



*International  
Virtual  
Observatory  
Alliance*

## Simple Spectral Access Protocol

### Version 0.95

*IVOA Working Draft May 12, 2006*

**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

Version 0.91 Oct. 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). SSA defines a common data model and query interface for this class of data.

The “simple” in Simple Spectral Access refers to the 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 **queryData** operation 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** operation (currently implemented merely as a data access URL) 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 **getCapabilities** operation is defined by which an external client (such as the Registry or a client application) can retrieve service metadata to determine the capabilities of a given SSA service instance.

A fourth **stageData** operation is envisaged but it not currently defined. This will be used to perform asynchronous or bulk staging of data.

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.

Parts of this protocol were adapted from the OpenGIS (Open Geospatial Consortium, Inc.) SSA service (SSA service) Implementation Specification. In particular, the basic service elements and the use of the HTTP protocol to formulate requests and responses is patterned after SSA service. Parts of the text of this specification were adapted directly from the SSA service specification.

## Contents

1	Terms and Definitions	5	
2	Abbreviations	5	
3	Overview	5	
3.1	Interface	5	
3.2	Basic Usage	6	
3.2.1	Query, Query Response		6
3.2.2	Access Individual Datasets via URL		6
3.2.3	Data Formats		6
4	Requirements for Compliance	6	
4.1	Minimal Service	7	
5	Basic Service Elements	7	

5.1	Introduction	7	
5.2	Version numbering and negotiation	7	
5.2.1	Version number form and value		7
5.2.2	Version number changes		8
5.2.3	Appearance in requests and in service metadata		8
5.2.4	Version number negotiation		8
5.3	General HTTP request rules	9	
5.3.1	Introduction		9
5.3.2	Reserved characters in HTTP GET URLs		9
5.3.3	HTTP GET		10
5.3.4	HTTP POST		11
5.4	General HTTP response rules	11	
5.5	Numeric and Boolean values	11	
5.6	Output formats	12	
5.7	Request parameter rules	12	
5.7.1	Parameter ordering and case		12
5.7.2	Range-List Parameters		12
5.8	Common request parameters	13	
5.8.1	VERSION		13
5.8.2	REQUEST		13
5.8.3	Extended capabilities and operations		13
5.9	Service result	14	
5.10	Service exceptions	14	
6	SSA Concepts	14	
6.1	Types of Data	14	
6.1.1	Types of Services		15
6.2	Data Model	16	
6.3	Data Representation	17	
7	SSA Operations	17	
7.1	Introduction	17	
7.2	Methods & Protocols	17	
7.3	QueryData (required)	18	
7.4	Query Parameters	18	
7.4.1	Mandatory Query Parameters		18
7.4.2	Recommended Query Parameters		20
7.4.3	Service-Defined Parameters		23
7.5	Query Response	24	
7.5.1	Query Metadata		25
7.5.2	Dataset Metadata		26
7.5.3	Target Metadata		27
7.5.4	Curation Metadata		27
7.5.5	Characterization Metadata		28
7.5.6	Instrument Metadata		31
7.5.7	Access Metadata		31
7.5.8	Additional Service-Defined Metadata		32
7.6	Error Response and Other Unsuccessful Results	32	
7.6.1	Error		32
7.6.2	Overflow		33
8	Data Access	33	

8.1	Access Reference URL	33	
8.2	Data Format	34	
8.3	Data Compression	34	
8.4	Error Response	34	
9	Service Metadata	34	
9.1	Metadata Query	35	
9.2	Metadata Response	35	
9.3	Service Registration	35	
9.3.1	Overflow Conditions		36
9.3.2	Service Type		36
Appendix A: Sample Query Response for <i>Mixed</i> Service		36	
Appendix B: Query Response for <i>Spectrum/TimeSeries</i> Service		38	
References		39	

## 1 Terms and Definitions

## 2 Abbreviations

## 3 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.

### 3.1 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 *queryData* 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 *getCapabilities* 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.

