



*International  
Virtual  
Observatory  
Alliance*

## Simple Spectral Access Protocol

### Version 0.91

*IVOA Working Draft Sep 20, 2005*

**This version:**

<http://www.ivoa.net/Documents/WD/SSA/WD-SSA-yyyymmdd.html>

**Latest version:**

<http://www.ivoa.net/Documents/WD/Identifiers/WD-SSA.html>

**Previous version(s):**

Version 0.90 May 2005

**Editors:**

D.Tody, M. Dolensky

**Authors**

M.Dolensky, D.Tody, T.Budavari, I.Busko, J.McDowell, P.Osuna, F.Valdes

### Abstract

The purpose of the Simple Spectral Access (SSA) protocol is to define a uniform interface to remotely access regularly or irregularly sampled, tabular spectrophotometric data, including 1D spectra, time series, and spectral energy distributions (SEDs).

The term *simple* in Simple Spectral Access refers to the design goal of defining a simple, uniform interface for retrieving spectrophotometric data from distributed data collections. It does not necessarily mean that implementing an SSA service is trivial, but the effort to make a data collection SSA compliant has been

minimized. The Data Access Layer (DAL) working group supports the implementation of reference code for services which can serve as a starting point for data providers when implementing a new service.

The SSA interface is similar to that of Simple Image Access (SIA) and Simple Cone Search (SCS). In particular the query/response protocol and data access method follow the same approach:

1. A *Query* method returns a table (VOTable format) describing candidate datasets which can be retrieved, including metadata describing each dataset, and an access reference which can be used to retrieve the data.
2. A *getData* method is used to access an individual dataset. The accessed data may be generated on-the-fly by the service at access time, e.g., to reformat or subset the data.
3. A *metadata query method* is defined by which an external client (such as the Registry) can determine the capabilities of a given SSA service instance.

SSA includes both a service protocol, and a spectral data model describing all the data types dealt with by SSA. External data is actively mediated at access time to return data to a client application which is conformant to the SSA data model.

## Status of This Document

This is a Working Draft. The first release of this document was in April 2005.

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

Comments on this document can be posted to the mailing list [dal@ivoa.net](mailto:dal@ivoa.net), uploaded to the collaborative web page [ivoaDAL](http://ivoaDAL), or sent to the authors directly.

It is expected that the Simple Image Access, Simple Cone Search and Simple Spectrum Access specifications will be homogenized prior to promotion to the proposed recommendation level. Additional changes are planned when support for SOAP and the Astronomy Data Query Language (ADQL) is integrated. Other changes such as a generalized metadata extension mechanism (*Proposal for an evolution of the SIA protocol*, Bonnarel et al. 2004) are also planned.

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

## Acknowledgements

This document has been developed with support from the 5th Framework Programme of the European Community for research, technological development and demonstration activities, contract HPRI-CT-2001-50030 and via a grant from the [National Science Foundation's](#) Information Technology Research program to develop the U.S. National Virtual Observatory.

Many of the ideas in this document originated from others involved in developing Virtual Observatory concepts and standards. In particular, the idea of using a logical name to group similar datasets was originally proposed by Roy Williams. Arnold Rots originated the idea of ranking query results via a *score* heuristic, and helped put the coordinate systems used in SSA on firm theoretical foundation via the development of STC. The technique for using dimensional analysis to consistently scale the physical axes of a spectrum came from Pedro Osuna, Jesus Salgado, and others at ESAC. Francois Bonnarel, Mireille Louys, Alberto Micol, and others made notable contributions to the representation of astronomical metadata. Laszlo Dobos contributed early implementations of the access protocol using the spectral archive at JHU.

Many thanks to all who contributed to the DAL survey among spectral data providers and consumers (Dolensky/Tody 2004): Ivo Busko, Mike Fitzpatrick, Satoshi Honda, Stephen Kent, Tom McGlynn, Pedro Osuna Alcalaya, Benoît Pirenne, Raymond Plante, Phillipe Prugniel, Enrique Solano, Alex Szalay, Francisco Valdes and Andreas Wicenec.

## Contents

|       |                                    |    |
|-------|------------------------------------|----|
| 1     | Overview                           | 4  |
| 1.1   | Concepts                           | 4  |
| 1.1.1 | Interface                          | 4  |
| 1.1.2 | Types of Data                      | 5  |
| 1.1.3 | Data Model                         | 7  |
| 1.1.4 | Data Representation                | 7  |
| 1.2   | Basic Usage                        | 8  |
| 1.2.1 | Query, Query Response              | 8  |
| 1.2.2 | Access Individual Datasets via URL | 8  |
| 1.2.3 | Data Formats                       | 8  |
| 2     | Requirements for Compliance        | 8  |
| 2.1   | Minimal Service                    | 8  |
| 2.2   | Terminology                        | 9  |
| 2.3   | Restrictions                       | 9  |
| 3     | Query Interface                    | 9  |
| 3.1   | Methods & Protocols                | 9  |
| 3.2   | Query                              | 10 |
| 3.2.1 | Mandatory Query Parameters         | 10 |

|       |   |    |
|-------|---|----|
| 3.2.2 | Recommended Query Parameters                                      | 12 |
| 3.2.3 | Optional Query Parameters   | 14 |
| 3.2.4 | Service-Defined Parameters  | 16 |
| 3.3   | Query Response  | 16 |
| 3.3.1 | Query Metadata  | 17 |
| 3.3.2 | Dataset Metadata  | 18 |
| 3.3.3 | Target Metadata   | 19 |
| 3.3.4 | Curation Metadata   | 20 |
| 3.3.5 | Characterization Metadata   | 20 |
| 3.3.6 | Instrument Metadata   | 23 |
| 3.3.7 | Access Metadata   | 24 |
| 3.3.8 | Additional Service-Defined Metadata                               | 24 |
| 3.4   | Error Response and Other Unsuccessful Results                     | 25 |
| 3.4.1 | Error   | 25 |
| 3.4.2 | Overflow  | 25 |
| 4     | Data Access   | 26 |
| 4.1   | Access Reference URL  | 26 |
| 4.2   | Data Format   | 26 |
| 4.3   | Data Compression  | 26 |
| 4.4   | Error Response  | 27 |
| 5     | Service Metadata  | 27 |
| 5.1   | Metadata Query  | 27 |
| 5.2   | Metadata Response   | 28 |
| 5.3   | Service Registration  | 28 |
| 5.3.1 | Overflow Conditions   | 28 |
| 5.3.2 | Service Type  | 28 |
|       | Appendix A: Sample Query Response for <i>Mixed</i> Service        | 29 |
|       | Appendix B: Query Response for <i>Spectrum/TimeSeries</i> Service | 30 |
|       | References  | 32 |

## 1 Overview

The Simple Spectral Access (SSA) protocol defines a uniform interface to remotely access regularly and irregularly sampled, tabular spectrophotometric data, including 1D spectra, time series, and SEDs. It shares many similarities with the Simple Image Access (SIA) protocol (Tody/Plante 2004) and the Simple Cone Search (SCS) protocol for catalog access.

### 1.1 Concepts

#### 1.1.1 Interface

All the VO data access interfaces have the same basic interface, differing mainly in the type of data being accessed. A query is used for data discovery, and to negotiate with the service the details of the datasets to be retrieved. Subsequent

data access requests can then be made to retrieve individual datasets of interest. While the DAL services can be used to retrieve static archive files (which are typically *external* data which does not conform to the SSA data model), more commonly data is generated on the fly at access time to optimize the data access for the data analysis being performed by the client. Active data model mediation may take place at access time to render external data into a standard form.

The basic data access pattern is as follows:

1. A *Query* method returns a table (VOTable format) describing candidate datasets which can be retrieved, including metadata describing each dataset, and an access reference which can be used to retrieve the data.
2. A number of *getData* methods may then be used to access individual datasets. The accessed data may be generated on-the-fly by the service at access time, e.g., to reformat or subset the data.

In addition, a *metadata query* method is defined by which an external client (such as the Registry) can determine the capabilities of a given SSA service. This information is most commonly used by clients before invoking a service, to select the services most appropriate for use by the client application.

### 1.1.2 Types of Data

In general data access is object-oriented, with a separate interface for each class of data: image, spectrum, catalog, and so forth. The service interface, including data model, query parameters, metadata, and service methods, is tailored for the type of data being accessed. By the time we get to an individual service the class of data supported has already been determined. SSA deals with tabular spectrophotometric data including spectra, time series, and SEDs.

