2019) RegTAP maps the concepts defined here into a relational structure. In that sense it is the user interface to what is specified here. RegTAP will need an update to support the space-time constraints added here.

MOC, v1.1 (Fernique and Boch et al., 2019) Multi-Order coverage maps are used by VODataService to communicate spatial coverage.

1.2 Purpose

The purpose of this extension is to define common XML Schema types – in particular new resource types – that are useful for describing data collections and services that access data. One aspect of such a description is the resource's *coverage*: the parts of the sky with which the data is associated and the time and frequency ranges that were observed or modeled to create the data. Another important aspect is the detailed metadata for tables underlying the resource, including names, types, UCDs (Derriere and Preite Martinez et al., 2005), units, and textual descriptions for the columns making them up.

Resource records using VODataService types are commonly used to register services that support standard IVOA data access layer protocols such as Simple Image Access (Dowler and Bonnarel et al., 2015) and Simple Cone Search (Plante and Williams et al., 2008). As of October 2018, there are more than 20000 resources of type `vs:CatalogService` in the VO Registry.

While other VOResource extensions define the protocol-specific metadata (encapsulated as a standard *capability* from VOResource), the general service resource description shares the common data concepts such as coverage and tabular data. Note, however, that a service described using the VODataService schema need not support any standard protocols. With the VODataService extension schema plus the core VOResource schema, it is possible to describe a custom service interface that accesses data.

As a legal extension of VOResource, the use of VODataService is subject to the rules and recommendations for XML (Bray and Paoli et al., 2008), XML Schema (Thompson and Beech et al., 2004), and VOResource itself.

1.3 Additional Use Cases for Version 1.2

In the following, we collect use cases that guided the development of VODataService to its version 1.2. We do not formally derive requirements from them but briefly note which new features enable or facilitate the specific use case.

A few of the changes in version 1.2 are necessary for consistency with other standards such as TAP (extendedType interpretation, requirement to use ADQL delimited identifier literals in names where appropriate) or VOTable (arraysize interpretation). These were obviously not guided by specific use cases.

What services have data for the Crab nebula covering the Hα line taken in the second half of 2015? In version 1.1, this use case would have been covered by the *stc:STCResourceProfile* type, which, however, was never properly standardised or widely adopted. In the current version, the *spatial*, *spectral*, and *temporal* children of *coverage* enable discovery by coverage on the various axes. It is worth noting that the spectral coverage is for the solar system barycenter, so this use case does *not* immediately enable the discovery of, say, Hα images of remote galaxies. Redshift correction has to be applied by the client based on knowledge about the object(s) investigated. At the time of writing, coverage also does not directly address non-celestial reference systems, although in particular planetary surfaces are considered in scope, and the coverage element's *@frame* attribute is defined to ensure non-ICRS coverages can safely be declared as the need arises.

Find all ObsCore services publishing data taken at the Telescope X. This use case could be satisfied in version 1.1 through the use of *vs:DataCollection* records and relationships to the respective TAP services. However, this scheme led to error-prone query patterns, and few such data collections were actually registered; see the IVOA Note on discovering data collections (Demleitner and Taylor, 2019) for details. To better support the scheme proposed there, version 1.2 adds the *vs:DataResource* and *vs:CatalogResource* types that identify a resource as data-like but permits the addition of various capabilities to the record (which *vs:DataCollection* did not). An analogous use case would be "Find all TAP services publishing tables from Gaia DR2".

Find a large-scale survey of sources between 20 and 40 GHz. While the spectral constraint is easily satisfied by the new coverage children, the "large-scale" part is much harder to operationalize. However, the plain table size often is a useful proxy in such discovery problems. The new *nrows* child of *vs:Table* communicates it.

1.4 Additional Use Cases for Future Versions

The following use cases were originally envisioned for VODataService 1.2, but were postponed because building multiply implemented solutions for them seemed likely to

unnecessarily delay the standardisation of, in particular, the STC part of the present document. They will likely be addressed by future versions of VODataService.

Find a resource that has sources in M51 down to 27 mag in V. The constraint about finding a resource that has V magnitudes for M51 is expressible using spatial coverage and the column's UCDs. To express something like "down to 27^{m}" one would at least need VOTable-style *VALUES* children for columns; however, metadata sufficient to address the next use case would certainly be sufficient here as well.

Plan a cross-service query. Systems like OGSA-DAI (Holliman and Alemu et al., 2011) perform orchestration of SQL-like queries between multiple services automatically, in particular cross-service JOINs. In order to work efficiently, such services need column statistics like histograms and the percentage of NULL values.

Find services serving time series. In the current registry model, users looking for spectra would select SSAP services. With the growing adoption of ObsCore (and a growing number of services abusing SSAP to serve time series), the model of selecting data types by constraining the service protocol no longer works; in the ObsCore example, clients now have to query all services and constrain the `dataproduct_type` column. However, for dataset types not overly common, well more than 90% of the services could be excluded without sending a query there based on a declaration of dataset types available in the Registry.

Facilitate discovery of full DALI services. The issue here is that DALI forsees synchronous and asynchronous endpoints as the standard case for many protocols – it already is standard for TAP. Also, several auxiliary endpoints (mostly defined in VOSI) are declared as separate capabilities and need to be matched with the functional endpoints. This matching is becoming a problem when multiple authentication schemes or mirror sites necessitate multiple sync/async pairs. A longer treatment of this problem has been published while this document was in WD (Demleitner, 2019), but at the time of writing the process of finding consensus has just begun, so again a normative solution has to be deferred to a later version of this standard.

2 The VODataService Data Model

The VODataService extension in general enables the description of two types of resources: Services that access data on the one side, and data being accessed through services on the other.

For simple services just publishing a simple resource – still a fairly common pattern – the metadata of the data published can be folded into the service record. Here is an example of such a record (abbreviated for the purposes of illustration)

that describes a service from the NASA Extragalactic Database (NED) that provides measured redshifts for a given object.

```
<ri:Resource xsi:type="vs:CatalogService" status="active"
           updated="2018-10-25T12:22:25" created="2005-10-14T01:46:00"
           xmlns:ri="http://www.ivoa.net/xml/RegistryInterface/v1.0"
           xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
           xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.1"
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0
                               http://www.ivoa.net/xml/VOResource/v1.0
                               http://www.ivoa.net/xml/VODataService/v1.1
                               http://www.ivoa.net/xml/VODataService/v1.1">

  <title>The NASA/IPAC Extragalactic Database</title>
  <shortName>NED_redshift</shortName>
  <identifier>ivo://ned.ipac/Redshift_By_Object_Name</identifier>
  <curation>
    <publisher>The NASA/IPAC Extragalactic Database</publisher>
    <contact>
      <name>Olga Pevunova</name>
      <email>contact@datacenter.edu</email>
    </contact>
  </curation>
  <content>
    <subject>redshift</subject>
    <subject>galaxies</subject>
    <description>
      NED is built around a master list of extragalactic objects for
      which cross- identifications of names have been established,
      accurate positions and redshifts entered to the extent possible,
      and some basic data collected. This service will return recorded
      redshifts for a given object.
    </description>
    <referenceURL>
        http://nedwww.ipac.caltech.edu/help/data_help.html#zdat
    </referenceURL>
    <type>BasicData</type>
    <contentLevel>Research</contentLevel>
  </content>

  <capability>
    <interface xsi:type="vs:ParamHTTP">
      <accessURL use="base">
http://nedwww.ipac.caltech.edu/cgi-bin/nph-datasearch?search_type=Redshifts&amp;
      </accessURL>
      <queryType>GET</queryType>
      <resultType>application/xml+votable</resultType>
      <param use="required">
        <name>objname</name>
        <description>Name of object</description>
        <dataType>string</dataType>
      </param>
      <param use="required">
```

```
52          <name>of</name>
53          <description>
54                  Output format parameter, must be "xml_main" for
55                  VOTable output.
56          </description>
57          <dataType>string</dataType>
58        </param>
59      </interface>
60    </capability>
61
62    <coverage>
63      <spatial>0/0-11</spatial>
64      <!-- arbitrary upper temporal limit for "ongoing acquisition" -->
65      <temporal>33282 100000</temporal>
66      <!-- Only radio and optical redshifts, here -->
67      <spectral>4e-28 3e-23</spectral>
68      <spectral>2.4e-19 5e-19</spectral>
69      <waveband>Radio</waveband>
70      <waveband>Optical</waveband>
71    </coverage>
72
73    <tableset>
74      <schema>
75        <name>default</name>
76        <table type="output">
77          <name>default</name>
78          <column>
79            <name>No.</name>
80            <description>
81               A sequential data-point number applicable to this list only.
82            </description>
83            <ucd>meta.number</ucd>
84            <dataType xsi:type="vs:VOTableType">int</dataType>
85          </column>
86          <column>
87            <name>Name in Publication</name>
88            <description>
89               The object's name in NED's standard format, of the object
90               to which the data apply.
91            </description>
92            <ucd>meta.id;name</ucd>
93            <dataType xsi:type="vs:VOTableType" arraysize="*">char</dataType>
94          </column>
95          <column>
96            <name>Published Velocity</name>
97            <description>
98              The radial velocity , derived from derived from the shift
99              of some spectral feature, in km/sec
100           </description>
101           <unit>km/sec</unit>
102           <ucd>src.spect.dopplerVeloc</ucd>
103           <dataType xsi:type="vs:VOTableType">int</dataType>
104         </column>
```

```
105       </table>
106     </schema>
107   </tableset>
108 </ri:Resource>
```

This example illustrates some of the features of the VODataService extension:

1. The specific type of resource indicated by the value of the *xsi:type* attribute; in this case *vs:CatalogService* indicates that this is describing a service that accesses tabular data (line 1).

