IVOA VOStore
Version 0.12

IVOA Working Draft
2004-09-21

This version:

0.13 http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices**/VOStore0.13.pdf**

Previous versions:

Editors:

Dave Morris, William O'Mullane, Guy Rixon.

Authors:

IVOA Web and Grid Services Working group

Please send comments to: webservices@ivoa.net

Abstract

This document describes the VOStore concept

Status of this document

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

Acknowledgments

This work is based on discussions at various IVOA meetings and continuing emails on the mailing list.

Contents

1 Introduction

The initial and strong driver for VOSpace remains the integration of various technologies which allow users to store data through a web based service. The two in the Virtual Observatory realm which we would particularly like to integrate are MYDB[1] and MYSpace[7]. The Storage Resource Broker[9] has additional interfaces that support access through web services, web browsers, Java, Python, Perl, DSpace, OAI, shell commands, C library, Windows browsers, OAI, etc. Large scale analyses will require access through high-performance interfaces[8]. Should we also be considering LDAP[3] or WS-Transfer [10].

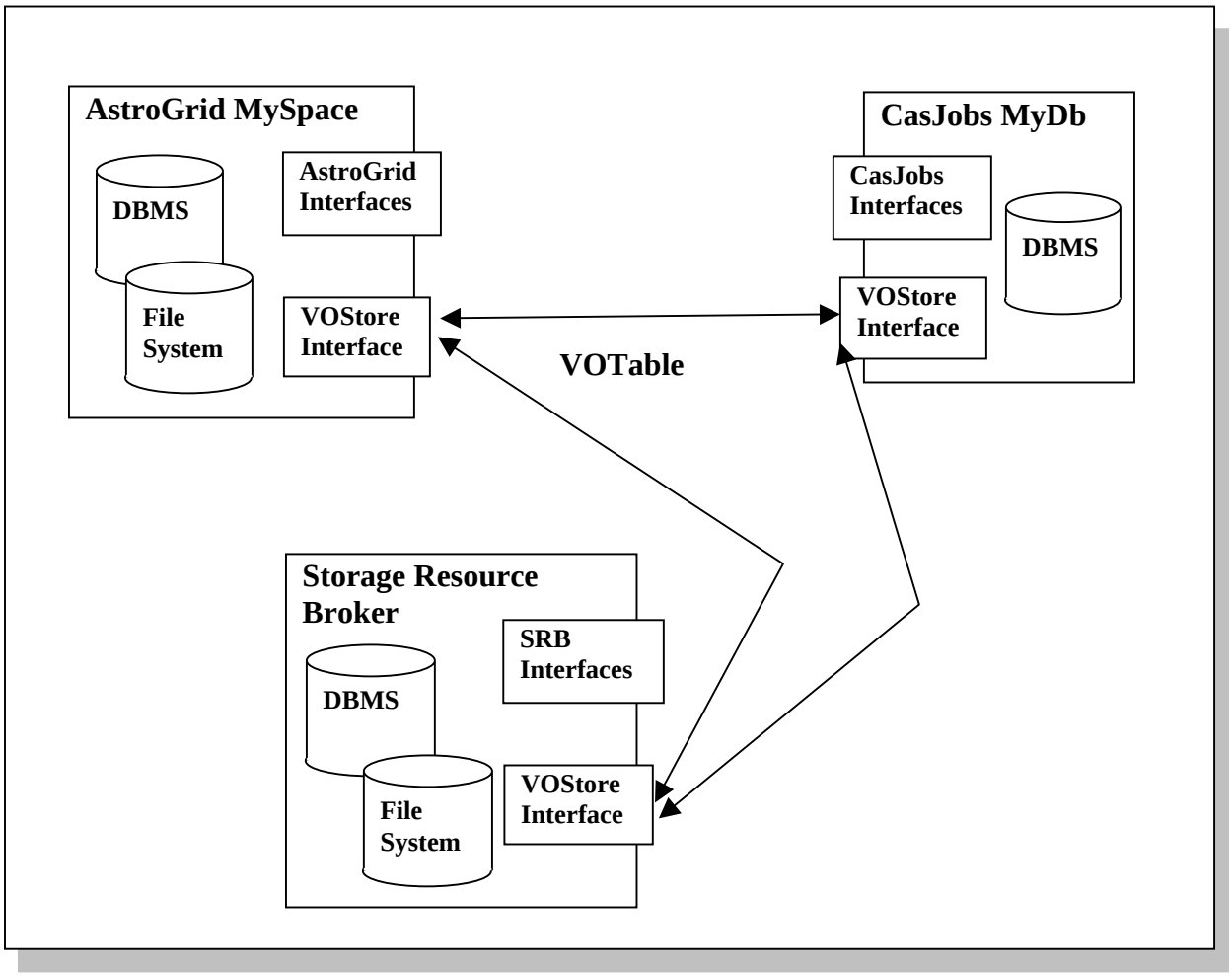
2 Goals of VOStore

The main goal of VOStore is to provide a uniform interface to existing or new data storage locations. Figure 1 below depicts the VOStore interface being implemented on three different systems, which currently exist in the Virtual Observatory realm. The notion is to keep the interface simple so that it should be easy to implement beside existing interfaces. Each of these systems have the capabilities to accept and transmit files – however each does it in a slightly different way. VOStore would define a common interface which should be relatively easy to implement on such systems hopefully it

would simply be a ‘facade’[1] in terms of programming patterns. Basically VOStore probably just needs to support get and put for a file or dataset. The particular transport mechanism for the data may be brokered between the stores e.g. if both support gridftp that could be used.

Figure 1. Example VOStore interfaces in VOSpace

The different VOStore providers would then, of course, be put in the registry where they could be looked up. We would refer to a Store by some logical name, specifically an