For a given type of data we can break things down further to say something about how the dataset returned to the client was *computed*:

- **Atlas** data consists of pre-existing archival datasets from uniformly observed and processed data collections, such as surveys. The entire dataset is returned. Data access does not involve any changes to the data samples, although data model mediation and reformatting may occur.
- **Pointed** data also involves pre-existing datasets which are returned in their entirety. The data is again from a specific collection, such as an instrument, but may not be uniformly observed and processed. Sky coverage tends to be nonuniform but may provide high quality data for astronomical objects of interest. Datasets may vary greatly in size.

- **Cutout** data is data which is “cut out” of large archival datasets based on the query region requested by the client. The underlying collection may be either atlas or pointed. The subset may be made on any physical axis, i.e., spatial (sky), time, or spectral. “Cut out” means that the data may be subsetting, but the data samples are not modified. An actual dataset may be constructed by combining samples from multiple physical datasets, so long as the samples are not modified. A cutout service can greatly enhance performance by returning only the data of interest to the client.
- **Resampled** data is the same as cutout data except that the data samples are computed at access time, e.g., by interpolating or reprojecting the data, by level matching, flux correction following scale changes, computation of pixels from event or visibility data, and so forth. In all these cases the *service* is changing the data values from what is in the underlying published data collection. A resampling service can be very powerful, returning data most closely matching the ideal data requested by the client, but resampling may not be desired for certain types of analysis.

In the case of SSA, a survey consisting of numerous 4096-point spectra would probably be most sensibly served up via an atlas service. A collection of ultra-high resolution spectra, or a collection of lengthy time series with tens of thousands of sample points, might be best served via a cutout service. A cutout service could be very useful, for example, for retrieval of individual line profiles from high resolution spectra. If a collection is made available via a cutout or resampling service, consider make it available via an atlas or pointed service as well, for services which prefer to manipulate the archival data directly.

Finally we can say something about the *derivation* of the data:

- **Observed** data is data resulting from an actual physical observation. The data samples were physically observed.
- **Composite** data is the result of combining data from multiple observations. A given data sample may be derived from multiple observed samples, for example merged or accumulated spectra.
- **Dynamic** data results from a dynamically simulated observation based upon actual observational data in some form, for example an aperture spectrum derived from a spectral image cube, or an image derived from event or visibility data. Real observational data is used, but the instrument simulated in software.
- **Synthetic** data is a dataset computed from some theoretical model. Synthetic data may or may not be generated dynamically, but it is based on a physical model rather than actual observational data.

In the case of SSA, a typical observed dataset would be a processed, calibrated 1D spectrum or time series. An example of composite data would be a spectrum produced by transforming several observed spectra onto a common spectral scale and combining the result to produce a single spectrum.

### 1.1.3 Data Model

SSA consists of both an access protocol and interface, and an underlying *data model* describing the data to be accessed. The term data model as used here refers to a logical model for the data detailing the decomposition of a complex dataset into simpler elements, defining the relationships between elements, the meaning of each element, the concepts upon which the model is based, the metadata used to describe the data elements and dataset, and so forth. In this document we refer to the underlying data model interchangeably as the SSA data model or the *spectral data model*. The name spectral data model is not entirely appropriate since SSA also deals with time series data, but is chosen for brevity. The data model used in SSA is described in McDowell *et. al.*, 2004.

Explicitly defining the data model assumed by a data object is important for a variety of reasons. Doing so helps greatly to document the structure and meaning of the data. Data analysis software has to understand data at a fundamental level in order to function correctly. Data model *mediation* – the process of transforming data from some externally-defined data model to a known data model (the SSA data model in our case) – makes it possible for a client application to deal uniformly with any data for which mediation to a standard model is available. SSA does data model mediation on the fly, at data access time, in the service used to publish a data collection to the VO. The service has full knowledge of the data it is dealing with, making it relatively straightforward for the data provider to carry out the mediation. Any number of client applications can then access the data without requiring detailed knowledge of the peculiarities of each specific data collection. If more detailed knowledge of a specific data collection is required, various pass-through techniques can still be used to gain access to such lower level information.

### 1.1.4 Data Representation

A data model defines the logical content of data, but says nothing about how the data is represented externally, e.g., in a file. In principle, the same data object may be represented externally in many different ways (e.g., FITS, VOTable, and so forth). So long as the data model does not change, and the data representation is expressive enough, data may be transformed from one representation to another without loss of information. If the data model itself is changed, some loss of information may occur in the process of translation. This can happen, for example, during mediation of external data to a known data model by an SSA service.

## 1.2 Basic Usage

Although SSA has a sizeable interface, basic usage can still be quite simple. A minimal query might be a query for 1D spectra by position on the sky – the classic cone search. The service or services to be queried would normally be selected in advance via a registry query.

### 1.2.1 Query, Query Response

In a simple case of a positional query the query URL is very similar to that for SIA or SCS. For example,

**Example:**

```
http://www.myorg.org/ssa?POS=22.438,-17.2&SIZE=0.02
```

The query response is a VOTable formatted as defined later in this document.

### 1.2.2 Access Individual Datasets via URL

Simple dataset retrieval is then a matter of examining the query response, selecting the dataset or datasets to be retrieved, and retrieving them by reading the document pointed to by the *access reference* (a URL) for the dataset.

### 1.2.3 Data Formats

For a fully compliant SSA service the data returned by the service will be in one of the SSA supported data formats, conformant to the SSA data model. The simplest possible case would be a text file in CSV format.

## 2 Requirements for Compliance

The keywords “MUST”, “REQUIRED”, “SHOULD”, and “MAY” as used in this document are to be interpreted as described in the W3C specifications (RFC 2119).

### 2.1 Minimal Service

In order to be minimally SSA compliant a service must implement all elements of the SSA protocol identified as “MUST” in this document. In summary the minimal service includes the following:

1. The SSA query method must implement the HTTP GET interface, returning the query response encoded as a VOTable document. At least the `POS`, `SIZE`, and `FORMAT` query parameters must be supported. The query response must include at least all fields identified as *MUST provide* in the protocol.
2. A *getData* method must be provided capable of returning data in at least one of the SSA-compliant data formats (VOTable is preferred if only one format is supported).

3. A metadata query method must be provided which returns the capabilities of the SSA service encoded as defined herein.

If a service cannot return data which is SSA-compliant, it is still useful to implement a service which has an SSA-compliant query but which returns external data. The service is said to be *query compliant*.

A service is said to be *fully compliant* if, in addition to the functionality required to be minimally compliant, the service implements all the “SHOULD” provide elements of the interface.

A top of the line service will be fully compliant, will implement some of the “MAY” provide elements of the interface, for example supporting additional query parameters or returning additional metadata, or providing access to external data as well as SSA-compliant data, and will be capable of returning data in any supported standard data format requested by the client.

## 2.2 Terminology

## 2.3 Restrictions

The following assumptions and restrictions apply:

- Unless otherwise indicated it is assumed that data are fully calibrated. SSA can deal with uncalibrated or partially calibrated data, but by default calibrated data is assumed.
- SSA is a machine interface and therefore does not include features, such as unit conversions, which are more appropriate for a user interface.

## 3 Query Interface

The purpose of the SSA query is to determine the availability and characterization of data satisfying the given search constraints. The result is returned encoded as a VOTable document.

### 3.1 Methods & Protocols

As with all the DAL interfaces, the intention is that a SSA service may simultaneously implement multiple interfaces, e.g., a GET interface and a SOAP interface. At this time only the GET interface is defined.

If the SSA query is transmitted as an HTTP GET request then the URL to express a query is formed like this:

<Service.BaseURL>[POS=ra,dec][&SIZE=diam][&<otherArgs>]

**Example:**

http://www.myorg.org/ssa?POS=22.438,-17.2&SIZE=0.02

The `Service.BaseURL` is stored in the IVOA service registry (Hanisch et al. 2004).

### 3.2 Query

The most basic query is positional; a minimally compliant service MUST support simple positional queries including the `FORMAT` parameter. More generally, a simple query is defined in terms of a 4-dimensional physical parameter space:

- spatial region (for SSA an aperture on the sky)
- time region
- spectral region

A simple SSA service MUST support at least these four parameters. Various other parameters specific to spectral data are also defined, and should also be supported by the service if possible.

Unless otherwise specified, if the service does not support a defined query parameter it MUST permit the parameter to be present in the query without error, even if the parameter is not actually used as a query constraint by the service. All parameters are used to constrain the query; if a given parameter is not specified or is not supported by the service, a logical value of “all” is generally assumed.