2. The extra namespace declaration for VODataService metadata with the canonical prefix (line 5).

3. The location of the VOResource-related schema documents used by this description (line 7ff.)

4. The core VOResource metadata (line 12ff.)

5. An interface described by the VODataService type *vs:ParamHTTP*; this type can indicate input arguments the service supports (line 40ff.)

6. A description of the coverage, including quantitative coverage plus waveband keywords (line 62ff.)

7. A description of the table that is returned by the service (line 73ff.)

2.1 The Schema Namespace and Location

The namespace associated with VODataService extensions is

`http://www.ivoa.net/xml/VODataService/v1.1.`

As required by the IVOA schema versioning policies (Harrison and Demleitner et al., 2018), this namespace is identical to the one associated with version 1.1 of this document. It is regrettable that a misleading minor version is present in the namespace URI, but dropping it would break existing software for creating and processing VODataService instance documents. Hence, the namespace URI ending in `1.1` is also used for schema versions 1.2, 1.3, and so forth.

Resolving the VODataService namespace URI will redirect to a schema document having the actual version number (for the schema associated with this document version, this will end in VODataService-1.2.xsd). Following the schema versioning policies, the minor version will be declared in the *version* attribute of this file's root element. This information should not in general be used in production software; all versions with the above schema URI are compatible with each other in the sense defined in the IVOA schema versioning policies.

Authors of VOResource instance documents may choose to provide a location for the VOResource XML Schema document and its extensions using the

xsi:schemaLocation attribute. While authors are free to choose a location (as long as it resolves to the schema document), this specification recommends using the VODataService namespace URI as its location URL (as illustrated in the example above), as in

```
xsi:schemaLocation="http://www.ivoa.net/xml/VODataService/v1.1
                    http://www.ivoa.net/xml/VODataService/v1.1"
```

The canonical prefix for VODataService is *vs*; this means, in particular, that in non-XML contexts (e.g., relational mappings like RegTAP) the VODataService types *must* be qualified with vs:.

As recommended by the VOResource standard, the VODataService schema sets *elementFormDefault* to *unqualified*. This means that element names defined in this schema may not be used with a namespace prefix. The only place the namespace prefix must be used is the type names given as the value of an *xsi:type* attribute.

2.2 Summary of Metadata Concepts

VODataService defines several resource types and some auxiliary classes necessary to describe resources of these new types.

2.2.1 Auxiliary Classes

The VODataService type *vs:Coverage* allows the declaration of the physical coverage of a resource on the sky (or on spherical bodies), in time, and in the energy of the messenger particle. In addition, the element should contain a rough indication of the messenger type (e.g., "Optical"), and it can contain a link to a footprint endpoint and an indication of the spatial resolution within a service.

VODataService has several classes for the declaration of the tabular structure underlying a service; this can be the tables in a TAP-accessible resource, or it can relate to the data structure returned by the service. This metadata is held in *vs:TableSet*-typed elements consisting of one or more (possibly anonymous) *vs:TableSchema* instances. These have some very basic metadata (name, title, description, utype) and otherwise contain *vs:Table* instances. These in turn have simple metadata, but mainly give column metadata (including UCDs and units) in *vs:TableParam*-typed children. These are the basis for enabling discovery queries like "find all resources having redshifts and far infrared fluxes".

Note that table and schema metadata is deliberately shallow. If the main resource metadata is not enough to discover the table – as is fairly typical when a TAP service contains multiple unrelated tables –, data providers should define separate records for them as described in sect. 2.2.3.

VODataService further defines a specialized interface type (inheriting from *vr:Interface*) called *vs:ParamHTTP*. This type is used to describe straightforward HTTP interfaces directly operating on arguments encoded as *name=value* pairs.

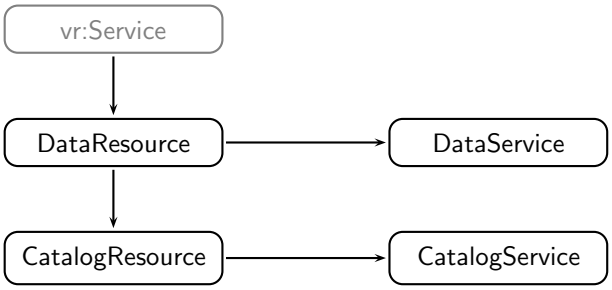

Figure 2: The four major resource classes in VODataService and their derivation tree

Such interface declarations can enumerate a service's input arguments, which enables clients to generate simple user interfaces from VOSI capabilities responses.

To be able to express the types of table columns and service arguments alike, VODataService defines several type systems. All of these are basically just enumerations of type names, possibly with some additional metadata like VOTable-style array sizes. In new resource records, only *vs:SimpleDataType* (for ParamHTTP parameters on non-DALI interfaces) and *vs:VOTableType* (for table columns) should be used.

2.2.2 VODataService Resource Classes

While no common VO service discovery pattern relies on the XSD type of the resource,[2] resource record authors should nevertheless choose appropriate types for their resources. At the very least, this helps Registry maintenance.

VODataService provides four major resource classes; their derivation tree is shown in Fig. 2. The vertical distinction in that figure reflects whether a resource has an associated tableset (DataX vs. CatalogX); you would typically use the DataX types when a resource does not naturally admit a relational structure. The classical example for this would be a collection of files on an FTP server. CatalogX, on the other hand, has an associated tableset. That does not mean that it is limited to what is conventionally thought of as a "catalogue", i.e., a table of data on astronomical objects. On the contrary, CatalogX should also be used for collections of images, spectra, time series, etc, as long as their metadata is sufficiently structured. That a data collection is published through the standard IVOA discovery protocols (ObsCore, SIAP, SSAP) certainly is a strong indication that this requirement is satisfied.

The horizontal distinction (XResource vs. XService) is somewhat more subtle and will be discussed in sect. 2.2.3.

Two further resource classes defined VODataService 1.1, *vs:DataCollection* and *vs:StandardSTC*, are deprecated in version 1.2. Resource record authors are

[2] Of course, in non-service discovery (e.g., authorities or standards), resource types are important.

requested to migrate or discard resource records using these deprecated types. If all such records have disappeared from the VO by version 1.3 of this specification, their type declarations may be removed from the schema.

2.2.3 Discovering Data Within Other Services

The content models of CatalogResource and CatalogService are identical. To understand why the two classes are present nevertheless, a short historical excursion is in order. A full treatment of this is found in the IVOA Endorsed Note on discovering data collections within services (Demleitner and Taylor, 2019), hereafter referred to as DDC.

In the early VO, there was almost throughout a $1:1$ relationship between data collections and services. For instance, a catalogue of variable stars had a single cone search interface, and the spectra from a given spectrograph had a single SSA interface. This is the model reflected by the CatalogService class, and it was, by and large, sufficient for IVOA's "simple" protocols (SCS, SIAP, SSAP, SLAP), although even these protocols have facilities for multi-collection services.

With the advent of services very naturally publishing what clearly are multiple different resources – TAP, with its multiple tables, and Obscore building on top of it are the prime examples –, this model proved inadequate; furthermore, publishers increasingly offered data through multiple interfaces: An object catalogue now quite typically is published as both a cone search service and within a TAP service; an image collection could have both SIAP and Obscore interfaces.

Thus, the relationship between data collections and services gradually became $n:m$. With this came the realisation that two classes of discovery need to be supported; for all-VO queries, validation, and the like, an enumeration of all *services* ("give me all SSAP services…") needs to be performed. For data discovery ("Where are images from instrument X of object Y?"), a selection over all *collections* needs to be made.

The solution eventually adopted for these problems is auxiliary capabilities as introduced in DDC. They provide stubs merely identifying an access protocol – at the same time identifying the capability as an auxiliary one – and access URLs, delegating all further service metadata to a separate record, linked to the original resource through an *isServedBy* relationship.

Thus, CatalogService-typed records should be used

- when a service serves a single data collection, in which case the metadata on the record describes the data collection (e.g., "These are observations of…").
- for a service serving multiple data collections, in which case the metadata on the record describes the service (e.g., "This is the TAP service of…"').

CatalogResource-typed records should be used for resources that only have auxiliary capabilities.

Here is a sketch of a resource record of a table within a TAP service::

```
<ri:Resource
  xsi:type="vs:CatalogResource"
  <title>Sample Table</title>
  <identifier>ivo://example/sample</identifier>
  <curation>...</curation>
  <content> ...
    <relationship>
      <relationshipType>IsServedBy</relationshipType>
      <relatedResource ivo-id="ivo://example/tap"
        >Example TAP service</relatedResource>
    </relationship>
  </content>
  <capability standardID="ivo://ivoa.net/std/TAP#aux">
    <interface role="std" xsi:type="vs:ParamHTTP">
      <accessURL use="base"
        >http://example.org/svcs/tap</accessURL>
    </interface>
 </capability>
  <coverage>...</coverage>
  <tableset>
    <schema>
      <name>someschema</name>
      <table>
        <name>someschema.sample</name>...
        ...
      </table>
    </schema>
  </tableset>
</ri:Resource>
```

A complete example can be found in DDC.

