International
Virtual
Observatory
Alliance

Describing Simple Data Access Services Version 1.1

IVOA Proposed Recommendation 2016-07-06

Working group
Registry

This version
http://www.ivoa.net/documents/SimpleDALRegExt/20160706

Latest version
http://www.ivoa.net/documents/SimpleDALRegExt

Previous versions
WD-20160525
REC-1.0
PR-20130911
PR-2012116
PR-20120517
WD-20110921

Author(s)
Raymond Plante, Markus Demleitner, Jesus Delago, Paul Harrison, Doug Tody

Editor(s)
Ray Plante, Markus Demleitner

Version Control
Revision 3593, 2016-10-04 11:33:39 +0200 (Tue, 04 Oct 2016)
http://volute.g-vo.org/svn/trunk/projects/registry/SimpleDALRegExt/SimpleDALRegExt.tex

Abstract

An application that queries or consumes descriptions of VO resources must be able to recognize a resource's support for standard IVOA protocols. This specification describes how to describe a service that supports any of the four typed data access protocols – Simple Cone Search (SCS), Simple Image Access (SIA), Simple Spectral Access (SSA), Simple Line Access (SLA) – using the VOResource XML encoding standard. A key part of this specification is the set of VOResource XML extension schemas that define new metadata that are specific to those protocols. This document describes rules for describing such services within the context of IVOA Registries and data discovery as well as the VO Support Interfaces (VOSI) and service self-description. In particular, this document spells out the essential mark-up needed to identify support for a standard protocol and the base URL required to access the interface that supports that protocol.

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 http://www.ivoa.net/Documents/.

Contents

Acknowledgements

This document has 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 Eurpean Commission's Sixth Framework Program via the Optical Infrared Coordination Network (OPTICON), and from BMBF grant 05A14VHA (GAVO).

Syntax Notation Using XML Schema

The Extensible Markup Language, or XML, is document syntax for marking textual information with named tags and is defined by the World Wide Web Consortium (W3C) Recommendation, XML 1.0 (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)

This document defines the VOResource schema using XML Schema. The full Schema documents are kept on the IVOA schema repository[1]. The files given there are authoritative and override XML schema fragments contained in specification in case of conflicts. Note that the schema files in the IVOA repository can change over time according to the rules laid down in Harrison and Demleitner et al. (2016).

Reference to specific elements and types defined in the VOResource schema include the namespaces prefix, *vr:*, as in *vr:Resource*. Reference to specific elements and types defined in the VODataService extension schema include the namespaces prefix, *vs:*, as in *vs:ParamHTTP*. Use of the *vs:* prefix in compliant instance documents is strongly recommended, particularly in the applications that involve IVOA Registries (Benson and Plante et al., 2009).

1 Introduction

Four data access service protocols play a key role in discovering data in the VO:

- Simple Cone Search (Plante and Williams et al., 2008) – searches a catalog for sources or observations that are within a given distance of a sky position.
- Simple Image Access (Dowler and Bonnarel et al., 2015) – searches an archive for spatially resolved data (like images and cubes) that overlap a given region of sky.
- Simple Spectral Access (Tody and Dolensky et al., 2012) – searches an archive for spectra of positions within a given region of sky.
- Simple Line Access (Osuna and Salgado et al., 2010) – searches a catalog specializing in descriptions of spectral line transitions.

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

They are called "simple" because a typical query can be formed using only a few search parameters encoded into a URL (i.e., an HTTP GET request). Their power for data discovery comes from the ability of an application to form a single query according to the rules of one of these protocols and send it to multiple services selected, say, for their relevence to a scientific topic which support that protocol. The results collected from those services, in effect then, represent all the relevent data of that type known to the VO. Thus, the key for an application wishing to do a comprehensive search of the VO is to discover all of the services that support the particular standard protocol.

Service discovery in the VO is done via a searchable registry as described by the Registry Interfaces standard (Benson and Plante et al., 2009) – i.e., a searchable repository of descriptions of resources in VO. These descriptions are comprised of common standard metadata as specified in the Resource Metadata document (Hanisch and IVOA Resource Registry Working Group et al., 2007) that capture information about what a resource contains or does and who provides it. A standard registry encodes these descriptions using the VOResource XML Schema (Plante and Benson et al., 2008). Service resources in particular include capability metadata that describe the functionality it supports along with interface metadata that describe how to access that functionality. It is within the capability metadata that it is possible to indicate support for a particular standard protocol.

Capability metadata play an important role beyond just identifying support for a standard interface. More generally, they describe how the service behaves, and if applications are to make use of this information in an automated way, the behavior must be described using standardized metadata. In general, the metadata necessary for describing that behavior will be specific to the particular kind of service. In the case of a standard protocol, in which it is common that some variation in behavior is allowed while still being in compliance, it can be important to an application to know how a service complies with the standard for two reasons:

1. The application may wish to search for and select services that support a particular protocol feature. For example, an application may wish to find image services that can create cut-outs on-the-fly.

2. The application may wish to plan its use of the service according its limitations, such as the maximum region of sky that can be searched in one query.

It is important to note that the relevent behavioral differences between separate services that support a common protocol–and thus the metadata used to describe those behaviors–will be specific to that protocol. That is, for example, the ability to create image cut-outs is irrelevent to the Simple Cone

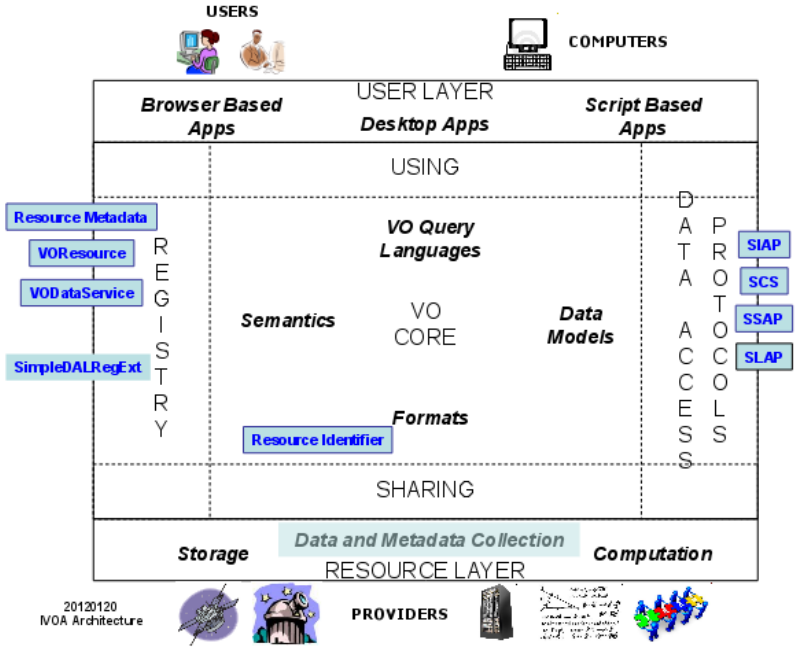

Figure 1: SimpleDALRegExt in the IVOA Architecture

Search protocol. Consequently, it is necessary to define protocol-specific metadata to adequately describe a service's support for that protocol. This document defines such capability metadata for SCS, SIA, SSA, and SLA.