#### 3.2.1 Mandatory Query Parameters

The following parameters MUST be implemented by a compliant service:

| # | Parameter | Sample value                | Physical unit   | Datatype  | Utype                |
|---|-----------|-----------------------------|-----------------|-----------|----------------------|
| 1 | POS       | 52,-27.8                    | dec. deg., ICRS | double(2) | SSA.Position         |
| 2 | SIZE      | 0.05                        | dec. deg.       | double    | SSA.RegionSize       |
| 3 | BAND      | 0.1,2.7D-7<br>(=10cm-2700Å) | m               | double(2) | SSA.SpectralBandpass |
| 4 | TIME      | 1998.45,1999.31             | UTC             | char(*)   | SSA.TimeBandpass     |
| 5 | FORMAT    | votable                     | -               | char(*)   | SSA.OutputFormat     |

**Table 1: Mandatory query parameters.**

While a compliant service must implement these parameters, a valid query can be composed from any combination of parameters; in other words, an SSA query does not have to be positional. All services must support queries by at least these five parameters, representing position on the sky, spectral and time coverage, and allowable client data formats.

### 3.2.1.1 POS

The center of the region of interest is expressed as the right-ascension and declination in decimal degrees in the ICRS coordinate system. The two values are comma separated with no embedded white space.

Example: `POS=52,-27.8`

### 3.2.1.2 SIZE

The diameter of the search region in decimal degrees.

Example: `SIZE=0.05`

A valid query does not have to specify a `SIZE` parameter. If `SIZE` is omitted in a positional query, the service should supply a default value intended to find nearby objects which are candidates for a match to the given object position. At most `TOP` candidate objects near the given position should be returned (see the discussion of the `TOP` parameter below).

### 3.2.1.3 BAND

The spectral bandpass given as a pair of comma separated values, each of which specifies a wavelength in vacuum in units of meters [RSM, Hanisch *et.al*, 2004]. The spectral rest frame MAY optionally be specified as either “source” or “observer”.

Example: `1D-6,3D-7,source`

For most queries the precision with which the spectral bandpass is specified probably does not matter; a rough bandpass broad enough to find all the interesting data will suffice. In some cases, for example a cutout service operating upon high resolution spectra, support at the service level for specifying the spectral rest frame could be important. If the service does not support specification of the spectral frame the syntax should be permitted but ignored.

### 3.2.1.4 TIME

The time or epoch of the observation defined by a pair of comma separated values. Times are specified in decimal years UTC, e.g., 1980.0.

**Note:**

Parameters such as `BAND` and `TIME` specify a range of acceptable values. A bounded range from `VAL1` to `VAL2` inclusive is specified as “`VAL1, VAL2`”. Either value may be omitted to specify an open range. For example,

“VAL1,” specifies any value greater than or equal to VAL1. Any dataset which contains data in the specified region is considered to match the query.

### 3.2.1.5 FORMAT

The `FORMAT` parameter defines the data formats the client is interested in retrieving via a subsequent `getData` call. The value is a comma-delimited list where each element can be any recognized MIME-type such as `application/x-votable+xml`, `application/fits`, `application/xml`, `text/plain`, `text/html`, and so forth.

In addition, these special values are defined:

| FORMAT    | Meaning  |
|-----------|--|
| ALL       | All formats supported by the service                             |
| COMPLIANT | Any SSA data model compliant format                              |
| EXTERNAL  | The opposite of <code>COMPLIANT</code> : an external data format |
| GRAPHIC   | Any of the graphics formats: JPEG, PNG, GIF                      |
| votable   | Shorthand for <code>application/x-votable+xml</code>             |
| fits      | Shorthand for <code>application/fits</code>                      |
| xml       | Shorthand for <code>application/xml</code>                       |

If `FORMAT` is omitted, `FORMAT=ALL` should be assumed.

The `FORMAT` parameter describes the desired format of returned data. If data is dynamically generated the service SHOULD generate the data in the requested format on the fly. Note `FORMAT` applies only to the data; the query response itself is always a `VOTable` document.

#### Note:

The service MUST permit reserved (=mandatory and optional) parameter names to be input in upper-case. The service MAY permit parameter names to be case insensitive since this allows upper-case names. By convention, upper case is used here for interface-defined terms in URLs, reserving lower case for service-defined parameters or parameter values.

### 3.2.2 Recommended Query Parameters

The following additional parameters SHOULD be implemented by a compliant service:

| # | Parameter | Sample value  | Unit | Datatype | Utype        |
|---|-----------|---------------|------|----------|--------------|
| 1 | APERTURE  | 0.00028 (=1") | dec. | double   | SSA.Aperture |

|   |            |             |      |         |                         |
|---|------------|-------------|------|---------|-------------------------|
|   |            |             | deg. |         |                         |
| 2 | SPECRES    | 5D-10 (=5Å) | m    | double  | SSA.SpectralResolution  |
| 3 | OBJTYPE    | Spectrum    | -    | char(*) | SSA.ObjectType          |
| 4 | COLLECTION | DSS2        | -    | char(*) | SSA.Collection          |
| 5 | TOP        | 20          | -    | int     | SSA.MaxTopRankedRecords |

**Table 2: Query parameters required for a fully-compliant service.**

### 3.2.2.1 APERTURE

The aperture through which the observation is performed, specified as a diameter in decimal degrees. The usage of the `APERTURE` parameter depends upon the type of service. For services which return pre-existing (atlas or pointed) data, `APERTURE` specifies the maximum aperture. For services which generate simulated data (data generated on-the-fly from more fundamental data, e.g., a spectrum generated from a spectral image cube) `APERTURE` defines the actual aperture size of the simulated instrument. Only circular apertures are currently supported as a query parameter. If an actual instrumental aperture is not circular a characteristic value should be used, for example the slit width on the sky for a slit spectrograph. A more precise specification of the actual aperture can be returned in the query response metadata for a dataset (see section 3.3.5.1).

If no aperture is explicitly specified by the client, a service which performs a simulated observation should use a default value appropriate to the data in question, for example, a circular aperture large enough to capture 98% of the signal from a point source in the aperture, knowing the PSF of the data in the desired spectral band. The size of the aperture to be used to generate the simulated data should be returned in the description of the data in the query response table.

### 3.2.2.2 SPECRES

The minimum spectral resolution, specified as the maximum acceptable FWHM of an unbroadened spectral line. The value is a floating point number in units of meters.

### 3.2.2.3 OBJTYPE

Possible values are `Spectrum`, `TimeSeries`, and `SED`. A given service will often support only a single type of data, identified when the service is registered. If the service returns multiple types of data the `OBJTYPE` parameter SHOULD be supported.

### 3.2.2.4 COLLECTION

The name of a data collection as defined by the creator of the data collection, for example `SDSS-DR2`, or `NRAO-VLA`. By data collection we refer to an organized,

uniform collection of datasets from a single source, for example a single data release from a survey, or an instrumental data collection from an observatory. The service should treat the search term as a case insensitive, minimum match string. For instance, “dss” would match either `dss1` or `ESO-DSS2`.

### 3.2.2.5 TOP

`TOP` limits the number of returned records in the query response table to the specified number of top ranking ones. Records are ranked according to a “score” heuristic. The details of the actual heuristic used are up to the service, but the general idea is that the better a candidate dataset matches the query, the higher the score it receives. Metrics such as distance from the specified position, or the degree of overlap with a specified bandpass or time interval, determine the score. If two datasets would otherwise have the same score, the service may use other unspecified dataset characteristics, such as some intrinsic data quality metric, to further rank candidate datasets. If the service implements a ranking heuristic the query response table should normally be returned sorted in order of decreasing score. `TOP` can also be used by the client to limit the size of the query response table in cases where the query might find a very large number of candidate objects.

### 3.2.3 Optional Query Parameters

The following parameters MAY be implemented by a compliant service:

| # | Parameter   | Sample value         | Phys. unit | Datatype  | Utype                  |
|---|-------------|----------------------|------------|-----------|------------------------|
| 1 | REDSHIFT    | -0.1, 2.0            | 1          | double(2) | SSA.Redshift           |
| 2 | SNR         | 5.0                  | -          | double    | SSA.MinSNR             |
| 3 | TARGETCLASS | star                 | -          | char(*)   | SSA.TargetClass        |
| 4 | CREATORID   | Ivo://auth/col#R1234 | -          | char(*)   | SSA.CreatorCreatorID   |
| 5 | PUBID       | ADS/col#R5983        | -          | char(*)   | SSA.PublisherCreatorID |
| 6 | COMPRESS    | (TRUE)               | -          | boolean   | SSA.Compress           |

#### 3.2.3.1 REDSHIFT

A photometric (observed) redshift range specified as a pair of comma separated floating point numbers. A negative redshift indicates a blueshift, e.g., an object in the local neighborhood with a proper motion towards the Earth. An open range may be used to specify a minimum or maximum value. The optical redshift convention should be used ( $d\lambda/\lambda$ ).