Note that it is legal to add auxiliary capabilities to CatalogService records as well. The classical example could be a cone search service serving a single catalogue that is also queriable within a larger TAP service.

Analogous considerations would apply to DataResource versus DataService, although at this point no obvious use cases have been identified; DataResource was included mainly for symmetry with CatalogResource.

3 The VODataService Metadata

This section enumerates the types and elements defined in the VODataService extension schema and describes their meaning.

3.1 VODataService Resource Types

For an overview of the systematics of the following resource types, please see Sect. 2.2.2.

3.1.1 DataResource

The *vs:DataResource* resource type is used to describe a resource containing generic astronomical data without a dominating tabular structure. An example might be a largely unstructured archive of various observations. Resources that have structured metadata tables (like most VO services) or are tabular in nature (like usual astronomical catalogues) should use *vs:CatalogResource*.

The type is derived from *vr:Service*, which means that instances can have capabilities. For *vs:DataResource*, only auxiliary capabilities (e.g., DataServices serving multiple DataResources) or plain capabilities with *vr:WebBrowser* interfaces should be given. Resources with a primary access mode dedicated to the resource's data content should use *vs:DataService*-typed resources.

In addition to *vr:Service*'s content, DataResource adds elements for declaring the observing facilities and/or instruments used to obtain the data, and it supports the declaration of the physical coverage of data via the *coverage* element.

vs:DataResource Type Schema Documentation

A resource publishing astronomical data.

This resource type should only be used if the resource has no common underlying tabular schema (e.g., an inhomogeneous archive). Use CatalogResource otherwise.

vs:DataResource Type Schema Definition

```
<xs:complexType name="DataResource" >
  <xs:complexContent >
    <xs:extension base="vr:Service" >
      <xs:sequence >
        <xs:element name="facility" type="vr:ResourceName" minOccurs="0"
                maxOccurs="unbounded" />
        <xs:element name="instrument" type="vr:ResourceName" minOccurs="0"
                maxOccurs="unbounded" />
        <xs:element name="coverage" type="vs:Coverage" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

vs:DataResource Extension Metadata Elements

Element *facility*

Type string with ID attribute: vr:ResourceName

Meaning The observatory or facility used to collect the data contained or managed by this resource.

Occurrence optional; multiple occurrences allowed.

Element *instrument*

Type string with ID attribute: vr:ResourceName

Meaning The instrument used to collect the data contain or managed by a resource.

Occurrence optional; multiple occurrences allowed.

Element *coverage*

Type composite: *vs:Coverage*

Meaning Extent of the content of the resource over space, time, and frequency.

Occurrence optional

3.1.2 DataService

The *vs:DataService* resource type has the same content model as *vs:DataResource*. It should be used instead of *vs:DataResource* when the resource's capabilities give access to (essentially) only the resource's data, as for instance "ftp service for the XY instrument", or for a service giving access to multiple resources; in this latter case, the resource-level metadata should pertain to the service itself rather than any specific data collection served by it.

As with *vs:CatalogResource*, instances SHOULD declare the metadata of the table(s) underlying the service or delivered by it in a *tableset* element.

Whenever the service operates on clearly definable tabular data, the resource should use the *vs:CatalogService* resource type in preference to *vs:DataService*, and that tabular structure should be given in a table set.

DataService's content model is exactly the one of *vs:DataResource*; please refer to sect. 3.1.1 for the motivation of this duplication.

3.1.3 CatalogResource

The *vs:CatalogResource* resource type is used to describe a resource containing astronomical data or metadata in a set of one or more tables. It extends *vs:DataResource* and thus has metadata on coverage as well as the facilities and instruments that produced the resource. Additionally, *vs:CatalogResource* instances SHOULD declare of the metadata of the table(s) making up the data *tableset* element.

As with *vs:DataResource*, this type should only be used if all capabilities declared in the resource are either auxiliary or nonstandard. This is typically the case for catalogues or data collections within larger TAP, ObsTAP, or perhaps SIAP services. When a service only publishes a single resource, use the *vs:CatalogService* type.

vs:CatalogResource Type Schema Documentation

A resource giving astronomical data in tabular form.

While this includes classical astronomical catalogues, this resource is also appropriate for collections of observations or simulation results provided their metadata are available in a sufficiently structured form (e.g., Obscore, SSAP, etc).

vs:CatalogResource Type Schema Definition

```
<xs:complexType name="CatalogResource" >
  <xs:complexContent >
    <xs:extension base="vs:DataResource" >
      <xs:sequence >
        <xs:element name="tableset" type="vs:TableSet" minOccurs="0" >
          <xs:unique name="CatalogService-schemaName" >
            <xs:selector xpath="schema" />
            <xs:field xpath="name" />
          </xs:unique>
          <xs:unique name="CatalogService-tableName" >
            <xs:selector xpath="schema/table" />
            <xs:field xpath="name" />
          </xs:unique>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

vs:CatalogResource Extension Metadata Elements

Element *tableset*

Type	composite: *vs:TableSet*
Meaning	A description of the tables that are accessible through this service.
Occurrence	optional
Comment	Each schema name must be unique within a tableset.

3.1.4 CatalogService

The *vs:CatalogService* resource type is used to describe a service publishing astronomical data or metadata based on tabular representations. Its relationship to *vs:CatalogResource* is as between *vs:DataResource* and *vs:DataService*: Use *vs:CatalogService* when either the resource's capabilties are exclusive to the resource described by the resource-level metadata ("SSAP service for the XY instrument") or if the service publishes multiple other CatalogResources; in that latter case, again, the resource-level metadata should not refer to any specific data collection contained.

This is the type that should be used for one-resource VO services.

CatalogService's content model is exactly the one of *vs:CatalogResource*; please refer to sect. 3.1.3 for the motivation of this duplication.

3.1.5 DataCollection

The *vs:DataCollection* type is deprecated and should no longer be used in new resource records. It was used in version 1.1 to define simple collections of data, much like *vs:CatalogResource* in the current version. It did not admit capabilities,

though, and since the addition of capabilities would essentially have created a CatalogService clone with a different child sequence, it was decided to abandon rather than repair it.

Existing *vs:DataCollection*-typed records should be migrated to records of type *vs:DataResource* or *vs:CatalogResource* as appropriate. If *accessURL* children are present in the *vs:DataCollection*s, they can be replaced with a plain capability with a *vr:WebBrowser*-typed interface.

This type may be removed from the schema when all resource records using it have vanished from the VO Registry.

3.1.6 StandardSTC

The *vs:StandardSTC* type is deprecated and should no longer be used in new resource records. Since the XML serialisation of the STC 1 data model was never promoted to an IVOA recommendation, there also is no properly standardised way of creating such records. Since no such records ever existed in the Registry, this type will probably be removed from the schema in version 1.3 of this specification.

3.2 Coverage in Space, Time, and Spectrum

The *vs:Coverage* type summarily describes how the data served is distributed on the sky, in energy, and in time.

In addition, there is the *waveband* element that originally contained a qualitative indication of the location of the resource's coverage on the electromagnetic spectrum. As more resources cover non-electromagnetic messengers, the element's meaning has somewhat shifted, and it now admits values from an extendable vocabulary of messengers[3] that, for instance, includes Neutrinos.

Historically, the quantitative footprints were expected to be given in the element of type *stc:STCResourceProfile*[4]. As discussed in Demleitner (2018), this expectation turned out to be erroneous, and the underlying standard, STC-X (Rots, 2005), never proceeded to become a recommendation. Hence, this version of VODataService deprecates the use of *STCResourceProfile*.

Instead, resource record authors are strongly encouraged to provide coverage information in the *spatial*, *spectral*, and *temporal* children of *coverage*.

Spatial coverage is conveyed as a MOC. To enable easy embedding into resource records written in VOResource (i.e., XML), VODataService uses the string serialisation defined in section 2.3.2 of Fernique and Boch et al. (2019).

By default, the MOCs are to be interpreted in the ICRS. Future extensions to non-celestial frames (e.g., on planet surfaces) will use the *frame* attribute. However, the only celestial reference system allowed is ICRS. If a resource's native coordinates

[3] http://www.ivoa.net/rdf/messenger

[4] Defined in http://www.ivoa.net/xml/STC/stc-v1.30.xsd

are given for another frame (e.g., Galactic or FK4 for some equinox), it is the resource record author's responsibility to convert the spatial coverage into the ICRS.

An important characteristic of MOCs is the order of the smallest scale (the "MOC resolution"). Higher orders yield more faithful representations of the actual coverage, but also lead to a possibly significant increase of the size of the serialized MOC. We suggest a "typical resolution" of the Registry of about a degree (i.e., MOC order 6), but resources are free to choose a higher maximum orders if appropriate and the resource record size remains reasonable.

Resources that need to communicate high-resolution spatial coverage, perhaps for some non-discovery use case, can use the *footprint* element with its *ivo-id* attribute set to `ivo://ivoa.net/std/moc` to declare a URL giving a footprint in one of the approved MOC serialisations and of arbitrary level and size.