This document describes for each of the standard data access protocols – SCS, SIA, SSA, and SLA – precisely how to describe a service that supports one of the protocols in terms of the VOResource XML encoding standard. This specification is intended to be applicable wherever VOResource records are used, but in particular, it is intended as the standard for encoding resource descriptions within an IVOA-compliant registry and for encoding capability metadata available through the VO Support Interfaces VOSI (Graham and Rixon et al., 2011).

1.1 The Role in IVOA Architecture

The IVOA Architecture (Arviset and Gaudet et al., 2010) provides a high-level view of how IVOA standards work together to connect users and applications with providers of data and services, as depicted in the diagram in Fig. 1.

In this architecture, data access protocols provide the means for users (via the User Layer) to access data from archives. Of particular importance are the standard protocols, SCS, SIA, SSA, and SLA, which allow a generic user tool to find data in any archive that supports those protocols. Registries

provide to tools in the User Layer a means to discover which archives support the standard protocols. A registry is a repository of descriptions of resources, such as standard services, that can be searched based on the metadata in those descriptions.

The Registry enables applications in the User Layer to discover archives in the Resource Layer and the services they provide for accessing data, particularly those that support the standard data access protocols like SIAP, SCS, SSAP, and SLAP (illustrated on the right). The registry metadata model standards (in blue text and boxes on the left) give structure to the information that enables that discovery. In particular, the SimpleDALRegExt standard defines the metadata used to describe standard data access services of the types listed on the right.

Resource descriptions have a well-defined structure: the core concepts are defined in the Resource Metadata standard (Hanisch and IVOA Resource Registry Working Group et al., 2007), and the format is defined by the VOResource XML standard (Plante and Benson et al., 2008). Additional metadata specialized to describe a specific kind of service are defined via extensions to the VOResource core XML Schema. SimpleDALRegExt is one such extension specifically for describing SCS, SIA, SSA, and SLA services in the registry.

1.2 Dependencies on Other Standards

This specification relies directly on other IVOA standards in the following ways:

VOResource, v1.03 (Plante and Benson et al., 2008)
: Descriptions of services that support the standard protocols are encoded using the VOResource XML Schema. The protocol-specific schemas defined in this document are extensions of the VOResource core schema.

Typed DAL Protocols
: The standards Simple Cone Search, v1.03 (Plante and Williams et al., 2008), Simple Image Access, v1.0 (Harrison and Tody et al., 2009), Simple Image Access, v2.0 (Dowler and Bonnarel et al., 2015), Simple Spectral Access, v1.04 (Tody and Dolensky et al., 2012), and Simple Line Access, v1.0 (Osuna and Salgado et al., 2010) describe the metadata concepts that should be included in a description of a service that supports the specification. We expect future versions of these standards to provide their own metadata schemes. Unless they do, however, the relevant metadata scheme from this document should be used.

VODataService, v1.1 (Plante and Stébé et al., 2010)
: The interface to the standard protocol functionality is described

with a specialized Interface type, vs:ParamHTTP, which is defined in the VODataService XML Schema, an extension to VOResource. This document also recommends describing the service using VODataService resource type, *vs:CatalogDataService*.

This specification refers to other IVOA standards:

Registry Interfaces, v1.0 (Benson and Plante et al., 2009)
: A registry that is compliant with both this specification and the Registry Interfaces standard will encode service resource descriptions according to the recommendations in this document.

VO Support Interfaces, v1.0 (Graham and Rixon et al., 2011)
: A service that supports one of the target protocols as well as the capability metadata retrieval method defined by VOSI is compliant with this specifcation if the capability metadata are encoded according the recommendations in this document.

Unlike with the previously mentioned specifications, this specification may apply to later versions of the RI and VOSI standards.

2 The Common Data Model for Simple DAL Services

This section describes common requirements for describing the target DAL services as VOResource records.

To be recognized as a service, the DAL service resource must be described as a resource type of *vr:Service* (defined in the VOResource schema) or one of its legal sub-types. As specified in the VOResource specification, the resource type is set by setting the *xsi:type* attribute on the element representing the root of the VOResource record to the namespace-qualified resource type name.

As the DAL services respond to queries with tables of available data products, their Registry records will typically be of the resource type *vs:CatalogService* (defined in the VODataService extension schema). In this case, record authors are encouraged to include a full description of the columns in the table returned in query response (assuming full verbosity). The *vs:CatalogService* resource type also allows the record to provide sky coverage information which authors are also encouraged to provide; an exception to this would be for pure SLA services as the spectral line catalogs they serve are not strictly sky observations.

The VOResource record must include a *capability* element that describes the services support for the DAL protocol. The contents of the element is described in section 3. In all cases, the value of the *capability*

Note

In VO practice, many clients discover the standard endpoints by looking for *capability* elements with the *standardID* of the protocol they are interested in and then locating a *vs:ParamHTTP*-typed *interface* in it without regard for it being marked up with `role="std"`. Resource record authors therefore should not include non-standard *vs:ParamHTTP* interfaces in capabilities with the *standardID*s defined here.

element's *standardID* unambiguously identifies the service's support for the particular DAL protocol. The resource may include other *capability* elements to describe support for other protocols.

The *capability* element describing support for the DAL protocol must include a child *interface* element that describes support for the required DAL interface; the *xsi:type* attribute on that element must be set to *vs:ParamHTTP*, and the role attribute must be set to `"std"`. A *accessURL* element within that *interface* must be set to the base URL, as defined in the DAL protocol specification, that provides access to the standard DAL protocol. It is not necessary to provide the *use* attribute to the *accessURL* element (as its value can be assumed); however, when it is provided, it must be set to `"base"`. Similarly, it is not necessary to provide the *interface* element with *queryType* or *resultType* elements; however, when provided, their values should be `"GET"` and `"application/x-votable+xml"`, respectively. The *vs:ParamHTTP* allows one to describe input parameters supported by the service; description authors are encouraged to list the optional parameters and any custom parameters supported by the instance of the service.

Here is a sample interface description for a simple DAL service.

```
<interface xsi:type="vs:ParamHTTP" role="std">
  <accessURL use="base">
    http://adil.ncsa.uiuc.edu/cgi-bin/voimquery?survey=f&
  </accessURL>

  <!-- here is a standard, optional parameter -->
  <param use="optional" std="true">
    <name>CFRAME</name>
    <description>request to shift to a given coordinate frame.</description>
    <dataType>string</dataType>
  </param>

  <!-- here is a site-specific parameter that this service supports -->
  <param use="optional" std="false">
    <name>FREQ</name>
    <description>Frequency of observation.</description>
    <unit>Hz</unit>
    <dataType>real</dataType>
```

```
    </param>
  </interface>
```

3 Describing Standard Capabilities

This section describes the specific VOResource metadata extension schemas used to describe support for the target DAL protocols. The purpose of these schemas are to provide the *vr:Capability* sub-type that identifies the specific protocol. These are defined employing the recommendations for *vr:Capability* extensions given in the VOResource standard. In particular, each extension schema has the following features:

- The namespace associated with the extension is a URI that is intended to resolve an HTTP URL to XML Schema document that defines the extension schema. This means that VOResource document authors may use this URI as the location URL within the value of *xsi:schemaLocation* attribute. Note that the IVOA Registry Interface standard actually requires that the VOResource records it shares with other registries provide location URLs via *xsi:schemaLocation* for the VOResource schema and all legal extension schemas that are used in the records. This rule would apply to the extension schemas defined in this standard.