Example: `-0.1, 1.2`

### 3.2.3.2 SNR

The minimum signal-to-noise ratio of a candidate dataset, for example specified as the ratio of the mean signal to the RMS noise of the background.

### 3.2.3.3 TARGETCLASS

A comma delimited list of strings denoting the types of astronomical objects to be searched for. At the moment there is no standard list of valid values, although a number of simple acronyms are in common use.

Examples: *star, galaxy, pulsar, PN, AGN, QSO, GRB*

### 3.2.3.4 CREATORID

The IVOA dataset identifier, assigned at creation time by the creator of the parent data collection (survey project, observatory, etc.). Datasets can have a unique CreatorID assigned prior to publication of the data to the VO, for example when the data is generated by a processing pipeline, or ingested into the master archive for the data collection. When a CreatorID has been assigned this is the most universal way to refer to a dataset.

Example: `ivo://nrao.edu/vla#1998s2/4992a`

### 3.2.3.5 PUBID

The IVOA identifier of a dataset assigned by an IVOA publishing authority such as a datacenter or journal. A dataset may be published (replicated) in multiple locations, and may or may not have multiple distinct publisher IDs, all of which refer to the same replicated dataset. All datasets published to the VO may not necessarily have creator-assigned dataset IDs, but any data published to the VO by a publishing authority will have a publisher-assigned ID. All datasets should have creator IDs, but this is not the case for much legacy data.

Example: `ivo://stsci.edu/nrao-vla#/vla-1998/491392b`

A special case of a publisher ID is the ADEC/ADS identifier of a dataset, assigned if and when the dataset is referenced in a paper published to a scientific journal indexed by the ADS service.

Example: `ads/nrao-vla#1998s2/4992a`

By convention the “ivo://” prefix is omitted from ADEC/ADS dataset identifiers.

### 3.2.3.6 COMPRESS

If this flag is present, datasets returned via the *getData* method may be compressed for transmission back to the client. The `COMPRESS` parameter does not normally require a value. By default data is not compressed.

Compression is performed by applying a whole-file compression algorithm such as *gzip*, and updating the metadata of the returned document accordingly. Whether or not a dataset will be compressed should be specified as part of the meta-information of the underlying network protocol. In case of HTTP GET this is part of the value of `Content-Encoding`, for example, `x-gzip`. If a client enables compression it should be prepared to receive either compressed or normal data.

### 3.2.4 Service-Defined Parameters

The service MAY support additional service-defined parameters. Parameter names *must not* match any of the reserved parameter names defined herein, independent of case.

Service defined parameter names should not be uppercase in order to distinguish them from reserved parameters. Values should be simple enough to fit into the table data elements of VOTables.

## 3.3 Query Response

The output returned by a query is an XML document compliant with VOTable V1.1 or higher (VOTable 2004) and should be returned with a MIME-type of `application/x-votable+xml`.

#### Note:

The `FORMAT` parameter has no influence on the query response. `FORMAT` applies only to the returned datasets, not to the query response. The query response is always returned as a VOTable.

The VOTable MUST contain a `RESOURCE` element, identified with the tag `type="results"`, containing a single `TABLE` element with the results of the query. Additional `RESOURCE` elements MAY be present, but the usage of any such elements is not defined here.

The `RESOURCE` element MUST contain an `INFO` with `name="QUERY_STATUS"`. Its value attribute should be set to `"OK"` if the query executed successfully, regardless of whether any matching data were found. All other possible values for the value attribute are described in section 3.4.

#### Examples:

```
<INFO name="QUERY_STATUS" value="OK" />
```

```
<INFO name="QUERY_STATUS" value="OK">Successful Search</INFO>
```

In the response table each row represents a different dataset which is potentially available to the client. VOTable GROUP constructs are used to associate related groups of fields.

**Hint:**

Put constant values in PARAM elements instead of repeating them in each table row.

Names of fields and parameters are left to the service provider. UTYPEs of standard fields are required for identification and MUST be given and MUST comply with the SSA protocol (this document) and the SSA data model (McDowell et al. 2005). UCDs are optional but should be consistent with the SSA interface specification where used.

Service-defined output metadata may use foreign UTYPEs and UCDs as long as they do not clash - and can be easily distinguished - from mandatory and reserved SSA output columns.

In the following, query response parameters which MUST be provided are marked as '●'. Parameters which SHOULD be provided are marked as '●'. Parameters which MAY be provided are marked as '○'. Additional attributes from the SSA data model may appear in the query response table. Those fields explicitly mentioned here are those we feel are most useful to include in the query response.

### 3.3.1 Query Metadata

Query metadata describes the query itself.

|   |   | Utype          | Example        | Description                     |
|---|---|----------------|----------------|---------------------------------|
| 1 | ● | Query.Score    | 5.34           | degree of match to query params |
| 2 | ○ | Query.LName    | 'gb-05s3-4387' | logical name (identifier)       |
| 3 | ○ | Query.LNameKey | 'format'       | join fields (ID REF)            |

#### 3.3.1.1 Score

A record with a higher score more closely matches the query parameters. The query response table should normally be returned sorted in order of decreasing values of score, with the top-scoring items at the top of the list. The details of the heuristic used to compute score are left to the service. See the discussion of the TOP parameter in section 3.2.2.5.

### 3.3.1.2 LName

Records which represent different views of what is essentially the same object may be associated by giving them the same *logical name* (LName). The form of LName should be a simple identifier suitable for use as a filename. For example, if the same dataset is available in several different formats, these would all share the same logical name, with all the metadata being the same except for FORMAT and the access reference. While the LName field is marked as optional since it is not always applicable, The LName field SHOULD be provided if the query response table contains multiple records which are logically associated.

### 3.3.1.3 LNameKey

If multiple associated records share the same logical name, those fields which *differ* within the logical association group may be indicated using LNameKey. The ID-Ref construct of VOTable is used to specify the table fields used for the key. The value of LNameKey should be a comma-delimited list of reference strings matching corresponding ID tags associated with table fields. In simpler cases a single field will be referenced in the key.

## 3.3.2 Dataset Metadata

Dataset metadata describes an entire dataset, and is often specific to the type of data object stored in the dataset (image, spectrum, table, etc.).

|    |                                     | Utype                    | Example    | Description                               |
|----|-------------------------------------|--------------------------|------------|---|
| 1  | <input checked="" type="checkbox"/> | Dataset.Type             | 'spectrum' | Spectrum, TimeSeries, SED, etc.           |
| 2  | <input checked="" type="checkbox"/> | Dataset.DataModel        | 'SSA-V1.0' | data model of dataset                     |
| 3  | <input checked="" type="checkbox"/> | Dataset.Title            | 'NGC 346'  | brief descriptive title of dataset        |
| 4  | <input checked="" type="checkbox"/> | Dataset.SSA.NSamples     | 4096       | number of samples in dataset              |
| 5  | <input checked="" type="checkbox"/> | Dataset.SSA.Aperture     | 0.053      | aperture diameter (decimal degrees)       |
| 6  | <input type="checkbox"/>            | Dataset.SSA.TimeAxis     | 'time'     | timeCoord axis (foreign data)             |
| 7  | <input type="checkbox"/>            | Dataset.SSA.SpectralAxis | 'wave'     | spectralCoord axis (foreign data)         |
| 8  | <input type="checkbox"/>            | Dataset.SSA.FluxAxis     | 'flux'     | flux axis (foreign data)                  |
| 9  | <input checked="" type="checkbox"/> | Dataset.CreationType     | 'atlas'    | atlas, pointed, cutout, resampled         |
| 10 | <input type="checkbox"/>            | Dataset.Derivation       | 'observed' | observed, composite, simulated, synthetic |

### 3.3.2.1 Dataset Type

Specifies the type of dataset, i.e., the type of data object stored in the dataset. For SSA the allowed values are 'spectrum', 'timeSeries', and 'SED'.

*Do we need a new type 'TED' as well, for a time-energy distribution? This would be a time series which consists of a number of observed segments. This is not a SED since there is only one spectral value and hence no spectral energy distribution.*

*Do we need a 'photometry' data object? I left it out for the moment as it is not clear we have sufficient use-cases for this as a top level data object. It can still be used as a SED segment, within a SED dataset.*

### 3.3.2.2 Data Model

Specifies the data model used for the data object stored in the dataset. Typically version information is included in the data model specifier. The identifier for the standard SSA data model is defined in the data model itself and will typically be something like 'SSA-V1.0'.

