IVOA SkyNode Interface
Version 0.6

IVOA Working Draft
2003-10-30

This version:
0.6 http://skyservice.pha.jhu.edu/develop/vo/adql/SkyNodeInterface-0.6.pdf

Previous versions:
0.5 http://skyservice.pha.jhu.edu/develop/vo/adql/SkyNodeInterface-0.5.pdf
0.4 http://skyservice.pha.jhu.edu/develop/vo/adql/SkyNodeInterface--0.4.pdf
0.3 http://skyservice.pha.jhu.edu/develop/vo/adql/QueryInterface-2003Aug.pdf
0.2 http://skyservice.pha.jhu.edu/develop/vo/adql/QueryInterface-2003July.pdf

Editors:
William O'Mullane

Authors:
IVOA VOQL Working group

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

Abstract

This document describes the minimum required interface to participate in the IVOA as a queryable VONode as well as requirements to be a Full OpenSkyNode, part of the OpenSkyQuery Portal.

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

At the Cambridge IVOA meeting there was general agreement to the ADQL(Astronomical Data Query Language) concept. This implied an SQL like language but passed as an XML document. VOQL (Virtual Observatory Query Language) was seen as a higher layer built on ADQL and web services. ADQL is defined in [ADQL].

We have now demonstrated that a single WSDL (Web Services Definition Language) file may be implemented in at least Java and C#, i.e. on Linux as well as windows. Also we

have demonstrated that a single client may interact with both implementations of the service.
In this document we would like to propose the astronomical data query interface for the virtual observatory. It is presented here as a set of Services which we believe any Virtual Observatory Service should implement and the specific services a Query Interface should implement. Any IVO site implementing the Query Interface would be considered a "Basic SkyNode".

1.1 OpenSkyQuery

SkyQuery (www.skyquery.net) is a successful implantation of distributed astronomical query system. This is implemetented using some proprietary Microsoft Technology but for the most part uses SOAP Services. Several people have expressed in interest in extending SkyQuery and making it more open. Currently to be a SkyNode one has to implement a set of web interfaces some of which would require .NET technology. The original prototype has proved the idea but now it is time to bring this in line with VO efforts and to make it properly platform independent.
We would like to Open up the SkyQuery protocol to enable other databases and servers to become "Full SkyNodes". These Nodes could then be utilized by a new implementation of the SkyQuery Portal (described in section 8 below).

Finally we include a link the WSDL for the proposed services. It is not included as an appendix here as it comes to some 25 pages and is really meant for use with SOAP tools rather than for human consumption.

2 Open SkyQuery Portal, IVOASkyNode, ADQL and VOQL – brief architectural discussion.

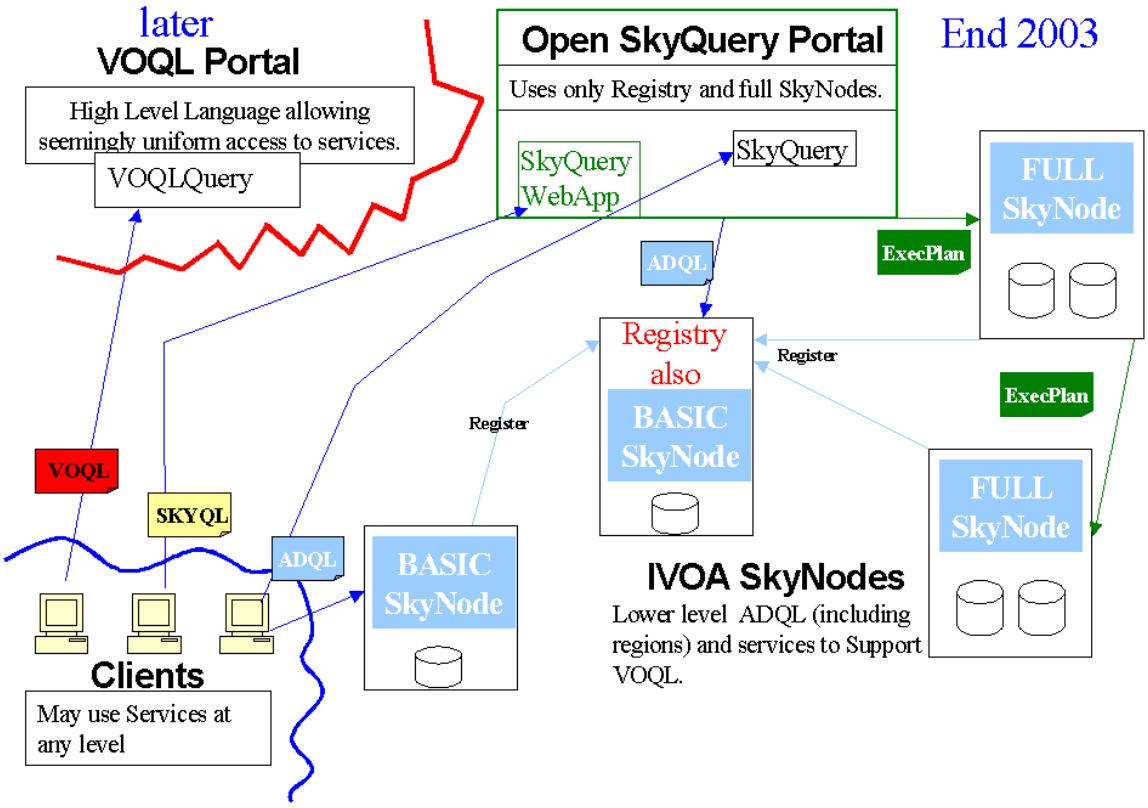

Figure 1. Architecture

The Astronomical Data Query Language (ADQL) is an XML document format for transported queries to IVOA SkyNodes. Different SkyNodes may not support all features of the Language (see the Matrix below in section 7.1). Hence ADQL would be passed from the SkyQuery Portal to the SkyNodes or it may come directly from a client or the VOQL portal. All nodes and the portals should be accessible via SOAP services. The boxes on the nodes in Figure 1 represent SOAP methods.