Time and spectral coverage are modeled as unions of simple intervals over real numbers; the serialisation here is a space-separated pair of floating point numbers as governed by the DALI *interval* xtype.

Times are given in Barycentric Dynamical Time (TDB) at the solar system barycenter. They must be specified as Modified Julian Dates. Since discovery use cases in which high-precision times are required are not forseen, resource record authors are encouraged to pad their actual temporal coverage such that differences in time scales (of the order of 10s of seconds) or reference positions (of the order of minutes between ground-based observatories and the barycenter) do not matter. In other words, the temporal resolution of the Registry at this point should be assumed to be of order 10 minutes.

Deviating from common VO practice (which currently fairly consistently uses wavelengths of electromagnetic waves in vacuum), spectral limits are given in Joules of messenger energy. This is intended to allow seamless integration of non-electromagnetic messengers. The reference position for the spectral axis is the solar system barycenter. Again, discovery use cases on a level where the difference between reference frames of ground-based observatories versus the solar system barycenter matters are not forseen, and resource record authors are advised to pad their intervals on that level.

vs:Coverage Type Schema Documentation

A description of how a resource's contents or behavior maps to the sky, to time, and to frequency space, including coverage and resolution.

vs:Coverage Type Schema Definition

```
<xs:complexType name="Coverage" >
  <xs:sequence >
    <xs:element ref="stc:STCResourceProfile" minOccurs="0" />
    <xs:element name="spatial" type="vs:SpatialCoverage" minOccurs="0" />
    <xs:element name="temporal" type="vs:FloatInterval" minOccurs="0"
             maxOccurs="unbounded" />
    <xs:element name="spectral" type="vs:FloatInterval" minOccurs="0"
```

```
            maxOccurs="unbounded" />
    <xs:element name="footprint" type="vs:ServiceReference" minOccurs="0" />
    <xs:element name="waveband" type="xs:token" minOccurs="0"
            maxOccurs="unbounded" />
    <xs:element name="regionOfRegard" type="xs:float" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

vs:Coverage Metadata Elements

Element *stc:STCResourceProfile*

An STC 1.0 description of the location of the resource's data on the sky, in time, and in frequency space, including resolution. This is deprecated in favour of the separate spatial, temporal, and spectral elements.

This element is imported from another schema. See there for details.

Element *spatial*

Type a string with optional attributes

Meaning An ASCII-serialized MOC defining the spatial coverage of the resource.

Occurrence optional

Comment The MOC is to be understood in the ICRS reference frame unless a frame attribute is given. Resources should give the coverage at least to order 6 (a resolution of about one degree). The order should be chosen so as to keep the resulting MOC smaller than a few dozens of kB. If desired, a more precise MOC can be provided on a dedicated endpoint declared in the footprint element.

Element *temporal*

Type string of the form: *[+-]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][+-]?[0-9]+)? [+-]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][+-]?[0-9]+)?*

Meaning A pair of lower, upper limits of a time interval for which the resource offers data.

Occurrence optional; multiple occurrences allowed.

Comment This is written as for VOTable tabledata (i.e., whitespace-separated C-style floating point literals), as in "47847.2 51370.2". The limits must be given as MJD. While they are not intended to be precise, they are to be understood in TDB for the solar system barycenter. The total coverage of the resource is the union of all such intervals.

Element *spectral*

Type string of the form: *[+-]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][+-]?[0-9]+)? [+-]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][+-]?[0-9]+)?*

Meaning A pair of lower, upper limits of a spectral interval for which the resource offers data.

Occurrence optional; multiple occurrences allowed.

Comment This is written as for VOTable tabledata (i.e., whitespace-separated C-style floating point literals). The limits must be given in Joules of particle energies. While the limits are not intended to be precise, they are to be understood for the solar system barycenter.

Comment For instance, the Johnson V waveband (480 .. 730 nm) would be specified as "2.72e-19 4.14e-19"

Element *footprint*

Type a URI with optional attributes

Meaning A reference to a footprint service for retrieving precise and up-to-date description of coverage.

Occurrence optional

Comment The ivo-id attribute here refers to the standard in which the footprint is given. The only value defined by VODataService at this point is ivo://ivoa.net/std/moc, which indicates that retrieving the footprint URL will return a MOC (any IVOA-approved serialisation is legal). Note that the ivo-id attribute was intended to have a different function in VODataService 1.1. The current meaning is what implementors actually adopted.

Element *waveband*

Type string: *xs:token*

Meaning A name of a messenger that the resource is relevant for (e.g., was used in the measurements). Terms must be taken from the vocabulary at http://www.ivoa.net/rdf/messenger.

Occurrence optional; multiple occurrences allowed.

Comment It is a bit unfortunate that the element is still called waveband when it is now also covers non-electromagnetic messengers. It was deemed that this slight notional sloppiness is preferable to introducing new and deprecating old elements.

Element *regionOfRegard*

Type floating-point number: *xs:float*

Meaning A single numeric value representing the angle, given in decimal degrees, by which a positional query against this resource should be "blurred" in order to get an appropriate match.

Occurrence optional

Comment In the case of image repositories, it might refer to a typical field-of-view size, or the primary beam size for radio aperture synthesis data. In the case of object catalogues RoR should normally be the largest of the typical size of the objects, the astrometric errors in the positions, or the resolution of the data.

In order to avoid future ambiguity in the spatial coverage, we already add a *frame* attribute to the *spatial* element. In its absence, the MOC contained is for ICRS coordinates. Any value indicates non-celestial coordinates, where actual, interoperable values are not defined here.

vs:SpatialCoverage Type Schema Documentation

A coverage on a sphere. By default, this refers to the celestial sphere in the ICRS frame. Non-celestial frames are indicated by non-NULL values of the frame attribute.

vs:SpatialCoverage Type Schema Definition

```
<xs:complexType name="SpatialCoverage" >
  <xs:simpleContent >
    <xs:extension base="xs:token" >
```

```
    <xs:attribute name="frame" type="xs:token" />
   </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

vs:SpatialCoverage Attributes

frame

Type string: ***xs:token***

Meaning When present, the MOC is written in a non-celestial (e.g., planetary) frame. Note that for celestial coverages, ICRS must be used.

Occurrence optional

Comment VODataService 1.2 does not prescribe a vocabulary for what values are allowed here. As long as no such vocabulary is agreed upon, the frame attribute should not be set.

Coverage declaration makes use of three types, *vs:FloatInterval*, *vs:ServiceReference*, and *vs:SpatialCoverage*, defined as follows:

vs:FloatInterval Type Schema Documentation

An interval of floating point numbers.

This uses VOTable TABLEDATA serialisation, i.e., simply a pair of XSD floating point numbers separated by whitespace; note that software utilising non-XSD aware parsers has to perform whitespace normalisation itself here (in particular, for the internal whitespace).

vs:FloatInterval Type Schema Definition

```
<xs:simpleType name="FloatInterval" >
  <xs:restriction base="xs:token" >
    <xs:pattern
             value="[+-]?([0-9]+\.?[0-9]*|\.[0-9]+)([eE][+-]?[0-9]+)? [+-]?([0-9]+\.?[0-9]*|\.[0-9]+
  </xs:restriction>
</xs:simpleType>
```

vs:ServiceReference Type Schema Documentation

The service URL for a potentially registered service. That is, if an IVOA identifier is also provided, then the service is described in a registry.

vs:ServiceReference Type Schema Definition

```
<xs:complexType name="ServiceReference" >
  <xs:simpleContent >
    <xs:extension base="xs:anyURI" >
      <xs:attribute name="ivo-id" type="vr:IdentifierURI" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

vs:ServiceReference Attributes

ivo-id

Type an IVOA Identifier URI: vr:IdentifierURI

Meaning The URI form of the IVOA identifier for the service describing the capability refered to by this element.

Occurrence optional

3.3 Tabular Data

The *vs:TableSet* type can be used to describe a set of tables that are part of a single resource and can be considered functionally all located at a single site. While there is no expectation that the table set directly corresponds to the responses of tabular services, services should not include metadata for purely internal tables.

vs:TableSet Type Schema Documentation

The set of tables hosted by a resource.

vs:TableSet Type Schema Definition

```
<xs:complexType name="TableSet" >
  <xs:sequence >
    <xs:element name="schema" type="vs:TableSchema" minOccurs="1"
            maxOccurs="unbounded" >
      <xs:unique name="DataCollection-tableName" >
        <xs:selector xpath="table" />
        <xs:field xpath="name" />
      </xs:unique>
    </xs:element>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" />
</xs:complexType>
```

vs:TableSet Metadata Elements

Element *schema*

Type composite: *vs:TableSchema*

Meaning A named description of a group of logically related tables.

Occurrence required; multiple occurrences allowed.

Comment The name given by the "name" child element must be unique within this TableSet instance. If there is only one schema in this set and/or there is no locally appropriate name to provide, the name can be set to "default".

Comment This aggregation does not need to map to an actual database, catalogue, or schema, though the publisher may choose to aggregate along such designations. Particular service protocols may require stricter patterns.

The *vs:TableSchema* type groups tables that are logically related. For example, a single resource may provide access to several major astronomical catalogues

(e.g., SDSS, 2MASS, and FIRST) from one site, enabling high-performance cross-correlations between them. Each catalogue can be described in a separate *schema* element, using the elements from the *vs:TableSchema* type.

vs:TableSchema Type Schema Documentation

A detailed description of a logically related group of tables.

vs:TableSchema Type Schema Definition