The data model attribute is also used to identify foreign (externally defined) data. "Foreign" data is any data which uses a data model not defined by SSA. A query compliant SSA service which supplies foreign data MUST identify the data model used. The data model identifier for foreign data MUST not start with the prefix "SSA", which is reserved for use by SSA. An example of a foreign data model identifier would be something like 'GMOS-V1.0'.

### 3.3.2.3 SSA NSamples

The number of data elements in the SSA data object. For a SED this is the total number of samples in all SED segments. For a spectrum or time series it is the number of spectral or time samples or bins.

### 3.3.2.4 SSA Time, Spectral, Flux Axis

These fields identify the data axes for foreign data. They are not needed for SSA-compliant data. The SHOULD be given if the service returns foreign data. The values used depend upon the format of the foreign dataset. For a table format the field or column name would normally be specified.

### 3.3.2.5 Dataset Creation Type

The creation type of the dataset, defining how the data was computed, as defined in section 1.1.2. Allowable values are 'atlas', 'pointed', 'cutout', and 'resampled'.

### 3.3.2.6 Dataset Derivation

The derivation or source of the dataset, as defined in section 1.1.2. Allowable values are 'observed', 'composite', 'simulated', and 'synthetic'.

## 3.3.3 Target Metadata

Target metadata describes the astronomical target or object, if any, corresponding to an observational dataset. For a more detailed description of these fields refer to the SSA data model.

|   |   | Utype       | Example  | Description                 |
|---|---|-------------|----------|-----------------------------|
| 1 | o | Target.Name | 'ngc346' | name of astronomical object |

|   |   |                      |      |  |
|---|---|----------------------|------|--|
| 2 | ○ | Target.Class         | 'PN' | target class (star, galaxy, QSO, etc.) |
| 3 | ○ | Target.SpectralClass | 'B'  | spectral class                         |
| 4 | ○ | Derived.Redshift     | 1.34 | measured redshift for object           |
| 5 | ○ | Derived.VarAmpl      | 0.18 | variability amplitude                  |
| 6 | ○ | Derived.SNR          | 7.4  | signal to noise ratio                  |

*Perhaps it would be simpler to collapse 'Derived' and 'Target' into one Target metadata class? Attributes such as Redshift and SpectralClass are very similar.*

### 3.3.4 Curation Metadata

Curation metadata is used to identify and locate a dataset externally. This is generic metadata which applies to any dataset.

|   |   | Utype                | Example               | Description                           |
|---|---|----------------------|-----------------------|---------------------------------------|
| 1 | ● | Curation.Collection  | 'SDSS-DR2'            | collection name (identifier)          |
| 2 | ○ | Curation.Creator     | 'SDSS'                | creator identity (identifier)         |
| 3 | ● | Curation.CreatorID   | 'ivo://sdss/dr2#4538' | ID assigned by creator (IVOident)     |
| 9 | ○ | Curation.PublisherID | 'ads/sdss#5942R2'     | ID assigned by a publisher (IVOident) |
| 6 | ○ | Curation.Date        | '1996-12-19T16:39'    | creation date (ISO Date string)       |
| 7 | ○ | Curation.Version     | '1.0a'                | version string                        |

The Curation metadata is described in more detail in the SSA data model and in section 3.2.3, as metadata such as `COLLECTION` and `CREATORID` are important query parameters.

A key point about this metadata is that, while the allowable values for variables such as `COLLECTION` and `CREATORID` may not be easily known apriori, they can be determined by querying the data service and seeing what values come back, then using these values to refine subsequent queries.

The values for variables such as `COLLECTION` and `CREATOR` should be "simple" names. A client application can look these names up in the registry to gain a full description of the resource referred to. Simple names are easier to display in a table, more efficient of storage, and avoid update problems for many-to-one references if the metadata for a frequently referenced resource such as a collection or data producer is changed.

### 3.3.5 Characterization Metadata

Characterization metadata is used to physically characterize a dataset. This is generic metadata which applies to any dataset. All characterization values are assumed to be approximate; it is far more important to provide approximate values than to fail to characterize the data because the values cannot be accurately computed.

### 3.3.5.1 Coverage

Coverage metadata specifies the coverage of the dataset in the three physical measurement axes: spatial, spectral, and time. A measure of the coverage in flux (the observable) is also given.

|    |                                     | Utype                             | Description                              |
|----|-------------------------------------|-----------------------------------|--|
| 1  | <input checked="" type="checkbox"/> | Coverage.Location.Spatial         | position (RA DEC)                        |
| 2  | <input checked="" type="checkbox"/> | Coverage.Location.Time            | observation time characteristic value    |
| 3  | <input checked="" type="checkbox"/> | Coverage.Location.Spectral        | spectral bandpass characteristic value   |
| 4  | <input checked="" type="checkbox"/> | Coverage.Location.Spectral.BandID | bandpass ID as in band or filter name    |
| 5  | <input type="checkbox"/>            | Coverage.Bounds.Spatial           | Aperture limits (polygon in RA,DEC IRCS) |
| 6  | <input checked="" type="checkbox"/> | Coverage.Bounds.Time              | low/high time values                     |
| 7  | <input checked="" type="checkbox"/> | Coverage.Bounds.Spectral          | low/high spectral values                 |
| 8  | <input checked="" type="checkbox"/> | Coverage.Bounds.Flux              | flux low/high limits (Jansky)            |
| 9  | <input type="checkbox"/>            | Coverage.Fill.Spatial             | aperture filling factor (1.0)            |
| 10 | <input type="checkbox"/>            | Coverage.Fill.Time                | time window filling factor (1.0)         |
| 11 | <input type="checkbox"/>            | Coverage.Fill.Spectral            | spectral window filling factor (1.0)     |

Coverage is part of the Characterization data model, which is integrated into the SSA data model. A full description of Coverage as used in SSA is given in the SSA data model. The basic idea is that 'Location' specifies the characteristic value of the data in the three physical axes. 'Bounds' specifies the bounds in the three physical axes of measurement. If the data is not fully sampled within the bounds of measurement this is indicated by a filling factor of less than 1.0.

Several special cases which are important for SSA deserve mention:

- The details of how the data is sampled are not specified at the level of the SSA query since the SSA data model allows the data to be irregularly sampled (to fully understand how the data is sampled one has to get the actual dataset). A rough indication of the sampling can be derived given the bounds and the number of sample elements for the dataset.
- While the spectral bandpass is formally specified by the numerical bounds, the optional `BandID` attribute can be used to assign a familiar name to the bandpass, such as a filter or waveband name.
- The spatial bounds of the measurement are trivial for basic SSA since the interface assumes a circular aperture. The location of the aperture on the sky (if applicable) is given by `Location.Spatial`. The diameter of the circular aperture is given by `SSA.Aperture` in the Dataset metadata. Specifying the spatial bounds of coverage does not make much sense for a simple circular aperture so this metadata is optional for the basic SSA interface. If however a non-circular aperture is used, `Bounds.Spatial` may be used to specify the footprint of the aperture on the sky as a polygon in world coordinates (RA, DEC in IRCS), and `SSA.Aperture` should be set to the mean aperture.

- The sensitivity and dynamic range of the data can be specified in physical units (Janskys) via `Bounds.Flux`. The lower flux limit is the flux sensitivity of the data, i.e., the minimum detectable flux. The upper flux limit is determined by the maximum linear dynamic range of the measurement.

### 3.3.5.2 Accuracy

Accuracy metadata gives a rough measure of the measurement quality in the various independent and dependent axes.

|    |   | Utype                        | Description                           |
|----|---|------------------------------|---------------------------------------|
| 1  | ● | Accuracy.Spatial.Calibrated  | is data spatially calibrated          |
| 2  | ○ | Accuracy.Spatial.SysErr      | astrometric systematic error          |
| 3  | ○ | Accuracy.Spatial.StatErr     | astrometric statistical error         |
| 4  | ● | Accuracy.Spatial.Resolution  | spatial resolution (FWHM PSF)         |
| 5  | ● | Accuracy.Time.Calibrated     | is data temporally calibrated         |
| 6  | ○ | Accuracy.Time.SysErr         | time measurement systematic error     |
| 7  | ○ | Accuracy.Time.StatErr        | time measurement statistical error    |
| 8  | ● | Accuracy.Time.Resolution     | time resolution (FWHM TSF)            |
| 9  | ● | Accuracy.Spectral.Calibrated | is data spectrally calibrated         |
| 10 | ○ | Accuracy.Spectral.SysErr     | spectral coordinate systematic error  |
| 11 | ○ | Accuracy.Spectral.StatErr    | spectral coordinate statistical error |
| 12 | ● | Accuracy.Spectral.Resolution | spectral resolution (FWHM LSF)        |
| 13 | ● | Accuracy.Flux.Calibrated     | is data flux calibrated               |
| 14 | ○ | Accuracy.Flux.SysErr         | flux value systematic error           |
| 15 | ○ | Accuracy.Flux.StatErr        | flux value statistical error          |

The accuracy model is discussed in detail in the SSA data model. Specific issues as they relate to the SSA query are noted below.