## 3.2 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.

### 3.2.1 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?VERSION=1.0&REQUEST=queryData&POS=22.438,-17.2&SIZE=0.02
```

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

### 3.2.2 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.

### 3.2.3 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.

## 4 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).

## 4.1 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 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.

## 5 Basic Service Elements

### 5.1 Introduction

This clause specifies aspects of SSA service behaviour that are independent of particular operations or are common to several operations.

### 5.2 Version numbering and negotiation

#### 5.2.1 Version number form and value

The SSA protocol defines a protocol version number. The version number applies to the XML schema and the request encodings defined in this document. The version number follows IVOA document conventions and contains two non-negative integers, separated by decimal points, in the form “*x.y*”, for example, “1.0”, or “1.13”.

### 5.2.2 Version number changes

The protocol version number shall be changed with each revision of this document. The number shall increase monotonically and shall comprise no more than two integers separated by decimal points, with the first integer being the most significant. There may be gaps in the numerical sequence. Some numbers may denote draft versions. Servers and their clients need not support all defined versions, but shall obey the negotiation rules below.

### 5.2.3 Appearance in requests and in service metadata

The version number shall appear in at least two places: in the service metadata and in the parameter list of client requests to a server. The version number used in a client's request of a particular server shall be equal to a version number which that server has declared it supports (except during negotiation, as described below). A server may support several versions, whose values clients may discover according to the negotiation rules.

### 5.2.4 Version number negotiation

A SSA client may negotiate with a server to determine a mutually agreeable protocol version. Negotiation is performed using the GetCapabilities operation (described in 7.2) according to the following rules.

All service metadata shall include a protocol version number and shall comply with the XML Schema defined for that version. In response to a GetCapabilities request (for which the VERSION parameter is optional) that does not specify a version number, the server shall respond with the highest version it supports. In response to a GetCapabilities request containing a version number that the server implements, the server shall send that version. If the server does not support the requested version, the server shall respond with output that conforms to a version it does support, as determined by the following rules:

- If a version unknown to the server and higher than the lowest supported version is requested, the server shall send the highest version it supports that is less than the requested version.
- If a version lower than any of those known to the server is requested, then the server shall send the lowest version it supports.
- If the client does not support the version sent by the server, it may either cease communicating with the server or send a new request with a different version number that the client does support.

The process may be repeated until a mutually understood version is reached, or until the client determines that it will not or cannot communicate with that particular server.

**EXAMPLE 1** Server understands versions 1, 2, 4, 5 and 8. Client understands versions 1, 3, 4, 6, and 7. Client requests version 7. Server responds with version 5. Client requests

version 4. Server responds with version 4, which the client understands, and the negotiation ends successfully.

EXAMPLE 2 Server understands versions 4, 5 and 8. Client understands version 3. Client requests version 3. Server responds with version 4. Client does not understand that version or any higher version, so negotiation fails and client ceases communication with that server.

The VERSION parameter is mandatory in requests other than GetCapabilities.

## 5.3 General HTTP request rules

### 5.3.1 Introduction

This document defines the implementation of the SSA service on a distributed computing platform (DCP) comprising Internet hosts that support the Hypertext Transfer Protocol (HTTP) (see IETF RFC 2616). Thus, the Online Resource of each operation supported by a server is an HTTP Uniform Resource Locator (URL). The URL may be different for each operation, or the same, at the discretion of the service provider. Each URL shall conform to the description in IETF RFC 2616 (section 3.2.2 “HTTP URL”) but is otherwise implementation-dependent; only the query portion comprising the service request itself is defined by this document.

While the SSA protocol currently only supports HTTP as the DCP for general parameterized operations, data access references are more general and may use other internet protocols, e.g., FTP, or potentially grid protocols.

HTTP supports two request methods: GET and POST. One or both of these methods may be offered by a server, and the use of the Online Resource URL differs in each case. Support for the GET method is mandatory; support for the POST method is optional.