```
<xs:complexType name="TableSchema" >
  <xs:sequence >
    <xs:element name="name" type="xs:token" minOccurs="1" />
    <xs:element name="title" type="xs:token" minOccurs="0" />
    <xs:element name="description" type="xs:token" minOccurs="0" />
    <xs:element name="utype" type="xs:token" minOccurs="0" />
    <xs:element name="table" type="vs:Table" minOccurs="0"
             maxOccurs="unbounded" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" />
</xs:complexType>
```

vs:TableSchema Metadata Elements

Element *name*

- *Type* string: *xs:token*
- *Meaning* A name for the group of tables.
- *Occurrence* required
- *Comment* This is used to uniquely identify the group of tables among several groups. If no title is given, this name can be used for display purposes.
- *Comment* If there is no appropriate logical name associated with this group, the name should be explicitly set to "default".

Element *title*

- *Type* string: *xs:token*
- *Meaning* A descriptive, human-interpretable name for the group of tables.
- *Occurrence* optional
- *Comment* This is used for display purposes. There is no requirement regarding uniqueness. It is useful when there are multiple schemas in the context (e.g., within a tableset; otherwise, the resource title could be used instead).

Element *description*

- *Type* string: *xs:token*
- *Meaning* A free text description of the group of tables that should explain in general how all of the tables in the group are related.
- *Occurrence* optional

Element *utype*

- *Type* string: *xs:token*
- *Meaning* An identifier for a concept in a data model that the data in this schema as a whole represent.
- *Occurrence* optional

Comment The form of the utype string depends on the data model; common forms are sequences of dotted identifiers (e.g., in SSA) or URIs (e.g., in RegTAP).

Element *table*

Type composite: *vs:Table*

Meaning A description of one table.

Occurrence optional; multiple occurrences allowed.

Each table in a schema is described in detail using the *vs:Table* type.

vs:Table Type Schema Definition

```
<xs:complexType name="Table" >
  <xs:sequence >
    <xs:element name="name" type="xs:token" minOccurs="1" />
    <xs:element name="title" type="xs:token" minOccurs="0" />
    <xs:element name="description" type="xs:token" minOccurs="0" />
    <xs:element name="utype" type="xs:token" minOccurs="0" />
    <xs:element name="nrows" type="xs:nonNegativeInteger" minOccurs="0" />
    <xs:element name="column" type="vs:TableParam" minOccurs="0"
             maxOccurs="unbounded" />
    <xs:element name="foreignKey" type="vs:ForeignKey" minOccurs="0"
             maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="type" type="xs:string" />
  <xs:anyAttribute namespace="##other" />
</xs:complexType>
```

vs:Table Attributes

type

Type string: *xs:string*

Meaning A name for the role this table plays. Recognized values include "output", indicating this table is output from a query; "base_table", indicating a table whose records represent the main subjects of its schema; and "view", indicating that the table represents a useful combination or subset of other tables. Other values are allowed.

Occurrence optional

vs:Table Metadata Elements

Element *name*

Type string: *xs:token*

Meaning The fully qualified name of the table. This name should include all catalogue or schema prefixes needed to sufficiently uniquely distinguish it in a query.

Occurrence required

Comment In general, the format of the qualified name may depend on the context; however, when the table is intended to be queryable via ADQL, then the catalogue and schema qualifiers are delimited from the table name with dots (.).

Element *title*

Type string: *xs:token*

Meaning A descriptive, human-interpretable name for the table.

Occurrence optional

Comment This is used for display purposes. There is no requirement regarding uniqueness.

Element *description*

Type string: *xs:token*

Meaning A free-text description of the table's contents

Occurrence optional

Element *utype*

Type string: *xs:token*

Meaning An identifier for a concept in a data model that the data in this table represent.

Occurrence optional

Comment The form of the utype string depends on the data model; common forms are sequences of dotted identifiers (e.g., in SSA) or URIs (e.g., in RegTAP).

Element *nrows*

Type *xs:nonNegativeInteger*

Meaning The approximate size of the table in rows.

Occurrence optional

Comment This is not expected to be exact. For instance, the estimates on table sizes databases keep for query planning purposes are suitable for this field.

Element *column*

Type composite: *vs:TableParam*

Meaning A description of a table column.

Occurrence optional; multiple occurrences allowed.

Element *foreignKey*

Type composite: *vs:ForeignKey*

Meaning A description of a foreign keys, one or more columns from the current table that can be used to join with another table.

Occurrence optional; multiple occurrences allowed.

Each column in a table can be described using the *vs:TableParam* type which is described in section 3.5. The foreign keys in the table that can be used to join it with another table can be described with the *vs:ForeignKey* type (section 3.3.2). A foreign key description should only refer to tables described within the current table set.

When the table set describes a set of TAP-accessible tables, the value of a *table*'s *name* child must be in a form immediately usable for use in ADQL or SQL queries. This corresponds to the analogous requirement for TAP_SCHEMA. This means that all qualifications (schema, catalogue) must be present, but also that when delimited identifiers are used as table names on the relational side, the quotes must be part of *name*'s value, and the capitalisation used in the DDL must be preserved.

The *vs:Table* also provides an attribute for indicating the role a table plays in the schema.

3.3.1 Unique Names for Tables

The definitions of the *tableset* elements used in the *vs:DataCollection* and *vs:CatalogService* types constrain certain names to be unique. In particular, all schema names within a *tableset* element must be unique, and all table names within a *schema* element must be unique. A schema and table may share a common name, such as "default". These constraints make it possible to uniquely locate the description of a schema or table within a VOResource description.

3.3.2 Foreign Keys

The *vs:ForeignKey* type is used to describe foreign keys in a table that allow it to be joined efficiently with another table. A foreign key is a set of columns that map to a corresponding set of columns in another table.

vs:ForeignKey Type Schema Documentation

A description of the mapping a foreign key – a set of columns from one table – to columns in another table.

When foreign keys are declared in this way, clients can expect that joins constrained with the foreign keys are preformed efficiently (e.g., using an index).

vs:ForeignKey Type Schema Definition

```
<xs:complexType name="ForeignKey" >
  <xs:sequence >
    <xs:element name="targetTable" type="xs:token" />
    <xs:element name="fkColumn" type="vs:FKColumn" minOccurs="1"
             maxOccurs="unbounded" />
    <xs:element name="description" type="xs:token" minOccurs="0" />
    <xs:element name="utype" type="xs:token" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

vs:ForeignKey Metadata Elements

Element *targetTable*
: *Type* string: *xs:token*
: *Meaning* The fully qualified name (including catalogue and schema, as applicable) of the table that can be joined with the table containing this foreign key.
: *Occurrence* required

Element *fkColumn*
: *Type* composite: *vs:FKColumn*

Meaning A pair of column names, one from this table and one from the target table that should be used to join the tables in a query.

Occurrence required; multiple occurrences allowed.

Element *description*

Type string: *xs:token*

Meaning A free-text description of what this key points to and what the relationship means.

Occurrence optional

Element *utype*

Type string: *xs:token*

Meaning An identifier for a concept in a data model that the association enabled by this key represents.

Occurrence optional

Comment The form of the utype string depends on the data model; common forms are sequences of dotted identifiers (e.g., in SSA) or URIs (e.g., in RegTAP).

In this model, the source of the foreign key is the current table being described (i.e., represented by the *table* element that contains the *vs:ForeignKey* description, and thus does not need to be named explicitly). The key that is described points to the table given by the *targetTable* child element. Each child *fkColumn* element then gives a pair of columns, one from the source table and one from the target table, that can be constrained to be equal in a query that joins the two tables.

3.3.3 Extending Table Metadata

It is envisioned that it may be useful in the future to provide richer metadata for describing tables within a VOResource description than what are defined in this document. This document recommends the use of the following extension mechanisms when richer descriptions are desired:

1. Use extended types by applying the *xsi:type* attribute to the *tableset*, *schema*, *table*, *column* and/or *dataType* elements. The values provided in the attributes must refer to an XML type legally extended from the types associated with these elements according to the rules of XML Schema (Thompson and Beech et al., 2004) and the VOResource specification.

2. Apply a globally-defined attribute from a schema other than VODataService (i.e. from a namespace other than `http://www.ivoa.net/xml/VODataService/v1.1`) to any of the *tableset*, *schema*, *table*, and/or *column* elements.

3. When the extended metadata is specific to how the table data is accessed via a particular service protocol, then the new metadata can be incorporated into a specific capability extension (as described in the VOResource specification). This extension may make use of the various names within the *tableset* to indicate where the extension metadata apply.

4. Use the *extendedType* attribute of the *dataType* element (see section 3.5.3) to indicate a more specific data type then those defined by the *vs:TableParam* type.

3.4 Interface Type Extension: ParamHTTP

The *vs:ParamHTTP* type is a specialized service interface description that extends the VOResource *vr:Interface* type (as recommended by VOResource 1.1, section 2.2). It describes a service interface that is invoked over HTTP via a GET or a POST in which the inputs are parameters encoded in multipart/form-data or application/x-www-form-urlencoded containers.

vs:ParamHTTP Type Schema Documentation

A service invoked via an HTTP Query (either Get or Post) with a set of arguments consisting of keyword name-value pairs.