Whether or not any absolute calibration has been performed is indicated by the `Calibrated` attribute, which applies to all independent and dependent axes. Allowable values are “uncalibrated”, “relative”, and “absolute”, depending upon whether the axis is uncalibrated, has a relative calibration, or is calibrated with respect to some physical standard of reference. If the calibration status is not specified, fully calibrated data is assumed.

Although resolution is not an error estimate it is a good guide for estimating other types of error, especially statistical error, and should be given where appropriate, depending upon the type of data. The spectral resolution SHOULD be given for a spectrum, and the time resolution SHOULD be given for a time series. The spatial resolution is important for all observed or simulated data and SHOULD be given for all SSA data except SEDs, where it may vary a great deal for different segments.

Systematic errors are optional, and if known, should be specified as a bound on the maximum systematic error.

### 3.3.5.3 Reference Frames

Reference frame metadata is required to specify the coordinate systems used in the dataset metadata.

|   |   | Utype                | Description                      |
|---|---|----------------------|----------------------------------|
| 1 | ○ | Frame.Sky.Type       | coordinate frame (ICRS)          |
| 2 | ○ | Frame.Sky.Equinox    | coordinate system equinox (2000) |
| 3 | ○ | Frame.Time.System    | timescale (TT)                   |
| 4 | ● | Frame.Time.SIDim     | SI factor and dimension          |
| 5 | ● | Frame.Spectral.SIDim | SI factor and dimension          |
| 6 | ● | Frame.Flux.SIDim     | SI factor and dimension          |
| 7 | ● | Frame.Flux.UCD       | UCD of flux value                |

For a detailed description of reference frame metadata as used in SSA refer to the SSA data model document. In general, frame metadata can be omitted when the default coordinate systems are used.

The `SIDim` attributes SHOULD be provided for spectra but are optional for other types of data. They provide a means (via dimensional analysis) to specify the physical units on an axis without having to parse complex units specification strings. Details on usage are given in the SSA data model.

### 3.3.6 Instrument Metadata

Instrument metadata describes the instrument and instrument setup used to conduct an observation. Instrument metadata is optional, as it applies only to simple observational data, as it is very difficult to standardize instrument metadata for the great number of complex instruments in use now or in the past.

|   |   | Utype               | Description                  |
|---|---|---------------------|------------------------------|
| 1 | ○ | Instrument.Name     | instrument name (identifier) |
| 2 | ○ | Instrument.Exposure | exposure time in seconds     |
| 3 | ○ | Instrument.<other>  | service-defined              |

The instrument name (if any) and aggregate exposure time are useful metadata for instrumental observations. Additional instrument metadata may be given as indicated above. The metadata used to describe a specific instrument is left up to the data provider, although it is suggested that the VO Observation data model be used as a guide. If extensive instrumental metadata is provided it is suggested that this be moved to VOTable extensions via metadata extension mechanism (Bonnarel et. al. 2005).

### 3.3.7 Access Metadata

Access metadata is required to tell a client how to access the datasets described in the SSA query response.

|   |   | Utype            | Example                                    | Description              |
|---|---|------------------|--|--------------------------|
| 1 | ☐ | Access.Reference | <code>http://myorg.org/get?oid=A134</code> | dataset access URL       |
| 2 | ☐ | Access.Format    | <code>application/x-votable+xml</code>     | MIME type of dataset     |
| 3 | ● | Access.Size      | 185000                                     | approximate dataset size |

#### 3.3.7.1 Access Reference

The access reference is a URI (typically a URL) which can be used to synchronously retrieve the specific dataset described in a row of the query table response. The dataset pointed to by the access reference often does not exist until it is accessed, at which time it is computed on the fly.

Since the datasets supported by SSA are typically small (compared to images), SSA does not currently support data staging and asynchronous data access. Support for this may need to be added in the future, e.g., to support generation of simulated or synthetic data.

#### 3.3.7.2 Output Format

The file format of a candidate dataset is specified by its MIME type. Both uncompressed and compressed data can be indicated in this fashion.

The file format says nothing about the data model used by whatever data object is stored in the file; this is specified by the `DataModel` attribute discussed in section 3.3.2.2.

A single data object may be available in multiple file formats. In such a case a *logical name* (section 3.3.1.2) should be given to indicate that the different formats all correspond to the same data object.

#### 3.3.7.3 Dataset Size Estimate

The estimated size of the dataset in bytes SHOULD be given to help the client determine whether it is worthwhile to retrieve the data. Only an approximate, order of magnitude type value is required.

### 3.3.8 Additional Service-Defined Metadata

A given service MAY return additional query response metadata not defined by the SSA protocol. This additional metadata may take the form of additional table columns, or additional `RESOURCE` elements in the query response `VOTable`.

### 3.4 Error Response and Other Unsuccessful Results

An INFO element within the "results" RESOURCE element of the VOTable is used to indicate success or failure of the image query. As described in the previous section, the INFO element must have name="QUERY\_STATUS"; if the query is successful (regardless of whether any spectrum rows are returned) the value attribute is set to "OK". The remainder of this section defines other possible values to indicate that the query was unsuccessful in some way. When the query is unsuccessful, the contents of INFO element (i.e. its PCDATA child node) SHOULD contain an error message suitable for display.

When the query is unsuccessful (in any of senses described below), the resulting VOTable is not required to contain any other elements as specified in section 3.3; although, it is not an error to do so.

The other allowed values for value attribute besides "OK" are as specified below.

#### 3.4.1 Error

The server failed to process the query. Possible reasons include:

- The input query contained a syntax error.
- The way the query was posed was invalid for some reason, e.g., due to an invalid query region specification.
- A constraint parameter value was given an illegal value; e.g. DEC=91.
- The server trapped an internal error (e.g., failed to connect to its database) preventing further processing.

In this case, the inclusion of a descriptive error message SHOULD be returned.

#### 3.4.2 Overflow

The query produced results that exceeded the limits of the service in some way. For instance, this could be the number of matching spectra. In this case, the service SHOULD include an error message indicating the nature of the overflow condition (see also section 4.4).

#### Example:

```
<INFO name="QUERY_STATUS" value="ERROR">DEC out of range:  
DEC=91</INFO>  
<INFO name="QUERY_STATUS" value="OVERFLOW">Number of matching  
spectra exceeds limit of 100</INFO>
```

## 4 Data Access

The spectrum retrieval request allows a client to retrieve a single spectrum or chunks of SEDs given an access reference `acref` as returned by a prior spectrum query. Since a spectrum query is required to obtain an `acref`, no further constraints are placed on the form the `acref` takes. This has the effect of hiding the details of the `acref` URL from the client, making it easy to layer an implementation of a getter web method on top of an existing retrieval service, and making it easier to hide changes to the implementation of existing services.

### 4.1 Access Reference URL

The access reference `acref` is a simple URL (RFC 1738). If the client issues a HTTP GET request using this URL, and the request is successful, the client will receive document of the type given in query response column with the `ucd="meta.code.mime"`. Since a query is REQUIRED to obtain an `acref`, no requirements are placed on the form the `acref` takes. This has the effect of hiding the details of the `acref` URL from the client, making it easy to layer an implementation of the GET web method on top of an existing data retrieval service, and making it easier to hide changes to the implementation of existing services.

### 4.2 Data Format

The output of the getter method MUST be a single spectrum document returned with a MIME-type identifying the file format. As described in (McDowell et al. 2004) a retrieved SED document may consist of several chunks of SED points and sub-spectra. Depending on the format of the spectrum various MIME types are possible; for example, a MIME-type `text/html` may be used when the `acref` URL points to an HTML description and/or preview of the spectrum. Some commonly used formats are:

- `application/x-votable+xml`
- `application/fits`
- `application/xml`
- `image/jpeg` - graphics preview
- `text/plain` - ascii table

### 4.3 Data Compression

If query parameter COMPRESS is present then the service MAY apply a compression method.