### 5.3.2 Reserved characters in HTTP GET URLs

The URL specification (IETF RFC 2396) reserves particular characters as significant and requires that these be escaped when they might conflict with their defined usage. This document explicitly reserves several of those characters for use in the query portion of SSA requests. When the characters “?”, “&”, “=”, and “;” appear in one of the roles defined in Table 1, they shall appear literally in the URL. When those characters appear elsewhere (for example, in the value of a parameter), they shall be encoded as defined in IETF RFC 2396. The server shall be prepared to decode any character escaped in this manner.

Table 1 — Reserved characters in SSA query string

Character	Reserved usage
?	Separator indicating start of query string.
&	Separator between parameters in query string.
=	Separator between name and value of parameter.

,	Separator between individual values in list-oriented parameters (such as POS, BAND and TIME)
---	--

### 5.3.3 HTTP GET

A SSA service shall support the “GET” method of the HTTP protocol (IETF RFC 2616).

An Online Resource URL intended for HTTP GET requests is in fact only a URL prefix to which additional parameters are appended in order to construct a valid Operation request. A URL prefix is defined in accordance with IETF RFC 2396 as a string including, in order, the scheme (“http” or “https”), Internet Protocol hostname or numeric address, optional port number, path, mandatory question mark “?”, and optional string comprising one or more server-specific parameters ending in an ampersand “&”. The prefix defines the network address to which request messages are to be sent for a particular operation on a particular server. Each operation may have a different prefix. Each prefix is entirely at the discretion of the service provider.

This document defines how to construct a query part that is appended to the URL prefix in order to form a complete request message. Every SSA operation has several mandatory or optional request parameters. Each parameter has a defined name. Each parameter may have one or more legal values, which are either defined by this document or are selected by the client based on service metadata. To formulate the query part of the URL, a client shall append the mandatory request parameters, and any desired optional parameters, as name/value pairs in the form “name=value&” (parameter name, equals sign, parameter value, ampersand). The “&” is a separator between name/value pairs, and is therefore optional after the last pair in the request string.

When the HTTP GET method is used, the client-constructed query part is appended to the URL prefix defined by the server, and the resulting complete URL is invoked as defined by HTTP (IETF RFC 2616).

Table 2 summarizes the components of an operation request URL when HTTP GET is used.

Table 2 — Structure of SSA request using HTTP GET

URL component	Description
<a href="http://host[:port]/path[?{name[=value]&amp;}]">http://host[:port]/path[?{name[=value]&amp;}]</a>	Base-URL (prefix) of service operation. [] denotes 0 or 1 occurrence of an optional part; {} denotes 0 or more occurrences.
name=value&	One or more standard request parameter name/value pairs as defined for each operation by this document.

### **5.3.4 HTTP POST**

SSA does not currently support the “POST” method of the HTTP protocol (IETF RFC 2616), but may do so in the future. POST could be used, for example, to permit large range-lists to be specified.

### **5.4 General HTTP response rules**

Upon receiving a valid request, the server shall send a response corresponding exactly to the request as detailed in Clause 7 of this document, or send a service exception if unable to respond correctly. Only in the case of Version Negotiation (see 6.2.4) may the server offer a differing result. Upon receiving an invalid request, the server shall issue a service exception as described in 6.11.

A server may send an HTTP Redirect message (using HTTP response codes as defined in IETF RFC 2616) to an absolute URL that is different from the valid request URL that was sent by the client. HTTP Redirect causes the client to issue a new HTTP request for the new URL. Several redirects could in theory occur. Practically speaking, the redirect sequence ends when the server responds with a SSA response. The final response shall be a SSA response that corresponds exactly to the original request (or a service exception).

Response objects shall be accompanied by the appropriate Multipurpose Internet Mail Extensions (MIME) type (IETF RFC 2045) for that object. A list of MIME types in common use on the internet is maintained by the Internet Assigned Numbers Authority (IANA) [2]. Allowable types for operation responses and service exceptions are discussed below. The basic structure of a MIME type is a string of the form “type/subtype”. MIME allows additional parameters in a string of the form “type/subtype; param1=value1; param2=value2”. A server may include parameterized MIME types in its list of supported output formats. In addition to any parameterized variants, the server should offer the basic unparameterized version of the format.

Response objects should be accompanied by other HTTP entity headers as appropriate and to the extent possible. In particular, the Expires and Last-Modified headers provide important information for caching; Content-Length may be used by clients to know when data transmission is complete and to efficiently allocate space for results, and Content-Encoding or Content-Transfer-Encoding may be necessary for proper interpretation of the results.