Note that the URL for help with this service can be put into the service/referenceURL element.

vs:ParamHTTP Type Schema Definition

```
<xs:complexType name="ParamHTTP" >
  <xs:complexContent >
    <xs:extension base="vr:Interface" >
      <xs:sequence >
        <xs:element name="queryType" type="vs:HTTPQueryType" minOccurs="0"
                maxOccurs="2" />
        <xs:element name="resultType" type="xs:token" minOccurs="0" />
        <xs:element name="param" type="vs:InputParam" minOccurs="0"
                maxOccurs="unbounded" />
        <xs:element name="testQuery" type="xs:string" minOccurs="0" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

vs:ParamHTTP Extension Metadata Elements

Element *queryType*

Type	string with controlled vocabulary
Meaning	The type of HTTP request, either GET or POST.
Occurrence	optional; up to 2 occurrences allowed.
Terms	

GET

POST

Comment The service may indicate support for both GET and POST by providing 2 queryType elements, one with GET and one with POST. Since the IVOA standard DALI requires standard services to support both GET and POST, this piece of metadata is not useful in the description of standard DAL services and does not need to be given for those.

Element *resultType*

Type string: *xs:token*

Meaning The MIME media type of a document returned in the HTTP response.

Occurrence optional

Element *param*

Type composite: *vs:InputParam*

Meaning A description of a input parameter that can be provided as a name=value argument to the service.

Occurrence optional; multiple occurrences allowed.

Element *testQuery*

Type string: *xs:string*

Meaning An ampersand-delimited list of arguments that can be used to test this service interface; when provided as the input to this interface, it will produce a legal, non-null response.

Occurrence optional

Comment When the interface supports GET, then the full query URL is formed by the concatenation of the base URL (given by the accessURL) and the value given by this testQuery element.

The extension metadata defined in the schema definition above are all optional. Nevertheless, even when an *interface* instance does not include any of these extended child elements, the use of `xsi:type="vs:ParamHTTP"` indicates that the interface accessed via the URL given by the *accessURL* element complies with the general parameter-based protocol described in this section.

An important intended use of the *vs:ParamHTTP* type is describing the interface of an IVOA standard service protocol of the "simple" variety, such as the Simple Image Access Protocol (Dowler and Bonnarel et al., 2015). In particular, it is recommended that specifications that define how a standard service is registered in a registry *require* the use of the *vs:ParamHTTP* interface type when it is applicable.

Normally, a VOResource description indicates its support for a standard protocol with a *capability* element having a *standardID* attribute set to specific URI representing the standard. The standard will usually spell out the HTTP query type, the returned media type, and input parameters required for compliance; therefore, it is not necessary that the *vs:ParamHTTP* description provide any of the optional extended metadata, as they are already implied by the *standardID*. The description need only reflect the optional or locally unique features of the interface. In particular, description may include

- a *queryType* element for a type that is not required by the standard (as long as the required query type is supported as well),

- *param* elements for any optional parameters or local extended parameters (when allowed by the standard).

Listing required parameters is allowed, even when describing a standard interface, as long as these are consistent with the service specification and the corresponding *param* elements include the attribute `std="true"` (see section 3.5.1). The *param* elements for custom parameters that are not part of the standard (but are rather local customizations) should include the attribute `std="false"`.

3.5 Data Parameters

The VODataService schema provides several element types for describing different kinds of data parameters used in datasets and services, including service input parameters and table columns. The parameter types describe a parameter in terms of elementary metadata including name, data type, and meaning.

All the VODataService parameter types derive from the base type *vs:BaseParam* which defines all the common parameter metadata except the data type.

vs:BaseParam Type Schema Documentation

A description of a parameter that places no restriction on the parameter's data type.

As the parameter's data type is usually important, schemas normally employ a sub-class of this type rather than this type directly.

vs:BaseParam Type Schema Definition

```
<xs:complexType name="BaseParam" >
  <xs:sequence >
    <xs:element name="name" type="xs:token" minOccurs="0" />
    <xs:element name="description" type="xs:token" minOccurs="0" />
    <xs:element name="unit" type="xs:token" minOccurs="0" />
    <xs:element name="ucd" type="xs:token" minOccurs="0" />
    <xs:element name="utype" type="xs:token" minOccurs="0" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" />
</xs:complexType>
```

vs:BaseParam Metadata Elements

Element *name*

Type	string: *xs:token*
Meaning	The name of the parameter or column.
Occurrence	optional

Element *description*

Type	string: *xs:token*
Meaning	A free-text description of a parameter's or column's contents.
Occurrence	optional

Element *unit*

Type string: *xs:token*

Meaning The unit associated with the values in the parameter or column.

Occurrence optional

Element *ucd*

Type string: *xs:token*

Meaning The name of a unified content descriptor that describes the scientific content of the parameter.

Occurrence optional

Comment There are no requirements for compliance with any particular UCD standard. The format of the UCD can be used to distinguish between UCD1, UCD1+, and SIA-UCD.

Element *utype*

Type string: *xs:token*

Meaning An identifier for a concept in a data model that the data in this schema represent.

Occurrence optional

Comment The form of the utype string depends on the data model; common forms are sequences of dotted identifiers (e.g., in SSA) or URIs (e.g., in RegTAP).

Leaving the data type metadatum out of *vs:BaseParam* allows the different kinds of parameters derived from *vs:BaseParam* to restrict the allowed data types to specific sets. The subsections below describe the different data types associated with input parameters (*vs:InputParam*) and table columns (*vs:TableParam*). The XML types associated with their *dataType* elements derive from a common parent, *vs:DataType*.

vs:DataType Type Schema Documentation

A type (in the computer language sense) associated with a parameter with an arbitrary name

This XML type is used as a parent for defining data types with a restricted set of names.

vs:DataType Type Schema Definition

```
<xs:complexType name="DataType" >
  <xs:simpleContent >
    <xs:extension base="xs:token" >
      <xs:attribute name="arraysize" type="vs:ArrayShape" />
      <xs:attribute name="delim" type="xs:string" />
      <xs:attribute name="extendedType" type="xs:string" />
      <xs:attribute name="extendedSchema" type="xs:anyURI" />
      <xs:anyAttribute namespace="##other" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

vs:DataType Attributes

arraysize

Type string of the form: *([0-9]+x)*[0-9]*[0-9*]*

Meaning The shape of the array that constitutes the value.

Occurrence optional

Comment Leave arraysize empty for scalar values. In version 1.1, this defaulted to 1, which was intended to indicate a scalar. This is now deprecated; an arraysize of 1 should be avoided now, the future interpretation, in accord with VOTable, will be "array of size 1".

delim

Type string: ***xs:string***

Meaning A string that is used to delimit elements of an array value in InputParams.

Occurrence optional

Comment Unless specifically disallowed by the context, applications should allow optional spaces to appear in an actual data value before and after the delimiter (e.g., "1, 5" when delim=",").

Comment This should not be used for VOTable types; there, VOTable (typcially TABLEDATA) conventions for writing arrays are binding.

extendedType

Type string: ***xs:string***

Meaning The data value represented by this type can be interpreted as of a custom type identified by the value of this attribute.

Occurrence optional

Comment If an application does not recognize this extendedType, it should attempt to handle the value assuming the type given by the element's value. string is a recommended default type.

Comment This element may make use of the extendedSchema attribute to refine the identification of the type. extendedTypes without an extendedSchema mean VOTable xtypes as defined by DALI.

extendedSchema

Type a URI: ***xs:anyURI***

Meaning An identifier for the schema that the value given by the extended attribute is drawn from.

Occurrence optional

Comment This attribute is normally ignored if the extendedType attribute is not present. A missing extendedSchema indicates that extendedType is a VOTable xtype.

The content of an element of type ***vs:DataType*** is the name of the data type for the current parameter. When the element is explicitly a ***vs:DataType*** (as opposed to one of its derived types), there are no restrictions on the names that may be included.

A data type description can be augmented via a common set of ***vs:DataType*** attributes. The ***arraysize*** attribute indicates the parameter is an array of values of the named type. Its value describes the shape of the array. When there is no natural serialisation defined through the type system, the ***delim*** attribute may be

used to indicate a delimiter that should appear between elements of an array value. This attribute must not be combined with VOTableDataType, which always uses VOTable serialisation rules.

More descriptive information about the type can be provided via *extendedType* and *extendedSchema*, which provide an alternate data type name. The name given in the type's element content, then, represents a commonly understood "fallback" type. For instance, in VOTable, timestamps are serialised into strings, which implies a VOTableType of char with arraysize *, as supported by all VOTable readers. VOTable readers interpreting the timestamp xtype, however, can turn this string into a timestamp value native to its clients. Such readers will interpret a VOTable *FIELD*'s *xtype* attribute. Such information is reflected in *extendedType*.

Arbitrary information can also be provided via any prefix-qualified, globally defined attribute drawn from an XML Schema other than VODataService (by virtue of the *xs:anyAttribute* specification present on *vs:DataType*).