- A particular namespace prefix is recommended for use when referring to the specialized *vr:Capability* sub-type defined in the schema. In general XML applications, instance documents may use any prefix; however, in a VO context, document authors are strongly advised to use the canonical prefixes given (and used) in this document to avoid confusion when raw XML is exposed to the users. This means that documents should not use two different versions of a given schema (as defined by a common canonical prefix) within the same namespace mapping; documents for which this is impossible are probably semantically invalid.

- Following VOResource practice, the schema sets *elementFormDefault* to `"unqualified"`. This means that element names defined in the schema do not take a namespace prefix (as there are no global elements defined). The only place namespaced names occur in SimpleDALRegExt instance elements is the Capability sub-type name given as the value of an *xsi:type* attribute on the *capability* element (see the examples in the subsections below).

- The specialized *vr:Capability* sub-type includes a *testQuery* element for encoding parameters that together can be used to test the service.

The format for encoding the individual parameters is customized for each of the four simple services covered in this specification.

3.1 Simple Cone Search

This section describes the ConeSearch VOResource metadata extension schema which is used to describe services that comply with the Simple Cone Search protocol (Plante and Williams et al., 2008).

3.1.1 The Standard Identifier

The *standardID* value for Simple Cone Search version 1.03 (and before) is

`ivo://ivoa.net/std/ConeSearch` .

Standard identifiers for later versions will be given in the respective standards.

3.1.2 The Schema Namespace

The namespace associated with the ConeSearch extension schema is

`http://www.ivoa.net/xml/ConeSearch/v1.0` ,

the canoncical prefix is *cs:*.

3.1.3 ConeSearch

The *cs:ConeSearch* type is a *vr:Capability* sub-type that should be used to describe a service's support for the Simple Cone Search protocol; it is defined as follows:

cs:ConeSearch Type Schema Documentation

The capabilities of a Cone Search implementation.

cs:ConeSearch Type Schema Definition

```
<xs:complexType name="ConeSearch" >
  <xs:complexContent >
    <xs:extension base="vr:Capability" >
      <xs:sequence >
        <xs:element name="maxSR" type="xs:float" minOccurs="0" maxOccurs="1" />
        <xs:element name="maxRecords" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="verbosity" type="xs:boolean" />
        <xs:element name="testQuery" type="cs:Query" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

`cs:ConeSearch` Extension Metadata Elements

Element *maxSR*

Type floating-point number: *xs:float*

Meaning The largest search radius, in degrees, that will be accepted by the service without returning an error condition. Not providing this element or specifying a value of 180 indicates that there is no restriction.

Occurrence optional

Comment Not providing a value is the prefered way to indicate that there is no restriction.

Element *maxRecords*

Type *xs:positiveInteger*

Meaning The largest number of records that the service will return. Not providing this value means that there is no effective limit.

Occurrence optional

Comment This does not refer to the total number of records in the catalog but rather maximum number of records the service is capable of returning. A limit that is greater than the number of records available in the archive is equivalent to their being no effective limit. (See RM, Hanisch 2007.)

Element *verbosity*

Type boolean (true/false): xs:boolean

Meaning True if the service supports the VERB keyword; false, otherwise.

Occurrence required

Element *testQuery*

Type composite: *cs:Query*

Meaning A query that will result in at least on matched record that can be used to test the service.

Occurrence optional

The custom metadata that the *cs:ConeSearch* type provides is given above. For the elements whose semantics map directly to service profile metadata called for in the SCS standard, section 3, there is an entry labeled "SCS Name"; this indicates the metadata name given in the SCS specification that the element in this schema corresponds to. The profile metadata listed in the SCS specification that is not covered by the elements below are covered by other metadata that are part of the core VOResource schema.

3.1.4 testQuery and the Query Type

The *testQuery* element is intended to help other VO components (e.g. registries, validation services, services that monitor the VO's operational health, but typically not end users) test that the service is up and operating correctly. It provides a set of legal input parameters that should return a legal

response that includes at least one matched record. Since this query is intended for testing purposes, the size of the result set should be small.

The *cs:Query* type captures the different components of the query into separate elements, as defined below:

cs:Query Type Schema Documentation

A query to be sent to the service

cs:Query Type Schema Definition

```
<xs:complexType name="Query" >
  <xs:sequence >
    <xs:element name="ra" type="xs:double" />
    <xs:element name="dec" type="xs:double" />
    <xs:element name="sr" type="xs:double" />
    <xs:element name="verb" type="xs:positiveInteger" minOccurs="0" />
    <xs:element name="catalog" type="xs:string" minOccurs="0" />
    <xs:element name="extras" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

cs:Query Metadata Elements

Element *ra*

- *Type* floating-point number: *xs:double*
- *Meaning* the right ascension of the search cone's center in decimal degrees.
- *Occurrence* required

Element *dec*

- *Type* floating-point number: *xs:double*
- *Meaning* the declination of the search cone's center in decimal degrees.
- *Occurrence* required

Element *sr*

- *Type* floating-point number: *xs:double*
- *Meaning* the radius of the search cone in decimal degrees.
- *Occurrence* required

Element *verb*

- *Type* *xs:positiveInteger*
- *Meaning* the verbosity level to use where 1 means the bare minimum set of columns and 3 means the full set of available columns.
- *Occurrence* optional

Element *catalog*

- *Type* string: *xs:string*
- *Meaning* the catalog to query.
- *Occurrence* optional

Comment When the service can access more than one catalog, this input parameter, if available, is used to indicate which service to access.

Element *extras*

Type string: *xs:string*

Meaning any extra (non-standard) parameters that must be provided (apart from what is part of base URL given by the accessURL element).

Occurrence optional

Comment this value should be in the form of name=value pairs delimited with ampersands (&).

3.2 Simple Image Access

This section describes the SIA VOResource metadata extension schema which is used to describe services that comply with versions of the Simple Image Access protocol for which the specifications do not give extensions themselves. This applies at least to versions 1.0 (Harrison and Tody et al., 2009) and 2.0 (Dowler and Bonnarel et al., 2015)..

3.2.1 The Standard Identifier

The *standardID* value for the Simple Image Access protocol version 1.0 is

`ivo://ivoa.net/std/SIA` .

Standard identifiers for later versions are given in the respective standards; for instance, SIA version 2.0 (Dowler and Bonnarel et al., 2015), specifies

`ivo://ivoa.net/std/SIA#query-2.0`

for its query capability.