### **5.5 Numeric and Boolean values**

Integer numbers shall be represented in a manner consistent with the specification for integers in XML Schema Datatypes ([8], section 3.3.13). This document shall explicitly indicate where an integer value is mandatory. Real numbers shall be represented in a manner consistent with the specification for double-precision numbers in XML Schema Datatypes ([8], section 3.2.5). This representation allows for integer, decimal and exponential notations. A real value is allowed in all numeric fields defined by this document unless the value is explicitly restricted to integer.

Sexagesimal formatting is not permitted other than in ISO 8601 formatted time strings unless otherwise specified in this document. In particular, astronomical coordinates should be rendered as real numbers as specified above.

Positive, negative and zero values are allowed unless explicitly restricted.

Boolean values shall be represented in a manner consistent with the specification for Boolean in XML Schema Datatypes ([8], section 3.2.2). The values “0” and “false” are equivalent. The values “1” and “true” are equivalent. Absence of an optional value is equivalent to logical false. This document shall explicitly indicate where a Boolean value is mandatory.

## **5.6 Output formats**

The response to a SSA request is always a computer file that is transferred over the Internet from the server to the client. The file may contain text, or the file may be a graphics or FITS-formatted file. As stated in 6.4, the type of the returned file shall be indicated by a MIME type string.

Text output formats are usually formatted as Extensible Markup Language (XML; MIME type text/xml). Text formats are used to convey service metadata, descriptions of error conditions, or responses to data queries. In particular, the response to a data query is always returned as an XML file in VOTable format.

## **5.7 Request parameter rules**

### **5.7.1 Parameter ordering and case**

Parameter names shall not be case sensitive, but parameter values shall be. In this document, parameter names are typically shown in uppercase for typographical clarity, not as a requirement.

Parameters in a request may be specified in any order.

When request parameters are duplicated with conflicting values, the response from the server may be undefined. This document does not mandate which of the duplicated values sent by the client are to be used by the server.

A SSA service shall be prepared to encounter additional request parameters that are not part of this document without reporting an error. In terms of producing results per this document, a SSA service shall not require such parameters, but may define additional service-defined parameters.

### **5.7.2 Range-List Parameters**

Parameters consisting of lists (for example, POS, BAND and TIME in SSA) shall use the comma (“,”) as the separator between items in the list. Additional white space shall not be used to delimit list items. If a list item value includes a space or comma, it shall be escaped using the URL encoding rules (see 6.3.2 and IETF RFC 2396).

In some lists, individual entries may be empty, and shall be represented by the empty string (“”). Thus, two successive commas indicate an empty item, as does a leading comma or a trailing comma. An empty list (“”) shall be interpreted either as a list containing no items or as a list containing a single empty item, depending on context.

Some parameters (for example BAND and TIME) may allow a parameter value to be specified as a range. Such range-valued parameters shall use the forward slash (“/”) character as the separator between elements of the range specification. For example, “5E-7/8E-7” would specify a range consisting of all values from 5E-7 to 8E-7, inclusive. If a third field is specified it is a step size for traversing the indicated range. If a parameter permits a step size the semantics of the step size are defined by the specific parameter.

An open range may be specified by omitting either value. If the first value is omitted the range is open toward lower values. If the second value is omitted the range is open toward higher values. Omitting both values indicates an infinite range which accepts all values. For example, “/5” is an open range which accepts all values less than or equal to 5.

A list may be qualified by appending the character “;” followed by the qualifier string. For example “1E-6/3E-7;source” could specify a spectral bandpass in the rest frame of the source.

List and range syntax may be combined, e.g., to indicate a list of scalar or range-valued parameter values.

## **5.8 Common request parameters**

### **5.8.1 VERSION**

The VERSION parameter specifies the protocol version number. The format of the version number, and version negotiation, are described in 6.2.

### **5.8.2 REQUEST**

The REQUEST parameter indicates which service operation is being invoked. The value shall be the name of one of the operations offered by the server.

### **5.8.3 Extended capabilities and operations**

The SSA service allows for optional extended capabilities and operations. Extensions may be defined within an information community when needed for additional functionality or specialization. A generic client shall not be required or expected to make use of such extensions. Extended capabilities or operations shall be defined by the service metadata. Extended capabilities provide additional metadata about the service, and may or may not enable optional new parameters to be included in operation requests. Extended operations allow additional operations to be defined.

A server shall produce a valid response to the operations defined in this document, even if parameters used by extended capabilities are missing or malformed (*i.e.* the server shall supply a default value for any extended capabilities it defines), or if parameters are supplied that are not known to the server.