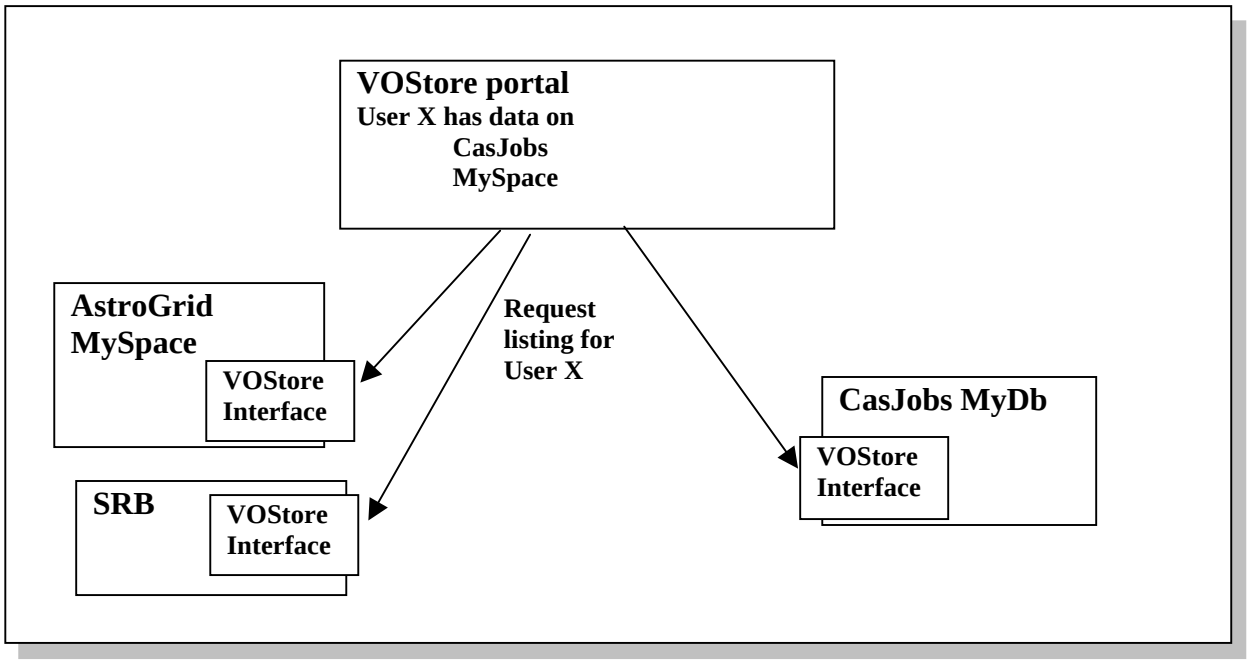


IVOA Identifier[5]. Access to the store would require authentication, nominally we would use the Single Sign On[6] approach emerging in the Web and Grid services Working group. This would certainly imply he the use WS-Security.

Logically one would want a portal to go to which gave an overview of the files and database tables available on the multiple VOStore resources. Such a portal could query each VOStore, looked up in the registry, for data for a given user. This implies an interface to query for allocations in a store based on a given user . The portal could cache such information and keep it fresh. We may wish to consider a version of VOStore which notified portals of allocations to users. WS-Notification may provide a way to do this but

initially this is a complication we should probably avoid. The portal itself may not need a specification.

Figure 2. Example of VOSpace portal concept. The portal queries 3 resources looked up from the registry the user has data on only two of those.



3 VOStore

VOS-1 VOStore shall implement the mandatory interfaces defined in the VO Support Services specification[2].

VOS-2 VOStore shall be defined in terms of a SOAP Service with a WSDL for the precise definition of the interface.

VOS-3 VOStore shall support as a minimum VOTable for tabular transmission and acceptance of data

VOS-4 VOStore shall support the “formats” interface. This shall return the list of formats supported by the service. The Formats shall be in the form strings containing mime types.

VOS-5 VOStore shall support the “transports” interface. This shall return the list of transports supported by the service. These will take the form of strings/ All services shall support SOAP-ATTACHMENT.

VOS-6 VOStore shall support the “Get” interface. This interface shall take as arguments the identifier of the dataset on the server, the required format (one of those listed in VOS-4) and the required transport mechanism (one of those listed in VOS-5)

VOS-7 VOStore shall support the “Put” interface. This interface shall take as arguments the identifier of the dataset required on the server, the upload format (one of those listed in