3.2.2 The Schema Namespace

The namespace associated with the SIA extension schema is

`http://www.ivoa.net/xml/SIA/v1.1` ,

the canonical namespace prefix is *sia:*

3.2.3 SimpleImageAccess

The *sia:SimpleImageAccess* type is a *vr:Capability* sub-type that should be used to describe a service's support for the Simple Image Access protocol; it is defined as follows:

sia:SimpleImageAccess Type Schema Documentation

The capabilities of an SIA implementation.

sia:SimpleImageAccess Type Schema Definition

```
<xs:complexType name="SimpleImageAccess" >
  <xs:complexContent >
    <xs:extension base="vr:Capability" >
      <xs:sequence >
        <xs:element name="imageServiceType" type="sia:ImageServiceType" />
        <xs:element name="maxQueryRegionSize" type="sia:SkySize" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="maxImageExtent" type="sia:SkySize" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="maxImageSize" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="maxFileSize" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="maxRecords" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="testQuery" type="sia:Query" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

sia:SimpleImageAccess Extension Metadata Elements

Element *imageServiceType*

Type string with controlled vocabulary

Meaning The class of image service: Cutout, Mosaic, Atlas, Pointed

Occurrence required

Allowed Values

Cutout

This is a service which extracts or "cuts out" rectangular regions of some larger image, returning an image of the requested size to the client. Such images are usually drawn from a database or a collection of survey images that cover some large portion of the sky. To be considered a cutout service, the returned image should closely approximate (or at least not exceed) the size of the requested region; however, a cutout service will not normally resample (rescale or reproject) the pixel data. A cutout service may mosaic image segments to cover a large region but is still considered a cutout service if it does not resample the data. Image cutout services are fast and avoid image degredation due to resampling.

Mosaic

This service is similar to the image cutout service but adds the capability to compute an image of the size, scale,

and projection specified by the client. Mosaic services include services which resample and reproject existing image data, as well as services which generate pixels from some more fundamental dataset, e.g., a high energy event list or a radio astronomy measurement set. Image mosaics can be expensive to generate for large regions but they make it easier for the client to overlay image data from different sources. Image mosaicing services which resample already pixelated data will degrade the data slightly, unlike the simpler cutout service which returns the data unchanged.

Atlas

This category of service provides access to pre-computed images that make up a survey of some large portion of the sky. The service, however, is not capable of dynamically cutting out requested regions, and the size of atlas images is predetermined by the survey. Atlas images may range in size from small cutouts of extended objects to large calibrated survey data frames.

Pointed

This category of service provides access to collections of images of many small, "pointed" regions of the sky. "Pointed" images normally focus on specific sources in the sky as opposed to being part of a sky survey. This type of service usually applies to instrumental archives from observatories with guest observer programs (e.g., the HST archive) and other general purpose image archives (e.g., the ADIL). If a service provides access to both survey and pointed images, then it should be considered a Pointed Image Archive for the purposes of this specification; if a differentiation between the types of data is desired the pointed and survey data collections should be registered as separate image services.

Element *maxQueryRegionSize*

Type composite: *sia:SkySize*

Meaning The maximum image query region size, expressed in decimal degrees. Not providing this element or specifying a value of 360 degrees indicates that there is no limit and the entire data collection (entire sky) can be queried.

Occurrence optional

Comment Not providing a value is the prefered way to indicate that there is no limit.

Element *maxImageExtent*

Type composite: *sia:SkySize*

Meaning An upper bound on a region of the sky that can be covered by returned images. That is, no image returned by this service will cover more than this limit. Not providing this element or specifying a value of 360 degrees indicates that there is no fundamental limit to the region covered by a returned image.

Occurrence optional

Comment When the imageServiceType is “Cutout” or “Mosaic”, this represents the largest area that can be requested. In this case, the “no limit” value means that all-sky images can be requested. When the type is “Atlas” or “Pointed”, it should be a region that most closely encloses largest images in the archive, and the ”no limit” value means that the archive contains all-sky (or nearly so) images.

Comment Not providing a value is the prefered way to indicate that there is no limit.

Element *maxImageSize*

Type *xs:positiveInteger*

Meaning A measure of the largest image the service can produce given as the maximum number of pixels along the first or second axes. Not providing a value indicates that there is no effective limit to the size of the images that can be returned.

Occurrence optional

Comment This is primarily relevant when the imageServiceType is “Cutout” or “Mosaic”, indicating the largest image that can be created. When the imageServiceType is “Atlas” or “Pointed”, this should be specified only when there are static images in the archive that can be searched for but not returned because they are too big.

Comment When a service is more fundementally limited by the total number of pixels in the image, this value should be set to the square-root of that number. This number will then represent a lower limit on the maximum length of a side.

Element *maxFileSize*

Type *xs:positiveInteger*

Meaning The maximum image file size in bytes. Not providing a value indicates that there is no effective limit the size of files that can be returned.

Occurrence optional

Comment This is primarily relevant when the imageServiceType is “Cutout” or “Mosaic”, indicating the largest files that can be created. When the imageServiceType is “Atlas” or “Pointed”, this should be specified only when there are static images in the archive that can be searched for but not returned because they are too big.

Element *maxRecords*

Type *xs:positiveInteger*

Meaning The largest number of records that the Image Query web method will return. Not providing this value means that there is no effective limit.

Occurrence optional

Comment This does not refer to the total number of images in the archive but rather maximum number of records the service is capable of returning. A limit that is greater than the number of

images available in the archive is equivalent to their being no effective limit. (See RM, Hanisch 2007.)

Element *testQuery*

Type composite: *sia:Query*

Meaning a set of query parameters that is expected to produce at least one matched record which can be used to test the service.

Occurrence optional

3.2.4 SkySize

The *sia:SkySize* type is used to capture simple rectangular extents on the sky along longitudinal and latitudinal directions. It is defined as follows:

sia:SkySize Type Schema Definition

```
<xs:complexType name="SkySize" >
  <xs:sequence >
    <xs:element name="long" type="xs:double" />
    <xs:element name="lat" type="xs:double" />
  </xs:sequence>
</xs:complexType>
```

sia:SkySize Metadata Elements

Element *long*

Type floating-point number: *xs:double*

Meaning The maximum size in the longitude (R.A.) direction given in degrees

Occurrence required

Element *lat*

Type floating-point number: *xs:double*

Meaning The maximum size in the latitude (Dec.) direction given in degrees

Occurrence required

3.2.5 testQuery and the Query Type

As with the other DAL *vr:capability* types, the *testQuery* element is intended to help other VO components (e.g. registries, validation services, services that monitor the VO's operational health–but typically not end users) test that the service is up and operating correctly. It provides a set of legal input parameters that should return a legal response that includes at least matched record. Since this query is intended for testing purposes, the size of the result set should be small.

The *sia:Query* type captures the different components of the query into separate elements, as defined below:

sia:Query Type Schema Documentation

A query to be sent to the service

sia:Query Type Schema Definition

```
<xs:complexType name="Query" >
  <xs:sequence >
    <xs:element name="pos" type="sia:SkyPos" minOccurs="0" />
    <xs:element name="size" type="sia:SkySize" minOccurs="0" />
    <xs:element name="verb" type="xs:positiveInteger" minOccurs="0" />
    <xs:element name="extras" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

sia:Query Metadata Elements

Element *pos*

Type composite: *sia:SkyPos*

Meaning the center position of the rectangular region that should be used as part of the query to the SIA service.

Occurrence optional

Element *size*

Type composite: *sia:SkySize*

Meaning the rectangular size of the region that should be used as part of the query to the SIA service.

Occurrence optional

Element *verb*

Type *xs:positiveInteger*

Meaning the verbosity level to use where 0 means the bare minimum set of columns and 3 means the full set of available columns.

Occurrence optional

Element *extras*

Type string: *xs:string*

Meaning any extra (particularly non-standard) parameters that must be provided (apart from what is part of base URL given by the accessURL element).

Occurrence optional

Comment this value should be in the form of name=value pairs delimited with ampersands (&).