Service providers shall choose extension names with care to avoid conflicting with standard metadata fields, parameters and operations.

## 5.9 Service result

The return value of a valid Service request shall correspond to the type requested in the FORMAT parameter. In an HTTP environment, the Content-type header of the response shall be exactly the MIME type given in the valid request.

## 5.10 Service exceptions

Upon receiving a request that is invalid according to this document, the server shall issue a service exception report. The service exception report is meant to describe to the client application or its human user the reason(s) that the request is invalid. The allowed service exception formats are defined for each operation below.

Errors may arise in software modules other than those which implement the SSA service, and may result in exception messages other than those defined by this document. For example, when an error condition occurs in the local computing environment of the SSA service server instance (e.g. out of memory or disk space), the server may be unable to process the SSA service request and may issue an error message. Or, upon receiving a request that is invalid according to the rules of the Distributed Computing Platform in use, the server may issue a service exception of a type valid in that DCP (e.g. if the URL prefix is incorrect, an HTTP 404 status code (IETF RFC 2616) may be sent).

# 6 SSA Concepts

## 6.1 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 the origin of the data:

- **Atlas** data consists of data which attempts to uniformly and comprehensively cover a given parameter space, for example a survey data collection.

- **Pointed** data targets specific astronomical objects, for example, an instrument data collection in which individual datasets are typically observations of specific objects. Sky coverage tends to be nonuniform but the observations may provide high quality data for astronomical objects of interest. Datasets may vary greatly in size.
- **Modeled** data is artificial data constructed from a theoretical model, for example, a library of synthetic spectra.

We can say also 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.
- **Simulated** 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.

### 6.1.1 Types of Services

Just as the characteristics of the data returned by a service can vary, the type of service provided to access the data can also vary.

- A **static** or whole-dataset service is one which returns precomputed, entire archive datasets. No data subsetting or filtering occurs and pixels are not modified. Data model mediation and file-oriented transformations such as a format conversion may however occur. The size of a dataset returned by a static service can vary greatly and may differ greatly from the ROI requested by the client.
- A **cutout** service “cuts out” portions of archival datasets to try to return the ROI (in a multiparameter space) requested by the client. The subset may be made on any physical axis, i.e., spatial (sky), time, or spectral. “Cut out” means that the data may be subsetted, 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.
- A **resampling** service is the same as a cutout service except that the data samples are computed at access time, e.g., by interpolating or reprojecting the data. Level matching, flux correction following scale changes, computation of pixels from event

or visibility data, etc., may take place as part of the pixels are computed. 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 virtual dataset requested by the client, but resampling can degrade the data and may not be desired for certain types of analysis.

A cutout service may also provide the option to return entire datasets, and a resampling service may provide the option to disable resampling and return cutouts or entire datasets.

Although to SSA a 1D spectrum, SED, or time series are similar objects in terms of the core photometric data model and access interface, from the client analysis or data provider perspective they are quite different objects and a given service instance will normally return only one of these types of data objects.

Example 1. In the case of SSA, a survey consisting of numerous 4096-point spectra would probably be most sensibly served up via a static whole-dataset service. A collection of large high resolution spectra, or a collection of lengthy time series with millions of sample points, might be best served via a cutout service. A cutout service could be very useful for retrieval of individual line profiles from high resolution spectra. If a collection is made available via a cutout or resampling service, it should also be available via a static service as well, for services which prefer to manipulate the archival data directly. A single service instance may also provide all these options.

## 6.2 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.

### 6.3 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.

## 7 SSA Operations

### 7.1 Introduction

The operations currently defined by the SSA protocol are **queryData**, **getData**, **stageData**, and **getCapabilities**. Of these, currently only queryData and getCapabilities are implemented as explicit parameterized operations. getData is currently not implemented as an explicit operation, rather it is implicit in the access reference URL used to retrieve datasets.

### 7.2 Methods & Protocols

As with all the DAL interfaces, a SSA service may eventually be defined for multiple distributed computing platforms, e.g., GET and SOAP. 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 data query is formed like this:

```
<Service.BaseURL>?VERSION=1.0&REQUEST=queryData&[POS=ra,dec][&SIZE=diam][  
&<otherArgs>]
```

#### Example:

```
http://www.myorg.org/ssa?VERSION=1.0&REQUEST=queryData&POS=22.438,-  
17.2&SIZE=0.02
```

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

### 7.3 QueryData (required)

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.

### 7.4 Query Parameters

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 minimally-compliant SSA service **must** support at least these four parameters. Various other parameters specific to spectrophotometric data are also defined, and should also be supported by the service if possible, and to be fully compliant.

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.

#### 7.4.1 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	0.1/2.7E-7 (=10cm-2700Å)	m	double(2)	SSA.SpectralBandpass
4	TIME	1998-05-21/1999	ISO 8601 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 in the fundamental physical measurement axes, and the desired client data formats.

##### 7.4.1.1 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 in list format (comma separated) with no embedded white space, as defined in section 5.7.2.

Example: POS=52,-27.8

### 7.4.1.2 SIZE

The diameter of the search region specified 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).

### 7.4.1.3 BAND

The spectral bandpass given in range-list form as defined in section 5.7.2. The units are wavelength in vacuum in units of meters [RSM, Hanisch *et.al*, 2004]. The spectral rest frame **may** optionally be qualified as either “source” or “observer”.

Example: 1E-6/3E-7;source

For most queries the precision with which the spectral bandpass is specified *in the query* 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.

### 7.4.1.4 TIME

The time bandpass (epoch) of the observation in range-list form as defined in section 5.7.2, specified in ISO 8601 format. If the time zone is not specified UTC is assumed.

### 7.4.1.5 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. *[We may still decide to go back to full MIME-types to simplify this parameter.]*

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 COMPLIANT: an external data format
GRAPHIC	Any of the graphics formats: JPEG, PNG, GIF
votable	Shorthand for application/x-votable+xml
fits	Shorthand for application/fits
xml	Shorthand for application/xml

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 parameter names to be specified in a case-insensitive form. By convention reserved parameter names are specified in upper case in this document, but this does not have to be the case when the service is invoked. Parameter values are case-sensitive.

## 7.4.2 Recommended Query Parameters

The following additional parameters **should** be implemented by a fully compliant service:

#	Parameter	Sample value	Unit	Datatype	Utype
1	APERTURE	0.00028 (=1")	dec. deg.	double	SSA.Aperture
2	SPECRES	5D-10 (=5Å)	m	double	SSA.SpectralResolution
3	SPATRES	Xxxx	Dec. deg	double	SSA.SpatialResolution
4	TIMERES	xxxx	s	double	SSA.TimeResolution
5	SNR	5.0	-	double	SSA.MinSNR
6	REDSHIFT	-0.1, 2.0	1	double(2)	SSA.Redshift
8	TARGETNAME	Mars	-	char(*)	SSA.TargetName
7	TARGETCLASS	star	-	char(*)	SSA.TargetClass
8	PUBID	ADS/col#R5983	-	char(*)	SSA.PublisherCreatorID
9	CREATORID	Ivo://auth/col#R1234	-	char(*)	SSA.CreatorCreatorID
10	COLLECTION	DSS2	-	char(*)	SSA.Collection
11	TOP	20	-	int	SSA.MaxTopRankedRecords
12	INDEX	50	Rows	Int	SSA.QueryIndex
13	COMPRESS	(TRUE)	-	boolean	SSA.Compress
14	SINCE	1995-04-15	ISO 8601	char(*)	SSA.Since
15	RUNID	<string>	-	char(*)	SSA.RUNID

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

### 7.4.2.1 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 7.5.5.1).

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.

### 7.4.2.2 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.

### 7.4.2.3 SPATRES

The minimum spatial resolution in decimal degrees.

### 7.4.2.4 TIMERES

The minimum time resolution, specified in seconds.

### 7.4.2.5 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.

### 7.4.2.6 REDSHIFT

A photometric (observed) redshift range specified as a range-list as defined in section 5.7.2. 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

#### 7.4.2.7 TARGETNAME

The target name, suitable for input to a name resolver. In general target name resolution should take place on the client side, where greater options are possible,

#### 7.4.2.8 TARGETCLASS

A comma delimited list of strings denoting the types of astronomical objects to be searched for. At the moment there is no standard classification for astronomical objects but it is suggested to use the “condensed” names from the list at <http://simbad.u-strasbg.fr/guide/chF.htx>.

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

#### 7.4.2.9 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`

#### 7.4.2.10 PUBID

The IVOA dataset identifier, assigned by the publisher of a dataset. The same dataset published in different places may have a different PUBID assigned by each publisher. ADS dataset identifiers are an example of a PUBID.

