IVOA Support Interfaces
Version 0.22

IVOA Working Draft 2005 May 11

This version:
0.22 http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices/VOSupportInterfaces-0.22.pdf

Previous versions:
0.21 http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices/VOSupportInterfaces-0.21.pdf
0.2 http://www.ivoa.net/internal/IVOA/VOSupportInterfaces-0.2.pdf
0.1 http://www.ivoa.net/internal/IVOA/StandardInterfaces-0.1.pdf

Editors:
William O'Mullane, Guy Rixon, Ani Thakar

Authors:
Web and Grid Services Working Group

Please send comments to: mailto:grid@ivoa.net

Abstract

This document describes the minimum required interface to participate in the IVOA as a web service.

Status of this document

This is a Working Draft. There are no prior released versions of this document.

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

Acknowledgments

This work is based on discussions and actions from the 2003 IVOA meeting in Strasbourg and further discussions on registry functionality at JHU late in 2003. The latest inputs came from a local meeting at JHU Sept. 2004.

Contents

1 Introduction

Web Services form an increasing part of the Virtual Observatory. It is felt there are some basic Interfaces that these services should provide. The word “interface” is used here in the SOAP sense - this has been referred to as a “port” in older SOAP documentation and perhaps still in the Grid community.

Hence this document attempts to define the standard interfaces that a service should provide.The VO also has many non-SOAP services that would benefit from standard interfaces. A method is also described for invoking these services in a CGI manner.
A new document is also expected which will give a VO-Profile for our services. This will tie in with WS-I specifications.

1.1 About the document

In the normal requirements manner the words “should” and “shall” are used to convey the level of necessity of the interface. Each interface is clearly given a short description and a requirement number of the form SI-N where N is a running number in the document..

2 Standard VO Interfaces

2.1 Registration MetaData

The service should be the single authoritative source for metadata. WEB services provide WSDL in a standard manner. WSDL alone is insufficient for the purposes of the VO. We need something more like the VOResource implementation of the RM (Resource Metadata) document[RM]. In a web service we would return the object type produced from the XSD for the RSM. The definition of VOResource must be included in the WSDL for the service. This does not preclude returning an extension of the VOResource.

The registry remains the place to go to find metadata but a standard service like this would allow us to build a registry where we just input our service URL and the registry then asks for and validates the metadata. It could even periodically check whether it is up to date.

SI-1 All VO services should implement the "getRegistration" interface. This shall return a valid VOResource document describing the metadata of this service. The owner will have to already get an authorityID and resourceKey from a provider for this.

SI-2 All Vo services should implement the "RegistrationChangedOn" interface. This shall return the date the metadata last changed.

2.2 Availability

The heartbeat interface is to tell us if the service is in operation. It should do a good check on the underlying service to see if it is still operational and not just be a simple return from a web server, e.g., if it relies on a database it should check that the database is still up.
Ultimately some portals may track these heartbeats and compile uptime statistics. With the location we could have 3D global maps of services and availability.

SI-3 All VO services shall implement the "getAvailability" interface. This shall return an XML document containing :

- Uptime - the up time of the service
- ValidTo - how long it believes this will be valid i.e. next scheduled downtime.
- ContactDetails – Name, Email and PhoneNumber of a person to contact if there is a problem.

2.3 Harvesting Logs

For a global Virtual Observatory, the ability to correlate global logs will be quite informative and beneficial. For example if we have multiple registries it would be nice to be able to have an idea of number of requests on each one and where they are coming from. While the mechanics and logistics of harvesting worldwide logs may be a gory topic, in order to make this even feasible it is necessary to have some rudimentary interface for collecting and harvesting logged information in some standard (preferably tabular) format.We envisage at least two types of logs that will need to be harvested in the VO. The standard web log based on W3C logging is one. More complex services may have distributed application servers running behind the web server – such services will

undoubtedly be able to provide significantly more detailed logging. Hence we propose a mandatory web log and optional Service log system.
Unlike the above ports, these would probably exist on a separate endpoint. In the registry, we would need a new relationship attribute - “isLogHarvester”. In this way a bunch of Cone or SIA services could provide logging through a single Web service that outputs their weblogs.
This interface would not necessarily be publicly available. It would be fine for providers to IP restrict access to the service. The VO aggregator would then just produce statistical reports i.e. clientIPs would never become publicly available

2.3.1 RunID

This is really outside the scope of this document, but we need a mechanism whereby a portal creates a run id in the form *serverName*:*sequenceNumber*. This id should be passed to all services called such that it is logged for each service call. Any service calling another service should pass this number on. Hence if DataScope calls 300 services from one user request they should all have the same RunID. For a web Service the run id could be in the header i.e. no change to the interface. For the Simple GET services it can simply be appended as &RunID=server:990990 which will automatically put it in the log.

2.3.2 Harvest Web Log

All web servers perform logging usually in w3c form [LOG]. A subset of this would be useful.

SI-4 All VO services should implement the “HarvestWebLog” interface. This interface should take three parameters “fromDate”, “toDate”, and “format”, and should return a URL that points to a file containing a serialization of a set of WebLogEntry in the chosen format. A WebLogEntry would contain the following information (fields that must be filled are marked with *):

- Date* - date and time of request in UT, ISO 8601 format
- ClientIP* - Address request came from
- ServerUrl* - in case server handles multiple websites
- Method* - post,get
- QueryString* - whats after ? in url – perhaps blank.
- Browser* - browser which submitted request
- Status* - http status code(500, 404 ..)
- bytes - number of bytes returned
- time-taken* - time to complete transaction in seconds

2.3.3 Harvest Service Log

Many services perform some extra logging with more refinement in internal server names and times to service requests with elapsed time available etc.

SI-5 All VO services should implement the “HarvestServiceLog” interface. This interface should take three parameters “fromDate”, “toDate” and “format”, and should return a URL that points to a file containing a serialization of a set of ServiceLogEntry in the chosen format. A ServiceLogEntry would contain the following information (fields that must be filled are marked with *):

- ClientIP* - Address request came from
- Server* - Server this ran on
- ClientAccessLevel - level of access one of public (anonymous) private(logged in) ,internal (privileged local usage)
- Userid - can be null may be provided if available and site wishes to do so
- Rund* - The one passed in if it was passed in – otherwise generated
- Request* - What was asked for .
- Event - What that means internally, e.g., Query, Harvest, Update etc.
- TimeIn* - Time request was received.
- TimeOut* - Time request was serviced.
- TimeActual* - Processor time waits removed.
- Volume - ~Size in bytes of the response
- Status* - 0 for pending, 1 for done, 2 for failed
- ErrorCode* - Numerical error code
- Response*- Error message if this failed, perhaps the actual response if it is a small response, otherwise some summary statement or blank. Mainly to have errors.
- IsVisible - 1 for public entries, 0 for private, default 1; this field is necessary because the host for a given service may want some log entries to be kept private and not displayed in the VO logs. However these entries should still be recorded in the logs for completeness and statistical purposes.

3 Changes from previous versions

Modified HarvestWebLog and HarvestServiceLog to return a URL instead of sets of log entries. Marked mandatory fields in each log and made minor changes to text. Added WSDL.

4 References

[RM] Resource Metadata for the Virtual Observatory; Bob Hanisch et al.;
http://www.ivoa.net/Documents/WD/ResMetadata/WD-ResMetadata.html
[LOG] W3C Logging ; http://www.w3.org/Daemon/User/Config/Logging.html and
http://www.w3.org/TR/WD-logfile.html

5 Appendix 1

5.1 Link to WSDL