3.2.6 SkyPos

The *sia:SkyPos* type is used to encode the *testQuery*'s *pos* element, the center position of the test region of interest.

sia:SkyPos Type Schema Definition

```
<xs:complexType name="SkyPos" >
  <xs:sequence >
    <xs:element name="long" type="xs:double" />
    <xs:element name="lat" type="xs:double" />
  </xs:sequence>
</xs:complexType>
```

sia:SkyPos Metadata Elements

Element *long*

Type	floating-point number: *xs:double*
Meaning	The sky position in the longitude (R.A.) direction
Occurrence	required

Element *lat*

Type	floating-point number: *xs:double*
Meaning	The sky position in the latitude (Dec.) direction
Occurrence	required

3.3 Simple Spectral Access

This section describes the SSA VOResource metadata extension schema which is used to describe services that comply with the Simple Spectral Access protocol, which primarily defines the *ssap:SimpleSpectralAccess* *vr:Capability* type to be used by services compliant with published SSA Recommendation (Tody and Dolensky et al., 2012).

3.3.1 The Standard Identifier

The *standardID* value for Simple Spectral access version 1.1 (and before) is

`ivo://ivoa.net/std/SSA` .

Standard identifiers for later versions will be given in the respective standards.

3.3.2 The Schema Namespace

The namespace associated with the SSA extension schema is `http://www.ivoa.net/xml/SSA/v1.1`. The namespace prefix, *ssap:*, should be used in applications where common use of prefixes improves interoperability (e.g. in the IVOA registries). Furthermore, we use the *ssap:* prefix in this document to refer to types defined as part of the SSA extension schema.

Note

Though it departs a bit from convention, the ssap prefix was chosen to avoid a collision with its use in SSA for identifying UTypes from the Spectral Data Model.

3.3.3 SimpleSpectralAccess

The *ssap:SimpleSpectralAccess* type is the *vr:Capability* sub-type that should be used to describe a service's support for the Simple Spectral Access protocol; it is defined as follows:

ssap:SimpleSpectralAccess Type Schema Documentation

The capabilities of an SSA service implementation.

ssap:SimpleSpectralAccess Type Schema Definition

```
<xs:complexType name="SimpleSpectralAccess" >
  <xs:complexContent >
    <xs:extension base="vr:Capability" >
      <xs:sequence >
        <xs:element name="complianceLevel" type="ssap:ComplianceLevel" />
        <xs:element name="dataSource" type="ssap:DataSource" minOccurs="1"
                maxOccurs="unbounded" />
        <xs:element name="creationType" type="ssap:CreationType" minOccurs="1"
                maxOccurs="unbounded" />
        <xs:element name="supportedFrame" type="ssap:SupportedFrame" minOccurs="1"
                maxOccurs="unbounded" />
        <xs:element name="maxSearchRadius" type="xs:double" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="maxRecords" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="defaultMaxRecords" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="maxAperture" type="xs:double" minOccurs="0" maxOccurs="1" />
        <xs:element name="maxFileSize" type="xs:positiveInteger" minOccurs="0"
                maxOccurs="1" />
        <xs:element name="testQuery" type="ssap:Query" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

ssap:SimpleSpectralAccess Extension Metadata Elements

Element *complianceLevel*

Type string with controlled vocabulary

Meaning The category indicating the level to which this instance complies with the SSA standard.

Occurrence required

Allowed Values

query
: The service supports all of the capabilities and features of the SSA protocol identified as "must" in the specification, except that it does not support returning data in at least one SSA-compliant format.

minimal
: The service supports all of the capabilities and features of the SSA protocol identified as "must" in the specification.

full
: The service supports all of the capabilities and features of the SSA protocol identified as "must" or "should" in the specification.

Comment Allowed values are "query", "minimal", and "full". See definitions of allowed values for details.

Element *dataSource*

Type string with controlled vocabulary

Meaning The category specifying where the data originally came from.

Occurrence required; multiple occurrences allowed.

Allowed Values

survey
: A survey dataset, which typically covers some region of observational parameter space in a uniform fashion, with as complete as possible coverage in the region of parameter space observed.

pointed
: A pointed observation of a particular astronomical object or field.

custom
: Data which has been custom processed, e.g., as part of a specific research project.

theory
: Theory data, or any data generated from a theoretical model, for example a synthetic spectrum.

artificial
: Artificial or simulated data.

Comment Allowed values are "survey", "pointed", "custom", "theory", "artificial"

Element *creationType*

Type string with controlled vocabulary

Meaning The category that describes the process used to produce the dataset.

Occurrence required; multiple occurrences allowed.

Allowed Values

archival
: The entire archival or project dataset is returned. Transformations such as metadata or data model mediation or

format conversions may take place, but the content of the dataset is not substantially modified (e.g., all the data is returned and the sample values are not modified).

cutout
: The dataset is subsetted in some region of parameter space to produce a subset dataset. Sample values are not modified, e.g., cutouts could be recombined to reconstitute the original dataset.

filtered
: The data is filtered in some fashion to exclude portions of the dataset, e.g., passing only data in selected regions along a measurement axis, or processing the data in a way which recomputes the sample values, e.g., due to interpolation or flux transformation.

mosaic
: Data from multiple non- or partially-overlapping datasets are combined to produce a new dataset.

projection
: Data is geometrically warped or dimensionally reduced by projecting through a multidimensional dataset.

spectralExtraction
: Extraction of a spectrum from another dataset, e.g., extraction of a spectrum from a spectral data cube through a simulated aperture.

catalogExtraction
: Extraction of a catalog of some form from another dataset, e.g., extraction of a source catalog from an image, or extraction of a line list catalog from a spectrum (not valid for a SSA service).

Comment Typically this describes only the processing performed by the data service, but it could describe some additional earlier processing as well, e.g., if data is partially precomputed.

Comment Allowed values are "archival", "cutout", "filtered", "mosaic", "projection", "spectralExtraction", "catalogExtraction"

Element *supportedFrame*

Type string with controlled vocabulary

Meaning The STC name for a world coordinate system frame supported by this service.

Occurrence required; multiple occurrences allowed.

Allowed Values

FK4
: the Fundemental Katalog, system 4, frame; Besselian

FK5
: the Fundemental Katalog, system 5, frame; Julien

ECLIPTIC
: Ecliptic coordinates

ICRS
: International Celestial Reference System

GALACTIC_I
: old Galactic coordinates

GALACTIC_II
: old Galactic coordinates

SUPER_GALACTIC
: Super-galactic coordinates with the north pole at GALACTIC_II (47.37, +6.32) and the origin at GALACTIC_II (137.37, 0).

AZ_EL
: The local azimuth and elevation frame where azimuth increases from north through east.

BODY
: A generic solar system body-centered coordinate frame

GEO_C
: 3D Geographic (geocentric) coordinates where the magnitude is expressed as a geocentric distance

GEO_D
: 3D Geographic (geocentric) coordinates where the magnitude is expressed as an elevation above sea-level.

MAG
: Geomagnetic coordinates.

GSE
: Geocentric Solar Ecliptic coordinates

GSM
: Geocentric Solar Magnetic coordinates

HGC
: Heliographic coordinates (Carrington)

HGS
: Heliographic coordinates (Stonyhurst)

HEEQ
: Heliographic Earth Equatorial coordinates

HRTN
: Heliographic Radial-Tangential-Normal coordinates

HPC
: Helioprojective Cartesian coordinates

HPR
: Helioprojective Polar coordinates

HCC
: Heliocentric Cartesian coordinates

HGI
: Heliographic Inertial coordinates

MERCURY_C
: Planteocentric coordinates on Mercury

VENUS_C
: Planteocentric coordinates on Venus

LUNA_C
: Selenocentric coordinates (for the Moon)

MARS_C
: Planteocentric coordinates on Mars

JUPITER_C_III
: Planteocentric coordinates on Jupiter, system III

SATURN_C_III
: Planteocentric coordinates on Saturn, system III

URANUS_C_III
: Planteocentric coordinates on Uranus, system III

NEPTUNE_C_III
: Planteocentric coordinates on Neptune, system III

PLUTO_C
: Planteocentric coordinates on Mercury

MERCURY_G
: Planteographic coordinates on Mercury

VENUS_G
: Planteographic coordinates on Venus

LUNA_G
: Selenographic coordinates (for the Moon)

MARS_G
: Planteographic coordinates on Mars

JUPITER_G_III
: Planteographic coordinates on Jupiter, system III

SATURN_G_III
: Planteographic coordinates on Saturn, system III

URANUS_G_III
: Planteographic coordinates on Uranus, system III

NEPTUNE_G_III
: Planteographic coordinates on Neptune, system III

PLUTO_G
: Planteographic coordinates on Mercury

UNKNOWN
: a frame that is either unknown or non-standard

Comment At least one recognized value must be listed. With SSA v1.1, ICRS must be supported; thus, this list must include at least this value.

Element *maxSearchRadius*

Type floating-point number: *xs:double*

Meaning The largest search radius, in degrees, that will be accepted by the service without returning an error condition. Not providing this element or specifying a value of 180 indicates that there is no restriction.

Occurrence optional

Comment Not providing a value is the prefered way to indicate that there is no restriction.

Element *maxRecords*

Type *xs:positiveInteger*

Meaning The hard limit on the largest number of records that the query operation will return in a single response. Not providing this value means that there is no effective limit.

Occurrence optional

Comment This does not refer to the total number of spectra in the archive but rather maximum number of records the service is capable of returning. A limit that is greater than the number of spectra available in the archive is equivalent to their being no effective limit. (See RM, Hanisch 2007.)

Element *defaultMaxRecords*

Type *xs:positiveInteger*

Meaning The largest number of records that the service will return when the MAXREC parameter not specified in the query input. Not providing a value means that the hard limit implied by maxRecords will be the default limit.

Occurrence optional

Element *maxAperture*

Type floating-point number: *xs:double*

Meaning The largest aperture that can be supported upon request via the APERTURE input parameter by a service that supports the spectral extraction creation method. A value of 180 or not providing a value means there is no theoretical limit.

Occurrence optional

Comment Not providing a value is the preferred way to indicate that there is no limit.

Element *maxFileSize*

Type *xs:positiveInteger*

Meaning The maximum spectrum file size in bytes that will be returned. Not providing a value indicates that there is no effective limit the size of files that can be returned.

Occurrence optional

Comment This is primarily relevant when spectra are created on the fly (see creationType). If the service provides access to static spectra, this should only be specified if there are spectra in the archive that can be searched for but not returned because they are too big.

Element *testQuery*

Type composite: *ssap:Query*

Meaning a set of query parameters that is expected to produce at least one matched record which can be used to test the service.

Occurrence optional

The custom metadata that the *ssap:SimpleSpectralAccess* type provides is given above. Note that some of these elements derive from the SSA standard; others, from the RM standard (Hanisch and IVOA Resource Registry Working Group et al., 2007). The "Semantic Meaning" entry provides the reference to the original definition.

3.3.4 testQuery and the Query Type

As with the other DAL *vr:capability* types, the *testQuery* element is intended to help other VO components (e.g. registries, validation services,

services that monitor the VO's operational health – but typically not end users) test that the service is up and operating correctly. It provides a set of legal input parameters that should return a legal response that includes at least matched record. Since this query is intended for testing purposes, the size of the result set should be small.

The *ssap:Query* type captures the different components of the query into separate elements, as defined below:

ssap:Query Type Schema Documentation

A query to be sent to the service

ssap:Query Type Schema Definition