VOS-4) and the transport mechanism which will be used to send the data (one of those listed in VOS-5)

VOS-8 VOStore shall support the "List" interface to list resources per user and for the entire server.

VOS-9 VOStore shall support the "Rename" interface which takes the identifier of the object to be renames and the new identifier to give it.

VOS-10 VOStore shall support the "Delete" interface which takes the identifier of the object to be deleted.

4 References

[1] Gamma E. et Al. Design Patterns ; Addison Wesley 1995.

[2] IVOA Grid & Web Services Working Group; VO Support Services; http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices/VOSupportInterfaces-0.2.pdf

[3] Lightweight Directory Access protocol; http://www.activexperts.com/activmonitor/functions/ldap/rfc1777/

[4] O'Mullane W. et. Al., Batch Query System with Interactive Local Storage for SDSS and the VO ; ADASS XII 2003.

[5] Plante R. et Al.. IVOA Identifiers, http://www.ivoa.net/Documents/latest/IDs.html

[6] Rixon G., Single Sign On, http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices/SSO-msg-protocol.html

[7] Walton N.A. et. Al., AstroGrid: Initial Deployment of the UK's Virtual Observatory; ADASS XII 2003.

[8] Moore, R., "Operations for Access, Management, and Transport at Remote Sites," Global Grid Forum, December 2003.

[9] Rajasekar, A.,M. Wan, R. Moore, "mySRB and SRB, Components of a Data Grid", 11th High Performance Distributed Computing conference, Edinburgh, Scotland, July 2002.

[10] WS Transfer; http://xml.coverpages.org/WS-transfer200409.pdf

Appendix A: SRB data and metadata access commands

A VOStore interface can provide multiple access mechanisms. The proposed WSDL service is useful for retrieval of small amounts of data. For large-scale analyses, additional interfaces are useful for bulk data access (retrieval of tens of thousands of files), parallel data access (use of parallel I/O streams to move large files), and data subsetting. The additional data movement commands may need to support interactions with firewalls and require both server-initiated and client-initiated parallel I/O stream invocation.