#### 7.4.2.11 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`.

#### 7.4.2.12 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.

#### 7.4.2.13 INDEX

The index into the result set of the query at which to return results. This is used to step through the result set for queries which are too large to be returned in a single operation. The value is a range-list containing two numbers as defined in section 5.7.2. The first value is the index of the first query result record to be returned, starting from zero, and the second value is the number of records to be returned. *[Whether this is sufficient is still a topic for discussion.]*

#### 7.4.2.14 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.

#### 7.4.2.15 SINCE

Find only datasets modified or created since the given date, specified in ISO 8601 format. Note this is not the same thing as `TIME`, which refers to time of observation.

### 7.4.3 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.

## 7.5 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 7.6.

### 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.

## 7.5.1 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)

### 7.5.1.1 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 7.4.2.12.

### 7.5.1.2 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.

### 7.5.1.3 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.

## 7.5.2 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	☑	Dataset.Type	'spectrum'	Spectrum, TimeSeries, SED, etc.
2	●	Dataset.DataModel	'SSA-V1.0'	data model of dataset
3	●	Dataset.Title	'NGC 346'	brief descriptive title of dataset
4	☑	Dataset.SSA.NSamples	4096	number of samples in dataset
5	●	Dataset.SSA.Aperture	0.053	aperture diameter (decimal degrees)
6	○	Dataset.SSA.TimeAxis	'time'	timeCoord axis (foreign data)
7	○	Dataset.SSA.SpectralAxis	'wave'	spectralCoord axis (foreign data)
8	○	Dataset.SSA.FluxAxis	'flux'	flux axis (foreign data)
9	●	Dataset.CreationType	'atlas'	atlas, pointed, cutout, resampled
10	○	Dataset.Derivation	'observed'	observed, composite, simulated, synthetic

### 7.5.2.1 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.*

### 7.5.2.2 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'.

### 7.5.2.3 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.

### 7.5.2.4 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.

### 7.5.2.5 Dataset Creation Type

The creation type of the dataset, defining how the data was computed, as defined in section 6.1. Allowable values are 'atlas', 'pointed', 'cutout', and 'resampled'.

### 7.5.2.6 Dataset Derivation

The derivation or source of the dataset, as defined in section 6.1. Allowable values are 'observed', 'composite', 'simulated', and 'synthetic'.

## 7.5.3 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	○	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.*

## 7.5.4 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 **Error! Reference source not found.**, 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.

## 7.5.5 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.

### 7.5.5.1 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.

### 7.5.5.2 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.

### 7.5.5.3 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.

### 7.5.6 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	<input type="radio"/>	Instrument.Name	instrument name (identifier)
2	<input type="radio"/>	Instrument.Exposure	exposure time in seconds
3	<input type="radio"/>	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).

### 7.5.7 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	<input checked="" type="checkbox"/>	Access.Reference	<a href="http://myorg.org/get?oid=A134">http://myorg.org/get?oid=A134</a>	dataset access URL
2	<input checked="" type="checkbox"/>	Access.Format	application/x-votable+xml	MIME type of dataset
3	<input checked="" type="checkbox"/>	Access.Encoding	x-gzip	encoding; compression
4	<input checked="" type="checkbox"/>	Access.Size	185000	approximate dataset size

#### 7.5.7.1 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.

#### 7.5.7.2 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 7.5.2.2.

A single data object may be available in multiple file formats. In such a case a *logical name* (section 7.5.1.2) should be given to indicate that the different formats all correspond to the same data object.

### 7.5.7.3 Encoding

This parameter corresponds to the value of `Content-Encoding` in the case of HTTP GET. When the `COMPRESS` query parameter is used it **should** be used to describe the compression method, for example, `x-gzip`.

### 7.5.7.4 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.

## 7.5.8 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`.

## 7.6 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 7.5; although, it is not an error to do so.

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

### 7.6.1 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.

## 7.6.2 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 8.4).

### 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>
```