```
<xs:complexType name="Query" >
  <xs:sequence >
    <xs:element name="pos" type="ssap:PosParam" minOccurs="0" />
    <xs:element name="size" type="xs:double" minOccurs="0" />
    <xs:element name="queryDataCmd" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

ssap:Query Metadata Elements

Element *pos*
- *Type* composite: *ssap:PosParam*
- *Meaning* the center position the search cone given in decimal degrees.
- *Occurrence* optional

Element *size*
- *Type* floating-point number: *xs:double*
- *Meaning* the size of the search radius.
- *Occurrence* optional

Element *queryDataCmd*
- *Type* string: *xs:string*
- *Meaning* Fully specified test query formatted as an URL argument list in the syntax specified by the SSA standard. The list must exclude the REQUEST argument which is assumed to be set to "queryData".
- *Occurrence* optional
- *Comment* This value must be in the form of name=value pairs delimited with ampersands (&). A query may then be formed by appending to the base URL the request argument, "REQUEST=queryData&", followed by the contents of this element.

3.3.5 PosParam

The *ssap:PosParam* type is used to encode the *testQuery*'s *pos* element, the center position of the test region of interest; it is defined as follows:

ssap:PosParam Type Schema Documentation

a position in the sky to search.

ssap:PosParam Type Schema Definition

```
<xs:complexType name="PosParam" >
  <xs:sequence >
    <xs:element name="long" type="xs:double" />
    <xs:element name="lat" type="xs:double" />
    <xs:element name="refframe" type="xs:token" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

ssap:PosParam Metadata Elements

Element *long*
- *Type* floating-point number: ***xs:double***
- *Meaning* The longitude (e.g. Right Ascension) of the center of the search position in decimal degrees.
- *Occurrence* required

Element *lat*
- *Type* floating-point number: ***xs:double***
- *Meaning* The latitude (e.g. Declination) of the center of the search position in decimal degrees.
- *Occurrence* required

Element *refframe*
- *Type* string: ***xs:token***
- *Meaning* the coordinate system reference frame name indicating the frame to assume for the given position. If not provided, ICRS is assumed.
- *Occurrence* optional

3.3.6 ProtoSpectralAccess

The *ssap:ProtoSpectralAccess* type still defined in the schema was inteded for seamless migration of services predating the SSAP specification. It should no longer be used.

3.4 Simple Line Access

This section describes the SLA VOResource metadata extension schema which is used to describe services that comply with the Simple Line Access protocol (Osuna and Salgado et al., 2010).

3.4.1 The Standard Identifier

The *standardID* value for Simple Line Access version 1.0 is

`ivo://ivoa.net/std/SLAP` .

Standard identifiers for later versions will be given in the respective standards.

3.4.2 The Schema Namespace

The namespace associated with the SLA extension schema is `http://www.ivoa.net/xml/SLAP/v1.0`. The namespace prefix, *slap:*, should be used in applications where common use of prefixes improves interoperability (e.g. in the IVOA registries). Furthermore, we use the *slap:* prefix in this document to refer to types defined as part of the SLA extension schema.

3.4.3 SimpleLineAccess

The *slap:SimpleLineAccess* type is a *vr:Capability* sub-type that should be used to describe a service's support for the Simple Line Access protocol; it is defined as follows:

slap:SimpleLineAccess Type Schema Documentation

The capabilities of an SLAP service implementation.

slap:SimpleLineAccess Type Schema Definition

```
<xs:complexType name="SimpleLineAccess" >
  <xs:complexContent >
    <xs:extension base="vr:Capability" >
      <xs:sequence >
        <xs:element name="complianceLevel" type="slap:ComplianceLevel" />
        <xs:element name="dataSource" type="slap:DataSource" />
        <xs:element name="maxRecords" type="xs:positiveInteger" minOccurs="0"
                 maxOccurs="1" />
        <xs:element name="testQuery" type="slap:Query" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

slap:SimpleLineAccess Extension Metadata Elements

Element *complianceLevel*

Type string with controlled vocabulary

Meaning The category indicating the level to which this service instance complies with the SLAP standard.