A VOStore can manage a collection, with operations defined for collection access, collection management, and data transport. The set of commands currently implemented in the SRB data grid are listed below. The VOStore could provide equivalent functionality – it would be interesting to decide the minimum set.

- Sannotate - provides a facility for inserting, deleting, updating, and accessing annotations on data objects
- Sappend - appends a local file or a SRB object to a target SRB object.
- Sattrs - lists the queriable MCAT attributes.
- Sbkupsrb - synchronizes copies of an SRB object across replicas
- Sbload - imports in bulk one or more local files and/or directories into SRB space
- Sbregister - registers in bulk one or more local files and/or directories into SRB space.
- Sbunload - exports in bulk a collection or a SRB container to local file system
- Scat - concatenate and display files read from SRB space
- Scd - change working SRB collection
- Schdefres - change default storage resource for the current session at the terminal.
- Schhost - change default connection host for the current session at the terminal.
- Schksum - Checksum SRB data files and/or collections.
- Schmod - changes permission modes and ownership for objects and collections in SRB space
- Scp - copies a srbObj or srbCollection in SRB space
- SdelValue - Deletes a SRB entry (user, physical resource, location)
- Senv - Displays SRB environmental file content
- Serror - describes SRB errors.
- Sexit - Ends a SRB session
- Sget - exports one or more objects from SRB space into the local file system
- SgetColl - display information about SRB data objects.
- SgetD - display information about SRB data objects.
- SgetR - display information about SRB resource(s).
- SgetT - display information about SRB tickets
- SgetU - display information about SRB user(s).
- Shelp - Displays a list of all available SRB S-commands
- Singestuser - adds an account to a SRB grid.
- Sinit - initializes SRB client environment.
- Sln - make links to srbObjects or to srbCollections.

- Sls - display objects and sub-collections in a SRB collection
- Slscont - list your SRB containers
- Smeta - modifies metadata information about SRB data objects.
- Smkcont - creates a new SRB container
- Smkdir - creates a new SRB collection
- SmodColl - modifies metadata information about a SRB collection.
- SmodD - modifies metadata information about SRB data objects.
- SmodifyUser - Modify user account settings.
- SmodR - modifies metadata information about a SRB resource.
- Smv - changes the collection for objects in SRB space
- Spasswd - changes password of current user
- Spcommand - Proxy command operation. Request a remote SRB server to execute arbitrary commands on behalf of client.
- Sphymove - moves an SRB object to a new resource
- Spullmeta - accesses metadata in bulk from a SRB.
- Spushmeta - ingests metadata in bulk into a SRB.
- Sput - imports one or more local files and/or directories into SRB space.
- Spwd - displays current working SRB collection
- Sregister - registers one or more files into SRB space.
- Sregisterlocation - register a new location (computer)
- Sregisterresource - register a new physical resource
- Sreplcont - Replicate a container copy.
- Sreplicate - replicates an SRB object
- Srm - Remove files from SRB space
- Srmcont - Remove a SRB container
- Srmdir - deletes a SRB collection
- Srmticket - remove a ticket
- Srmtrash - Purge files and collections in the "trash" directories.
- Srsync - Synchronize the data between a local copy (local file system) and the copy stored in SRB or between two SRB copies.
- Ssh - Invokes the SRB shell.
- Ssyncd - synchronizes copies of an SRB object

- Ssyncont - Synchronizes the "permanent" copy of the container with the "cache" copies.
- Stcat - display files read from SRB space for a ticketuser
- Sticket - issue tickets for access to SRB objects and collections.
- Stls - display objects and sub-collections in SRB collection for a given ticket
- Stoken - display information about SRB native types (data types, resource types, user types, domain, location, action, access constraints, zones, resource class)
- Sufmeta - modifies attribute-value-unit triplets of user-defined metadata information about SRB data objects.
- Szone - modifies metadata information about SRB data objects.