Note that in the derived parameter description types described below, the *dataType* element is optional. Its absence from the parameter description does *not* mean that the parameter can support any data type; rather, it means that the data type simply has not been provided (which may limit what an application can do with the parameter). If a parameter can truly support any data type, the *vs:BaseParam* type can be used directly when the context permits.

3.5.1 Input Parameters

Actual parameters are normally described with types derived from *vs:BaseParam*. The *vs:InputParam* is intended for describing an input parameter to a service or function. In previous versions of VODataService, input parameters were restricted to be defined in terms of simple data types, which are intended to be sufficient for describing an input parameter to a simple REST-like service or a function in a weakly-typed language. They are defined in *vs:SimpleDataType*:

vs:SimpleDataType Type Schema Documentation

A data type restricted to a small set of names which is imprecise as to the format of the individual values.

This set is intended for describing simple input parameters to a service or function.

vs:SimpleDataType Type Schema Definition

```
<xs:complexType name="SimpleDataType" >
  <xs:simpleContent >
    <xs:restriction base="vs:DataType" >
      <xs:enumeration value="integer" />
      <xs:enumeration value="real" />
      <xs:enumeration value="complex" />
      <xs:enumeration value="boolean" />
      <xs:enumeration value="char" />
```

```
      <xs:enumeration value="string" />
      <xs:attribute name="arraysize" type="vs:ArrayShape" />
      <xs:attribute name="delim" type="xs:string" default=" " />
      <xs:attribute name="extendedType" type="xs:string" />
      <xs:attribute name="extendedSchema" type="xs:anyURI" />
      <xs:anyAttribute namespace="##other" />
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

vs:SimpleDataType Attributes

arraysize
: *Type* string of the form: *([0-9]+x)*[0-9]*[0-9*]*
: *Meaning* See vs:DataType.
: *Occurrence* optional

delim
: *Type* string: ***xs:string***
: *Meaning* See vs:DataType.
: *Occurrence* optional
: *Default* a space character

extendedType
: *Type* string: ***xs:string***
: *Meaning* See vs:DataType.
: *Occurrence* optional

extendedSchema
: *Type* a URI: ***xs:anyURI***
: *Meaning* See vs:DataType.
: *Occurrence* optional

Since VODataService 1.2, input parameters can use any type system, where non-DALI-compliant services SHOULD use ***vs:SimpleDataType*** and DALI-compliant services SHOULD use ***vs:VOTableType***. To ensure schema validation catches mistakes, resource record authors are advised to declare the type system intended using ***xsi:type***; for instance, a service accepting a DALI point might declare:

```
<param std="true">
  <name>pos</name>
  <description>ICRS position of target object</description>
  <dataType arraysize="2"
    xsi:type="vs:VOTableType"
    extendedType="point">double</dataType>
</param>
```

The ***vs:InputParam*** class then is a ***vs:BaseParam*** furnished with additional metadata defining how to use the parameter and whether or not it is defined in the standard governing the capability the interface is in (if any):

vs:InputParam Type Schema Documentation

A description of a service or function parameter having a fixed data type.

DALI-compliant services should use VOTableType here, others should use SimpleDataType.

vs:InputParam Type Schema Definition

```
<xs:complexType name="InputParam" >
  <xs:complexContent >
    <xs:extension base="vs:BaseParam" >
      <xs:sequence >
        <xs:element name="dataType" type="vs:DataType" minOccurs="0" />
      </xs:sequence>
      <xs:attribute name="use" type="vs:ParamUse" default="optional" />
      <xs:attribute name="std" type="xs:boolean" default="true" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

vs:InputParam Attributes

use

Type string with controlled vocabulary

Meaning An indication of whether this parameter is required to be provided for the application or service to work properly.

Occurrence optional

Terms

required
: The parameter is required for the application or service to work properly.

optional
: The parameter is optional but supported by the application or service.

ignored
: The parameter is not supported and thus is ignored by the application or service.

Default optional

std

Type boolean (true/false): xs:boolean

Meaning If true, the meaning and behavior of this parameter is reserved and defined by a standard interface. If false, it represents an implementation-specific parameter that effectively extends the behavior of the service or application.

Occurrence optional

Default true

vs:InputParam Extension Metadata Elements

Element *dataType*

Type a string with optional attributes

Meaning A type of data contained in the parameter.
Occurrence optional

Here is an example for a description of an input parameter that might appear inside an *vs:ParamHTTP* interface description. As noted in section 3.4, a *param* element uses the *vs:InputParam* type to describe itself:

```
<param use="required">
  <name> radius </name>
  <description>
    search radius; returned objects are restricted to fall
    within this angular distance of the search position.
  </description>
  <ucd> phys.angSize </ucd>
  <dataType xsi:type="vs:SimpleDataType"> real </dataType>
</param>
```

3.5.2 Table Columns

The *vs:TableParam* is also derived from *vs:BaseParam*, and is designed for describing a column of a table.

vs:TableParam Type Schema Documentation

A description of a table parameter having a fixed data type.

vs:TableParam Type Schema Definition

```
<xs:complexType name="TableParam" >
  <xs:complexContent >
    <xs:extension base="vs:BaseParam" >
      <xs:sequence >
        <xs:element name="dataType" type="vs:TableDataType" minOccurs="0" />
        <xs:element name="flag" type="xs:token" minOccurs="0"
                maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="std" type="xs:boolean" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

vs:TableParam Attributes

std

Type boolean (true/false): xs:boolean
Meaning If true, the meaning and use of this parameter is reserved and defined by a standard model. If false, it represents a parameter specific to the data described If not provided, then the value is unknown.

Occurrence optional

vs:TableParam Extension Metadata Elements

Element *dataType*

Type *vs:DataType* with optional attributes

Meaning A type of data contained in the column

Occurrence optional

Element *flag*

Type string: *xs:token*

Meaning A keyword representing traits of the column. Recognized values include "indexed", "primary", and "nullable".

Occurrence optional; multiple occurrences allowed.

Comment While other values are allowed, the following semantics is defined by this specification: indexed – The column has an index on it for faster search against its values; primary – The values column in the column represent in total or in part a primary key for its table; nullable – the column may contain null or empty values.

A table column's data type is indicated with the *dataType* element with a name drawn from a standard set of names. Because *dataType*'s XML type, *vs:TableDataType*, is abstract, the element MUST include an *xsi:type* attribute. As discussed in sect. 3.5.3, this SHOULD be *vs:VOTableType* for VODataService 1.2 table columns.

When the tableset describes a set of TAP-accessible tables, the value of a *tableColumn*'s *name* child must be in a form immediately usable for use in ADQL or SQL queries. This corresponds to the analogous requirement for TAP_SCHEMA and is usually only a problem when delimited identifiers are used for column names on the relational side. In that case, the quotes must be part of *name*'s value, and the capitalisation used in the DDL must be preserved.

As examples, here are declarations of a column called "Dec" containing a double-valued declination, of the (deprecated) `size` column from TAP 1.0 which requires quotes because it otherwise clashes with a SQL reserved name, and a SIAP 1.0-compatible `wcs_cdmatrix` column that shows how multidimensional arrays are declared; note that in the last case, the content of *unit* pertains to *elements* of the array, not the array as a whole.

```
<column>
   <name> Dec </name>
   <description> the J2000 declination of the object </description>
   <ucd> pos.eq.dec </ucd>
   <dataType xsi:type="vs:VOTableType"> double </dataType>
</column>

<column>
  <name>"size"</name>
  <description>Length of variable length datatypes</description>
```

```
  <dataType xsi:type="vs:VOTableType">int</dataType>
  <flag>nullable</flag>
</column>

<column>
  <name>wcs_cdmatrix</name>
  <description>World coordinates at WCS reference pixel</description>
  <unit>deg</unit>
  <ucd>VOX:WCS_CoordRefValue</ucd>
  <dataType arraysize="2x2" xsi:type="vs:VOTableType">double</dataType>
  <flag>nullable</flag>
</column>
```

3.5.3 Table Column Data Types

The VODataService schema defines two type systems that derive from *vs:TableDataType*: *vs:VOTableType* and *vs:TAPType*. Following the move to VOTable-compatible type descriptions in TAP 1.1 (Dowler and Rixon et al., 2019), this version of VODataService deprecates the use of *vs:TAPType*. New software should only generate *vs:VOTableType*s in *tableset*s.

vs:VOTableType Type Schema Documentation

A data type supported explicitly by the VOTable format

vs:VOTableType Type Schema Definition

```
<xs:complexType name="VOTableType" >
  <xs:simpleContent >
    <xs:restriction base="vs:TableDataType" >
      <xs:enumeration value="boolean" />
      <xs:enumeration value="bit" />
      <xs:enumeration value="unsignedByte" />
      <xs:enumeration value="short" />
      <xs:enumeration value="int" />
      <xs:enumeration value="long" />
      <xs:enumeration value="char" />
      <xs:enumeration value="unicodeChar" />
      <xs:enumeration value="float" />
      <xs:enumeration value="double" />
      <xs:enumeration value="floatComplex" />
      <xs:enumeration value="doubleComplex" />
      <xs:attribute name="arraysize" type="vs:ArrayShape" />
      <xs:attribute name="delim" type="xs:string" default=" " />
      <xs:attribute name="extendedType" type="xs:string" />
      <xs:attribute name="extendedSchema" type="xs:anyURI" />
      <xs:anyAttribute namespace="##other" />
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

vs:VOTableType **Attributes**

arraysize
: *Type* string of the form: *([0-9]+x)*[0-9]*[0-9*]*
: *Meaning* See vs:DataType.
: *Occurrence* optional

delim
: *Type* string: ***xs:string***
: *Meaning* See vs:DataType.
: *Occurrence* optional
: *Default* a space character

extendedType
: *Type* string: ***xs:string***
: *Meaning* See vs:DataType.
: *Occurrence* optional

extendedSchema
: *Type* a URI: ***xs:anyURI***
: *Meaning* See vs:DataType.
: *Occurrence* optional

When the actual column type is not well matched to a VOTable data type, authors are encouraged to use the ***extendedType*** attribute to refer to a more specific type. This will usually be a VOTable ***xtype*** as defined by DALI (Dowler and Demleitner et al., 2017). Using different extended type schemes is possible by setting ***extendedSchema*** to a non-empty value; data providers using this extension mechanism should explain their schema in an IVOA Note.