Occurrence required

Allowed Values

minimal

The service supports all of the capabilities and features of the SLAP protocol identified as "must" in the specification.

full
: The service supports, at a minimum, all of the capabilities and features of the SLAP protocol identified as "must" or "should" in the specification.

Comment Allowed values are "minimal" and "full". See definitions of allowed values for details.

Element *dataSource*

Type string with controlled vocabulary

Meaning The category specifying where the data accessed by the service originally came from.

Occurrence required

Allowed Values

observational/astrophysical
: Lines observed and identified in real spectra of astrophysical observations by different instrument/projects

observational/laboratory
: Lines observed and identified in real spectra of laboratory measurements

theoretical
: Servers containing theoretical spectral lines

Comment Allowed values are "observational/astrophysical", "observational/laboratory", "theoretical"

Element *maxRecords*

Type *xs:positiveInteger*

Meaning The hard limit on the largest number of records that the query operation will return in a single response. Not providing this value means that there is no effective limit.

Occurrence optional

Comment This does not refer to the total number of spectra in the archive but rather maximum number of records the service is capable of returning. A limit that is greater than the number of spectra available in the archive is equivalent to their being no effective limit. (See RM, Hanisch 2007.)

Element *testQuery*

Type composite: *slap:Query*

Meaning A set of queryData parameters that is expected to produce at least one matched record which can be used to test the service.

Occurrence optional

Comment The value should include all parameters required for the test query but should exclude the baseURL and the REQUEST parameter.

3.4.4 testQuery and the Query Type

As with the other DAL *vr:capability* types, the *testQuery* element is intended to help other VO components (e.g. registries, validation services, services that monitor the VO's operational health – but typically not end users) test that the service is up and operating correctly. It provides a set of

legal input parameters that should return a legal response that includes at least matched record. Since this query is intended for testing purposes, the size of the result set should be small.

The *slap:Query* type captures the different components of the query into separate elements, as defined below:

slap:Query Type Schema Documentation

A query to be sent to the service, e.g., a test query.

slap:Query Type Schema Definition

```
<xs:complexType name="Query" >
  <xs:sequence >
    <xs:element name="wavelength" type="slap:WavelengthRange" minOccurs="0" />
    <xs:element name="queryDataCmd" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

slap:Query Metadata Elements

Element *wavelength*

Type composite: *slap:WavelengthRange*

Meaning Spectral range in meters to be used to constrain the query of spectral lines.

Occurrence optional

Element *queryDataCmd*

Type string: *xs:string*

Meaning Fully specified queryData test query formatted as an URL argument list in the syntax specified by the SLAP standard. The list must exclude the REQUEST argument which is assumed to be set to "queryData". VERSION may be included if the test query applies to a specific version of the service protocol.

Occurrence optional

Comment If queryDataCmd is used to form a query, the default value of WAVELENGTH specified above is not used; if the test query requires WAVELENGTH it should be included directly in queryDataCmd.

Comment This value must be a string in the form of name=value pairs delimited with ampersands (&). A query may then be formed by appending to the baseURL the request argument, "REQUEST=queryData&", followed by the contents of this element.

3.4.5 WavelengthRange

The *slap:WavelengthRange* type is used to encode the *testQuery*'s *wavelength* element, the range of wavelengths to search.

slap:WavelengthRange Type Schema Documentation

Spectral range in meters to be used to constrain the query of spectral lines

slap:WavelengthRange Type Schema Definition

```
<xs:complexType name="WavelengthRange" >
  <xs:sequence >
    <xs:element name="minWavelength" type="xs:double" minOccurs="0" />
    <xs:element name="maxWavelength" type="xs:double" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

slap:WavelengthRange Metadata Elements

Element *minWavelength*

Type floating-point number: *xs:double*

Meaning Minimum wavelength in meters to be used to constrain the query of spectral lines

Occurrence optional

Element *maxWavelength*

Type floating-point number: *xs:double*

Meaning Maximum wavelength in meters to be used to constrain the query of spectral lines

Occurrence optional

A Supporting Multiple Versions of DAL Protocols

This section is non-normative.

It is possible for a VOResource-encoded resource description to indicate support for multiple versions of standard service. This is described generally in Section 2.2.2 ("The service data model") of the VOResource specification (Plante and Benson et al., 2008). In that section, the specification says that a *capability* element can contain multiple *interface* elements, each describing a different version.

In VO practice, in particular after the publication of StandardsRegExt (Harrison and Burke et al., 2012), it turned out that versioning of IVOA standards now (typically) occurs on the capability level rather the interface level. In consequence, future standards use a different syntax for *standardID*.

We start by noting that the *standardID* values for each of the DAL protocols described in this document refer to the standards generally, without reference to the particular version. For example, the IVOA identifier for the Simple Cone Search protocol is `ivo://ivoa.net/std/ConeSearch`. Thus a *capability* element can logically describe support for any version or multiple versions of the standard DAL protocol as long as the extension schema for that protocol is same for all of the versions.

Here is an example a service that supports both SIA versions 1.0 and 2.0, as well as a web browser interface on the 1.0 endpoint:

```
<vr:Resource xsi:type="vs:CatalogService">
  <title>Example Image Service</title>
   [...]
  <capability standardID="ivo://ivoa.net/std/SIA">
    <!-- this describes a SIA version 1 "face" of the service -->
    <interface role="std" xsi:type="vs:ParamHTTP">
      <!-- this is the SIA version 1.0 endpoint, the one standard
        clients talk to-->
      <accessURL use="base">http://example.com/asvc/sia.xml?</accessURL>
      <queryType>GET</queryType>
      <resultType>application/x-votable+xml</resultType>
      <param std="true">
        <name>POS</name>
        <description>ICRS Position, RA,DEC decimal degrees</description>
         [... enumerate the parameters supported ...]
      </param>
    </interface>

    <interface xsi:type="vr:WebBrowser">
      <!-- this a a very SIA-like interface renderable in a web browser.
        If the web interface is functionally fairly different in
        interaction from a SIA version 1, put this into a separate,
        untyped capability -->
      <accessURL use="full">http://example.com/asvc/form.html</accessURL>
    </interface>

    <imageServiceType>Pointed</imageServiceType>
    <maxRecords>1000000</maxRecords>
    <testQuery>
      <pos>
        <long>230.444</long>
        <lat>52.929</lat>
      </pos>
      <size>
        <long>0.1</long>
        <lat>0.1</lat>
      </size>
    </testQuery>
  </capability>

  <capability standardID="ivo://ivoa.net/std/SIA#query-2.0">
    <!-- this describes a SIA version 2 "face" of the service -->
    <interface role="std" xsi:type="vs:ParamHTTP">
      <accessURL use="base">http://example.com/asvc/sia2.xml?</accessURL>
      <queryType>GET</queryType>
      <resultType>application/x-votable+xml</resultType>
      <param std="true">
        <name>POS</name>
        <description>Specification of a region of ... </description>
         [... enumerate the parameters supported for SIAv2...]
      </param>
```

```
    </interface>

    <imageServiceType>Pointed</imageServiceType>
    <maxRecords>10000</maxRecords>
    <testQuery>
      <pos>
        <long>230.444</long>
        <lat>52.929</lat>
      </pos>
      <size>
        <long>0.1</long>
        <lat>0.1</lat>
      </size>
    </testQuery>
  </capability>
</vr:Resource>
```

B Change History

B.1 Changes from PR-2016-07-06

- References to auxiliary capabilities removed again.