In case of an HTTP GET the keyword `Content-Encoding` informs the receiver about the encoding, like for instance `x-gzip`. Care needs to be taken to treat the encoding distinct from the MIME-type of the encoded data stream.

In general, the metadata query SHOULD return info about the supported compression method if any. This is particularly important if lossy compression is applied.

#### 4.4 Error Response

Depending on the environment the client SHOULD take the usual precautions to catch errors and exceptions. It is up to the application logic and user interface how to treat the many possible causes

### 5 Service Metadata

Compliant spectrum services are described in two ways:

1. By registering with a registry service and supplying service metadata about themselves and any associated data collections.
2. By responding to a metadata query via the service metadata query method.

When the service provider registers an SSA service, the registry can execute the metadata query to collect the capability metadata. The gathering of service and capability metadata from all such services enables a client to use the registry to discover the services most appropriate for a particular computation.

#### 5.1 Metadata Query

1. *We probably want to move away from the use of VOTable for the metadata query, and instead have the service return a VOResource record for the service. This is structured XML which directly describes the capabilities of the service, including the functionality implemented, any additional non-standard query parameters, and so forth.*
2. *The text below has not yet been updated to reflect these changes..*

A compliant service MUST support spectrum queries with `FORMAT=METADATA`, used to query the service metadata; only metadata is returned by the service in this case. In particular, the response to this query advertises two things about the service:

- supported input parameters (section 3.2)
- possible output columns (3.3)

Note that the SSA specification is designed so that a client does not need to know this information to make use of the service. It is useful for communicating non-standard input and output parameters. The metadata query can also be helpful to a registry for verification purposes.

The overall structure of the VOTable by a metadata request is the same as described in section 3 except that it contains no table rows. Each input parameter supported by the service SHOULD be listed as a `PARAM` element of the `RESOURCE` that normally contains the spectrum table. Each `PARAM` SHOULD have a name attribute of the form "INPUT:param\_name", or "OUTPUT:param\_name" where *param\_name* is the parameter name as it appears in the query. For example, name="INPUT:POS" refers to the `POS` input parameter. All input parameters meant to be available to clients of the service MUST be listed as `PARAM` elements, including REQUIRED parameters (`POS`, `SIZE`, and `FORMAT`), optional parameters (defined in section 3.2.2), and non-standard parameters specific to the service (section 3.2.4). The `PARAM` MAY contain a value attribute with the default value that will be assumed if the parameter is not set in the query. Implementors are encouraged to include, as children of the `PARAMS`, `DESCRIPTION` elements to describe the parameter and (where appropriate) `VALUES` elements to given allowed ranges or values.

## 5.2 Metadata Response

The columns that are returned by a spectrum query are described with the `FIELD` elements exactly as they are described in a normal spectrum query response. When `FORMAT=METADATA` is given, all other input parameters SHOULD be ignored.

## 5.3 Service Registration

In the context of this document the registry refers to the *Resource Metadata for the Virtual Observatory* standard (Hanisch et al. 2004).

### 5.3.1 Overflow Conditions

An overflow condition occurs when a spectrum query exceeds the capabilities of a service. Those capability meta data are defined in section *Capabilities metadata* of (Hanisch et al. 2004). Currently there are two such conditions:

1. `Service.MaxSearchRadius` (float in decimal degrees)
2. `Service.MaxReturnRecords` (int)

A service MAY define further overflow conditions. The metadata query SHOULD return information about the potential cause of such conditions.

### 5.3.2 Service Type

The `SERVICE.TYPE` in the registry corresponds to the supported `SEGMENTTYPES`. A service MAY support `PHOTOMETRY` points and/or `SPECTRUM` and/or `TIMESERIES`. If more than one `SEGMENTTYPE` is supported then the value of `SERVICE.TYPE` is `Mixed`.

## Appendix A: Sample Query Response for *Mixed* Service

```
<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.1"
  xmlns:sdm="http://www.ivoa.net/xml/SpectralDataModel/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.ivoa.net/xml/VOTable/v1.1">
  <RESOURCE type="results">
    <INFO name="QUERY_STATUS" value="OK">Successful Search</INFO>
    <TABLE>
      <GROUP utype="sdm:SSA.Access">
        <FIELDref ref="acref"/>
        <FIELDref ref="fileSize"/>
        <FIELDref ref="score"/>
        <FIELDref ref="logicalName"/>
        <PARAM name="format" datatype="char" arraysze="*" ucd="meta.code.mime"
          utype="sdm:SSA.Access.Format" value="application/x-votable+xml"/>
        <PARAM name="encoding" datatype="char" arraysze="*"
          ucd="meta.ref.url;meta.code" utype="sdm:SSA.Access.Encoding"
          value="application/x-gzip"/>
      </GROUP>
      <GROUP utype="sdm:SSA.Dataset">
        <FIELDref ref="datasetType"/>
        <PARAM name="dataset derivation" datatype="char" arraysze="*"
          utype="sdm:SSA.Dataset.Derivation" value="observed"/>
        <FIELDref ref="nelements"/>
      </GROUP>
      <GROUP utype="sdm:SSA.Coverage">
        <GROUP utype="sdm:SSA.Coverage.Location">
          <PARAM name="Celestial Coordinates" datatype="float" arraysze="2"
            utype="sdm:SSA.Coverage.Location.Spatial" ucd="pos.eq" unit="deg"
            value="132.4210 -12.1232"/>
          <FIELDref ref="locationTime"/>
        </GROUP>
        <GROUP utype="sdm:SSA.Coverage.Bounds">
          <FIELDref ref="boundsTime"/>
          <FIELDref ref="boundsSpectral"/>
        </GROUP>
      </GROUP>
      <PARAM name="creator" datatype="char" arraysze="*"
        utype="sdm:SSA.Curation.Creator" value="my observatory"/>
      <!-- further optional parameters and fields could go here -->
      <!-- further service defined parameters and fields could go here -->
      <FIELD name="access reference" datatype="char" arraysze="*" ID="acref"
        ucd="meta.ref.url" utype="sdm:SSA.Access.Reference"/>
      <FIELD name="file size" datatype="long" ID="fileSize"
        ucd="phys.size;meta.file" unit="Byte" utype="sdm:SSA.Access.Size"/>
      <FIELD name="score" datatype="double" ID="score"
        ucd="meta.code.number;stat.likelihood" utype="sdm:SSA.Query.Score"/>
      <FIELD name="logical name" datatype="char" arraysze="*" ID="logicalName"
        ucd="meta.id.assoc;meta.dataset" utype="sdm:SSA.Query.LName"/>
      <FIELD name="target name" datatype="char" arraysze="*"
        utype="sdm:SSA.Target.Name"/>
      <FIELD name="number of samples" datatype="int" ID="nelements"
        ucd="meta.number" utype="sdm:SSA.Dataset.SSA.NEelements"/>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

```

<FIELD name="type of dataset" datatype="char" arraysize="*"
  ID="datasetType" utype="sdm:SSA.Dataset.Type"/>
<FIELD name="Midpoint of Exposure" datatype="float" ID="locationTime"
  utype="sdm:SSA.Coverage.Location.Time" ucd="time.obs" unit="d"/>
<FIELD name="start/stop time of exposure" datatype="float" arraysize="2"
  ID="boundsTime" utype="sdm:SSA.Coverage.Bounds.Time" ucd="time.expo"
  unit="d"/>
<FIELD name="upper/lower bounds of spectral bandwidth"
  datatype="float" arraysize="2" ID="boundsSpectral"
  utype="sdm:SSA.Coverage.Bounds.Spectral"
  ucd="instr.bandwidth" unit="Angstrom"/>
<DATA>
  <TABLEDATA>
    <TR>
      <TD><![CDATA[http://spectrum.org/getme?oid=A134]]></TD>
      <TD>185270</TD>
      <TD>2.34</TD>
      <TD>A134_M100</TD>
      <TD>NGC 4321</TD>
      <TD>2200</TD>
      <TD>spectrum</TD>
      <TD>52148.3252</TD>
      <TD>52148.3240 52148.3264</TD>
      <TD>1250.0 3000.0</TD>
    </TR>
    <TR>
      <TD><![CDATA[http://spectrum.org/getme?oid=A139]]></TD>
      <TD>22340</TD>
      <TD>1.03</TD>
      <TD>A139_M100</TD>
      <TD>NGC 4321</TD>
      <TD>120</TD>
      <TD>spectrum</TD>
      <TD>50204.898</TD>
      <TD>50204.892 50204.904</TD>
      <TD>1510.2 1587.4</TD>
    </TR>
    <!--TR> ... </TR-->
  </TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>

```

## Appendix B: Query Response for *Spectrum/TimeSeries* Service