A Changes from previous versions

A.1 Changes since PR-20210223

- Sect. 3.1: Added prose on how to pick the appropriate resource type.
- Schema: Various minor style fixes that should have no impact on validation.

A.2 Changes since PR-20190715

- Sect. 3.1 and schema: dropped `vs:Waveband` and changed waveband to being controlled by a vocabulary that initially grows a generic Photon and a Neutrino concept over what the previous Waveband had.
- Sect. 3.3 and schema: table/@nrows is now constrained to be non-negative.
- Sect. 3.5.1 and schema: Now allowing any vs:DataType element to define vs:InputParams. In order to still ensure schema validation of type names, now advising to have an explicit xsi:type in param's dataTypes.

A.3 Changes since WD-20181026

- Sect. 2.2.3: Added a summary of the discovering data collections EN.
- Sect. 3.2 and schema: Spatial coverage now has a frame attribute.
- Sect. 3.1.3: Added a SHOULD requirement on CatalogResources to have a tableset.
- Sect. 3.4 and schema: Restricted interface/@testQuery to zero or one occurrences (this is because there is no clear semantics to having multiple of those).
- Sect. 3.2: Sanctioning the use of footprint/@ivo-id to indicate the footprint standard used (which, of course, totally goes against the semantics of ServiceReference underlying footprint).

A.4 Changes since REC-1.1

- Sect. 3.2 and schema: Deprecated STCResourceProfile and replaced it with *spatial*, *temporal*, and *spectral* elements.
- Sect. 3.1 and schema: Introduced new DataResource and CatalogResource resource types and wove them into the inheritance hierarchy to CatalogService; these are to be used for "dependent" resources.
- Sect. 3.2: Deprecated DataCollection and StandardSTC (which are no longer needed).
- Sect. 3.3 and schema: Added an nrows element to *vs:Table*.
- Sect. 3.3: Required inclusion of quotes for delimited identifiers in a SQL context.
- Sect. 3.5 and schema: DataType/@arraysize no longer defaults to 1, and the interpretation of arraysize=1 as a scalar is withdrawn. Use empty arraysize for scalars now.
- Sect. 3.5 and schema: DataType's delim attribute no longer defaults to blank. That would be very unfortunate with VOTable, where other conventions are in place (e.g., for string arrays). Now discouraging the use of delim outside of InputParams.
- Sect. 3.5.3: Deprecated TAPType.
- Sect. 3.5.3: extendedType is now defined to correspond to VOTable xtypes in the absence of extendedSchema.

- Sect. 3.3: No longer requiring unique table names within a tableset; uniquness is now required within a schema (actually, many services have been in violation of the old unique-within-tableset rule for a long time without operational difficulties); but then that's largely a moot point because for the main uses of tableset, fully qualified names are now required.
- Ported source to IVOATEX.

A.5 Changes since PR-20100916

- updated status for elevation to Recommendation.
- cleaned-up mis-labeled and mis-ordered change history.

A.6 Changes since PR-20100914

- added change history for PR-20100412.
- added Note about STC mark-up in 3.2
- reworded sentence describing content of *vs:DataType* in section 3.5.

A.7 Changes since PR-20100412

- fix numerous typos discovered in TCG review
- added section 1.1 to describe role of standard in the VO architecture, including diagram.
- corrected frequency range for the UV waveband
- corrected links to reference documents

A.8 Changes since PR-20090903

- added *testQuery* to *vs:ParamHTTP*
- in text, added explanation of *vs:Format*
- grammatical clean-up

A.9 Changes since WD-20090508 (v1.10)

- corrected errors in example in Introduction
- added *description* and *utype* elements to the *vs:ForeignKey* type for consistency with TAP.
- changed type names *vs:TAP* to *vs:TAPType* and *vs:VOTable* *vs:VOTableType*.

References

Arviset, C., Gaudet, S. and IVOA Technical Coordination Group (2010), 'IVOA Architecture Version 1.0', IVOA Note 23 November 2010.
http://doi.org/10.5479/ADS/bib/2010ivoa.rept.1123A

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

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E. and Yergeau, F. (2008), 'Extensible markup language (XML) 1.0 (fifth edition)', W3C Recommendation.
http://www.w3.org/TR/REC-xml/

DataCite Metadata Working Group (2016), 'DataCite metadata schema – documentation for the publication and citation of research data version 4.0', DataCite publication.
https://schema.datacite.org/meta/kernel-4.0/doc/DataCite-MetadataKernel_v4.0.pdf

Demleitner, M. (2018), 'A roadmap for space-time discovery in the VO registry', IVOA Note.
http://ivoa.net/documents/Notes/Regstc/20180208/NOTE-regstcnote-1.0-20180115.html

Demleitner, M. (2019), 'On the use of capabilities in the VO', IVOA Note.
http://ivoa.net/documents/caproles/

Demleitner, M., Harrison, P., Molinaro, M., Greene, G., Dower, T. and Perdikeas, M. (2019), 'IVOA Registry Relational Schema Version 1.1', IVOA Recommendation 11 October 2019.
https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.1011D

Demleitner, M. and Taylor, M. (2019), 'Discovering Data Collections Within Services Version 1.1', IVOA Note 20 May 2019.
http://doi.org/10.5479/ADS/bib/2019ivoa.rept.0520D

Derriere, S., Preite Martinez, A., Preite Martinez, A., Derriere, S., Gray, N., Mann, R., McDowell, J., Mc Glynn, T., Ochsenbein, F., Osuna, P., Rixon, G. and Williams, R. (2005), 'The UCD1+ controlled vocabulary Version 1.11', IVOA Recommendation 31 December 2005.
http://doi.org/10.5479/ADS/bib/2005ivoa.spec.1231D

Dowler, P., Bonnarel, F. and Tody, D. (2015), 'IVOA Simple Image Access Version 2.0', IVOA Recommendation 23 December 2015.
http://doi.org/10.5479/ADS/bib/2015ivoa.spec.1223D

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., Tody, D. and Demleitner, M. (2019), 'Table Access Protocol Version 1.1', IVOA Recommendation 27 September 2019.
https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.0927D

Fernique, P., Boch, T., Donaldson, T., Durand, D., O'Mullane, W., Reinecke, M. and Taylor, M. (2019), 'MOC - HEALPix Multi-Order Coverage map Version 1.1', IVOA Recommendation 07 October 2019.
https://ui.adsabs.harvard.edu/abs/2019ivoa.spec.1007F

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

Hanisch, R., IVOA Resource Registry Working Group and NVO Metadata Working Group (2007), 'Resource Metadata for the Virtual Observatory Version 1.12', IVOA Recommendation 02 March 2007, arXiv:1110.0514.
http://doi.org/10.5479/ADS/bib/2007ivoa.spec.0302H

Harrison, P., Demleitner, M., Major, B. and Dowler, P. (2018), 'XML Schema Versioning Policies Version 1.0', IVOA Endorsed Note 29 May 2018.
http://doi.org/10.5479/ADS/bib/2018ivoa.spec.0529H

Holliman, M., Alemu, T., Hume, A., van Hemert, J., Mann, R. G., Noddle, K. and Valkonen, L. (2011), Service Infrastructure for Cross-Matching Distributed Datasets Using OGSA-DAI and TAP, *in* I. N. Evans, A. Accomazzi, D. J. Mink and A. H. Rots, eds, 'Astronomical Data Analysis Software and Systems XX', Vol. 442 of *Astronomical Society of the Pacific Conference Series*, p. 579.
http://ads.ari.uni-heidelberg.de/abs/2011ASPC..442..579H

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

Rots, A. (2005), 'STC-X: Space-time coordinate (STC) metadata XML implementation version 1.00', IVOA Note.
http://ivoa.net/documents/latest/STC-X.html

Rots, A. H. (2007), 'Space-Time Coordinate Metadata for the Virtual Observatory Version 1.33', IVOA Recommendation 30 October 2007, arXiv:1110.0504.
http://doi.org/10.5479/ADS/bib/2007ivoa.spec.1030R

Thompson, H. S., Beech, D., Maloney, M. and Mendelsohn, N. (2004), 'XML schema part 1: Structures second edition', W3C Recommendation.
http://www.w3.org/TR/xmlschema-1/