B.2 Changes from REC-1.0

- *standardID* values are no longer fixed for the various capability types.
- Now giving the *standardID* values of the existing standards in the text (since they are no longer in the schema).
- XML schemas are no longer included in the document; the files in the IVOA repository are declared authoritative.
- We now claim, essentially, to describe the S-protocol metadata schemas until the respective standards define one themselves.
- Updated example in the appendix to the style of Identifiers 2.0
- Mentioning auxiliary capabilities and giving a standard id for them.
- Removing most material on ProtoSpectralAccess.

B.3 Changes since PR-v1.0 20130911

- none other than date and status.

B.4 Changes from PR-v1.0 20121116

- for SSA's creationType, changed specialExtraction to spectralExtraction.
- corrected Creation Type reference to section in SSA doc.
- made long and lat elements in ssap:PosParam required.
- incremented SSA schema version to 1.1 in namespace.
- refresh App. A from official schemas
- fixed typos ("IRCS" and value type for maxFileSize)
- noted that the <long> and <lat> values within the sia:SkySize type are given in degrees.
- Fixed documentation of SIA's sia:Query type in the schema.

B.5 Changes from PR-v1.0 20120517

- The namespace URIs given in Sections 3.1.1, 3.2.1, 3.3.1, and 3.4.1 were updated to match that specified in the XSDs (i.e. to include a "v" preceding the version field).
- Several capability metadata with types xs:int and xs:float were changed to xs:positiveInterger xs:double to allow for larger/more precise numbers.
- Capability metadata that indicated maximum allowed values (e.g. <maxRecords>, <maxImageSize>, etc.) were made optional to avoid large, meaningless numbers from being provided. Now not specifying a value is the preferred way to indicate that no upper limit applies.
- Semantic definition of <sia:maxImageExtent> clarified to differentiate it from <sia:maxQueryRegionSize>
- The type for <sia:maxImageSize> was changed to xs:positiveInteger, a single number that represents the length of a side in pixels. The sia:ImageSize type (no longer needed) was dropped.
- The version field in the SIA namespace was incremented to 1.1 due to the non-backward-compatible change to <sia:maxImageSize>
- various typos and grammatical errors corrected.

B.6 Changes from WD-v1.0 20110921

- Now recommend ssap as prefix; changed all occurances of ssa in text and schema.
- added <supportedFrame> to ssap:SimpleSpectralAccess
- removed import of VODataService schema from SIA, SSA, and Conesearch schemas.
- change base type of controlled vocab types from xs:string to xs:token for consistancy with VOResource.

References

Arviset, C., Gaudet, S. and the IVOA Technical Coordination Group (2010), 'IVOA architecture', IVOA Note.
URL: *http://www.ivoa.net/documents/Notes/IVOAArchitecture*

Benson, K., Plante, R., Auden, E., Graham, M., Greene, G., Hill, M., Linde, T., Morris, D., O'Mullane, W., Rixon, G., Stébé, A. and Andrews, K. (2009), 'IVOA Registry Interfaces Version 1.0', IVOA Recommendation 04 November 2009, arXiv:1110.0513.
URL: *http://adsabs.harvard.edu/abs/2009ivoa.spec.1104B*

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

Dowler, P., Bonnarel, F. and Tody, D. (2015), 'IVOA Simple Image Access Version 2.0', IVOA Recommendation 23 December 2015.
URL: *http://adsabs.harvard.edu/abs/2015ivoa.spec.1223D*

Graham, M., Rixon, G. and Grid andWeb Services Working Group (2011), 'IVOA Support Interfaces Version 1.0', IVOA Recommendation 31 May 2011, arXiv:1110.5825.
URL: *http://adsabs.harvard.edu/abs/2011ivoa.spec.0531G*

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.
URL: *http://adsabs.harvard.edu/abs/2007ivoa.spec.0302H*

Harrison, P., Burke, D., Plante, R., Rixon, G., Morris, D. and IVOA Registry Working Group (2012), 'StandardsRegExt: a VOResource Schema

Extension for Describing IVOA Standards Version 1.0', IVOA Recommendation 08 May 2012, arXiv:1402.4745.
URL: *http://adsabs.harvard.edu/abs/2012ivoa.spec.0508H*

Harrison, P., Demleitner, M., Major, B. and Dowler, P. (2016), 'XML schema versioning policies', IVOA Note.
URL: *http://ivoa.net/documents/Notes/XMLVers*

Harrison, P., Tody, D. and Plante, R. (2009), 'Simple Image Access Specification Version 1.0', IVOA Recommendation 11 November 2009, arXiv:1110.0499.
URL: *http://adsabs.harvard.edu/abs/2009ivoa.spec.1111H*

Osuna, P., Salgado, J., Guainazzi, M., Barbarisi, I., Dubernet, M.-L. and Tody, D. (2010), 'Simple Line Access Protocol Version 1.0', IVOA Recommendation 9 December 2010, arXiv:1110.0500.
URL: *http://adsabs.harvard.edu/abs/2010ivoa.specQ1209O*

Plante, R., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T., Rixon, G., Stébé, A. and IVOA Registry Working Group (2008), 'VOResource: an XML Encoding Schema for Resource Metadata Version 1.03', IVOA Recommendation 22 February 2008, arXiv:1110.0515.
URL: *http://adsabs.harvard.edu/abs/2008ivoa.spec.0222P*

Plante, R., Stébé, A., Benson, K., Dowler, P., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T. and Rixon, G. (2010), 'VODataService: a VOResource Schema Extension for Describing Collections, Services Version 1.1', IVOA Recommendation 02 December 2010, arXiv:1110.0516.
URL: *http://adsabs.harvard.edu/abs/2010ivoa.spec.1202P*

Plante, R., Williams, R., Hanisch, R. and Szalay, A. (2008), 'Simple Cone Search Version 1.03', IVOA Recommendation 22 February 2008, arXiv:1110.0498.
URL: *http://adsabs.harvard.edu/abs/2008ivoa.specQ0222P*

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

Tody, D., Dolensky, M., McDowell, J., Bonnarel, F., Budavari, T., Busko, I., Micol, A., Osuna, P., Salgado, J., Skoda, P., Thompson, R., Valdes, F. and Data Access Layer Working Group (2012), 'Simple Spectral Access Protocol Version 1.1', IVOA Recommendation 10 February 2012, arXiv:1203.5725.
URL: *http://adsabs.harvard.edu/abs/2012ivoa.spec.0210T*