```

<?xml version="1.0" encoding="UTF-8"?>
<VOTABLE version="1.1"
  xmlns:sdm="http://www.ivoa.net/xml/SpectralDataModel/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.ivoa.net/xml/VOTable/v1.1">
  <RESOURCE type="results">
    <INFO name="QUERY_STATUS" value="OK">Successful Search</INFO>
    <TABLE>
      <GROUP utype="sdm:SSA.Access">
        <FIELDref ref="acref"/>
        <FIELDref ref="fileSize"/>
        <FIELDref ref="score"/>
        <FIELDref ref="logicalName"/>

```

```

<PARAM name="format" datatype="char" arraysize="*" ucd="meta.code.mime"
  utype="sdm:SSA.Access.Format" value="application/x-votable+xml"/>
<PARAM name="encoding" datatype="char" arraysize="*"
  ucd="meta.ref.url;meta.code" utype="sdm:SSA.Access.Encoding"
  value="application/x-gzip"/>
</GROUP>
<GROUP utype="sdm:SSA.Dataset">
  <FIELDref ref="datasetType"/>
  <FIELDref ref="nelements"/>
  <PARAM name="creation type" datatype="char" arraysize="*"
    utype="sdm:SSA.Dataset.CreationType" value="filtered"/>
</GROUP>
<GROUP utype="sdm:SSA.Curation">
  <PARAM name="publisher" datatype="char" arraysize="*" ucd="meta.id"
    utype="sdm:SSA.Curation.PublisherID"
    value="ivo://archive.eso.org/ox443"/>
  <FIELDref ref="datasetId"/>
</GROUP>
<GROUP utype="sdm:SSA.Coverage">
  <GROUP utype="sdm:SSA.Coverage.Location">
    <PARAM name="Celestial Coordinates" datatype="float" arraysize="2"
      utype="sdm:SSA.Coverage.Location.Spatial" ucd="pos.eq" unit="deg"
      value="132.4210 -12.1232"/>
    <FIELDref ref="locationTime"/>
  </GROUP>
  <GROUP utype="sdm:SSA.Coverage.Bounds">
    <FIELDref ref="boundsTime"/>
    <PARAM name="upper/lower bounds of spectral coverage"
      datatype="float" arraysize="2"
      utype="sdm:SSA.Coverage.Bounds.Spectral"
      ucd="instr.bandwidth" unit="MHz" value="1400 1800"/>
  </GROUP>
</GROUP>
<!-- further optional parameters and fields could go here -->
<!-- further service defined parameters and fields could go here -->
<FIELD name="access reference" datatype="char" arraysize="*" ID="acref"
  ucd="meta.ref.url" utype="sdm:SSA.Access.Reference"/>
<FIELD name="file size" datatype="long" ID="fileSize"
  ucd="phys.size;meta.file" unit="Byte" utype="sdm:SSA.Access.Size"/>
<FIELD name="score" datatype="double" ID="score"
  ucd="meta.code.number;stat.likelihood" utype="sdm:SSA.Query.Score"/>
<FIELD name="logical name" datatype="char" arraysize="*" ID="logicalName"
  ucd="meta.id.assoc;meta.dataset" utype="sdm:SSA.Query.LName"/>
<FIELD name="type of dataset" datatype="char" arraysize="*"
  ID="datasetType" utype="sdm:SSA.Dataset.Type"/>
<FIELD name="number of samples" datatype="int" ID="nelements"
  ucd="meta.number" utype="sdm:SSA.Dataset.SSA.NElements"/>
<FIELD name="dataset ID" datatype="char" arraysize="*" ID="datasetId"
  ucd="meta.id,meta.dataset" utype="sdm:SSA.Curation.DatasetID"/>
<FIELD name="midpoint of Exposure" datatype="float" ID="locationTime"
  utype="sdm:SSA.Coverage.Location.Time" ucd="time.obs" unit="d"/>
<FIELD name="start/stop time of exposure" datatype="float" arraysize="2"
  ID="boundsTime" utype="sdm:SSA.Coverage.Bounds.Time"
  ucd="time.expo" unit="d"/>
<DATA>
  <TABLEDATA>
    <TR>
      <TD><![CDATA[http://spectrum.org/getme?oid=DS6321_B]]></TD>
      <TD>12476</TD>
      <TD>3.21</TD>
      <TD>DS6321_4321</TD>
      <TD>timeseries</TD>
      <TD>52</TD>
    </TR>
  </TABLEDATA>
</DATA>

```

```

        <TD>DS6321</TD>
        <TD>51264.24045</TD>
        <TD>51264.24000 51264.24090</TD>
    </TR>
    <TR>
        <TD><![CDATA[http://spectrum.org/getme?oid=DS6314_B]]></TD>
        <TD>11006</TD>
        <TD>3.02</TD>
        <TD>DS6314_4321</TD>
        <TD>timeseries</TD>
        <TD>20</TD>
        <TD>DS6314</TD>
        <TD>51388.24248</TD>
        <TD>51388.24208 51388.24288</TD>
    </TR>
    <!--TR> ... </TR-->
</TABLEDATA>
</DATA>
</TABLE>
</RESOURCE>
</VOTABLE>

```

## References

### [Allen et al. 2003]

Allen, M., Boch, T., 2003, Plotting SEDs in the AVO prototype: Catalog Requirements, Conventions and Conversions, [http://www.euro-vo.org/internal/Avo/WorkPackageTwoTwo/SEDs\\_in\\_AVO.pdf](http://www.euro-vo.org/internal/Avo/WorkPackageTwoTwo/SEDs_in_AVO.pdf)

### [Bonnarel et al. 2004]

Bonnarel, F. et al., 2004, Proposal for an evolution of the SIA protocol V1.00, <http://www.ivoa.net/Documents/latest/SIAPEvol.html>

### [Bunclark/Rots 1996]

Bunclark, P., Rots, A. 1996, Precise re-definition of DATE-OBS Keyword encompassing the millennium, <http://www.cv.nrao.edu/fits/documents/standards/year2000.txt>

### [Dolensky/Tody 2004]

Dolensky, M., Tody, D., 2004, Survey among Spectral Data Providers and Consumers, ASP Conf. Ser., 314, 338

### [Hanisch et al. 2004]

Hanisch, R. 2004, Resource Metadata for the Virtual Observatory V 1.01, <http://www.ivoa.net/Documents/REC/ResMetadata/RM-20040426.html>

### [McDowell et al. 2004]

McDowell, J. et al. 2004, Work in progress: IVOA SED Data Model V0.93, <http://hea-www.harvard.edu/~jcm/vo/docs/spec0.93.html>

### [Osuna/Salgado 2004]

Osuna, P., Salgado, J. 2004, Simple Spectral Access for ISO data V1.0, <http://www.ivoa.net/Documents/latest/SADimEq.html>

### [RFC 1738]

Berners-Lee et al. 1994, Uniform Resource Locators (URL), IETF RFC 1738, <http://www.ietf.org/rfc/rfc1738.txt>

### [RFC 2119]

Bradner et al. 1997, Key words for use in RFCs to Indicate Requirement Levels, IETF RFC 2119, <http://www.ietf.org/rfc/rfc2119.txt>

**[Rots 2005]**

Rots, A., Space-Time Coordinate Metadata for the Virtual Observatory V1.10,  
<http://www.ivoa.net/Documents/latest/STC.html>

**[Tody/Plante 2004]**

Tody, D., Plante, R. 2004, Simple Image Access Specification V1.0,  
<http://www.ivoa.net/Documents/WD/SIA/sia-20040524.html>

**[UCD 2004]**

Derriere, S., Preite Martinez, A., Williams, R. 2004, Work in progress: UCD  
(Unified Content Descriptor) - moving to UCD1+,  
<http://www.ivoa.net/Documents/PR/UCD/UCD-20041026.html>

**[Valdes 2003]**

Valdes, F. 2003, A Virtual Observatory Data Model,  
<http://iraf.noao.edu/projects/vo/dal/datamodel.html>

**[VOTable 2004]**

Ochsenbein, F. et al., 2004, VOTable Format Definition V1.1,  
<http://www.ivoa.net/Documents/REC/VOTable/VOTable-20040811.html>