## 8 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.

### 8.1 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.

## 8.2 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

## 8.3 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.

## 8.4 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

## 9 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.

## 9.1 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 7.4)
- possible output columns (7.5)

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 7 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 7.4.2), and non-standard parameters specific to the service (section 7.4.3). 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.

## 9.2 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.

## 9.3 Service Registration

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

### 9.3.1 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.

### 9.3.2 Service Type

The SERVICE.TYPE in the registry corresponds to the supported SEGMENTYPES. A service **may** support PHOTOMETRY points and/or SPECTRUM and/or TIMESERIES. If more than one SEGMENTYPE 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" 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"/>
        <PARAM name="dataset derivation" datatype="char" arraysize="*"
          utype="sdm:SSA.Dataset.Derivation" value="observed"/>
        <FIELDref ref="nsamples"/>
      </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="locationBand"/>
          <FIELDref ref="locationTime"/>
        </GROUP>
        <GROUP utype="sdm:SSA.Coverage.Bounds">
          <FIELDref ref="boundsTime"/>
        </GROUP>
      </GROUP>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

```

    <FIELDref ref="boundsSpectral"/>
  </GROUP>
</GROUP>
<PARAM name="creator" datatype="char" arraysize="*"
  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" 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="target name" datatype="char" arraysize="*"
  utype="sdm:SSA.Target.Name"/>
<FIELD name="number of samples" datatype="int" ID="nsamples"
  ucd="meta.number" utype="sdm:SSA.Dataset.SSA.NSamples"/>
<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"/>
<FIELD name="Bandpass Characteristic Value" datatype="float"
  ID="locationSpectral" ucd="instr.bandpass"
  utype="sdm:SSA.Coverage.Location.Spectral" 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>2125.0</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>1548.8</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="nsamples"/>
        <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="Bandpass Characteristic Value"
            datatype="float" utype="sdm:SSA.Coverage.Location.Spectral"
            ucd="instr.bandpass" unit="MHz" value="1600"/>
          <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="nsamples"
    ucd="meta.number" utype="sdm:SSA.Dataset.SSA.NSamples"/>
<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>
      <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.nao.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>