Additionally for the Open SkyQuery Portal some form of string based query like the current SkyQL (www.skyquery.net) would be accepted. A parser would easily convert this to ADQL i.e. SKYQL would have the same semantics as ADQL but the syntax would be an SQL like string rather than XML. The Portal would use the registry to resolve server names for LEV3 and LEV4 servers and make the Execution Plan (see Section 6 below) and pass it on to the first node in the plan setting up an execution chain as in the current SkyQuery portal..

Finally the Virtual Observatory Query Language(VOQL) is an ambitious language at a higher level than ADQL. A VOQL portal would take VOQL programs. This would need all the work of the SkyQuery portal and more to make it function.

In summary we may see 3 layers of VOQL :

- VOQL1 WebServices : ADQL and VOTABLE to exchange information between machines
- VOQL2 Federation : SQL-like query language and federation system i.e. combination of SkyQuery , JVOQL and VO standards.
- VOQL3 SkyXQuery: future XML-based query language.

3 Error Handling

All errors in the services should be returned as SOAP exceptions. There is no perceived need at this time to have special Exceptions although this may become useful in the future. In the initial version then all Exceptions may simply be a useful message in a generic Exception.

4 Standard VO Interface

For WebServices the service should be the single authoritative source for metadata. Currently 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 RSM (Resource & Metadata) document. 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. This may be seen more clearly by looking at the dummy service http://skyservice.pha.jhu.edu/devel/skynode/skynode.asmx

QI-1 All VO WebServices shall implement the "MetaData" port. This shall return a valid VOResource describing the metadata of this service.

Note: The metadata for the tables /data in the system are accessed as described below.

QI-2 All VO WebServices shall implement the "Heartbeat" port. This shall return an xml document contiaining the up time of the service and how long it believes this will be valid.

5 IVOA SkyNode Interface

5.1 Feature Matrix

The following matrix tries to categorize features and SkyNode/ADQL Levels.
Basic SkyNode here is the minimum IVOA SkyNode Interface – this is useful in itself as it allows one to send queries to a system using ADQL. This is also just one step up from cone search. A matrix has been used as any feature on their own may be useful i.e. a node which can do XMATCH is already useful even if it may not participate in the portal because it lacks other features.
It is assumed large surveys would be Full SkyNodes as footprint services should make queries more efficient on the large surveys.
This table is meant as a summary, more explicit requirements are detailed below.

Feature	Basic SkyNode	Full SkyNode	Optional
ADQL – circle	X	X	
Functions	X	X	

Xmatch		X	
Performance Query		X	
Takes exec plan		X	
Footprint – Region intersect		X	
MYDB			X
Authentication – must have with MYDB			X

5.2 IVOA SkyNode Interface

The skynode interface requires several methods to work. Several of these methods are for Metadata – in this case we need to know about Tables and Columns that may be included in a query. To facilitate interoperation we will also need to be able to get UCDs for each column. ADQL also exposes standard ways for querying metadata but it seems sensible to have explicit WebService calls for this also.
Another service will need to take an ADQL document and return a VOTable of data. The current incarnation of ADQL syntax supports only queries which would return data i.e. there is no provision for Data Manipulation Language which would have no data to return.
Several levels are discussed in the matrix above – here the requirements are simply split between Level 1 (minimum) and the rest.

5.2.1 Basic Sky Node

QI-3 SkyNodes shall register with the registry with type="OpenSkyNode"
We need to add a new SubClass of VODescription for this – there will be other metadata we need which is specific to OpenSkyNode i.e.

- SkyNodeType : Basic or Full
- AcceptsUCDs : True or False
- Xmatch : True or False
- Regions : True or False

Sepcifying AcceptsUCDs means theat a quewry may be specified using UCDs – this is not a requirement and shall probably not become one, specifying the metadata simply allows experimentation with this idea. UCDs do not carry unit information so although it makes sens to use a UCD to request a column it is unclear if this is useful in specifying the predicate of a query – here one would need to know th unit and type of the native SkyNode column to specify the query correctly.

QI-4 SkyNodes shall implement the "Tables" port, which returns a list of all tables that may be used in ADQL passed to this endpoint.

QI-5 SkyNodes shall implement the "Columns" port, which returns information about the columns of a given table name. This shall return an array of MetaColums this structure shall contain the Column Name, Type (as in SQL type), Unit, Precision (number of significant digits), Description, and UCD.

QI-6 SkyNodes shall implement the "Formats" port. This takes no parameters and returns a list of formats which this Node supports for Query Results. Hence this may return VOTable,DataSet,ASCII.

QI-7 SkyNodes shall implement the “Functions” port, which returns an array of MetaFunctions this structure shall contains the function name parameter list and comments for all extra functions this node supports. This also implies a new ADQL subclass which contains these functions (some technical work needed here).

QI-8 SkyNodes shall implement the "PerformQuery" port, which takes an XML document including an ADQL query and an optional format request (as listed from "Formats") as parameters, and returns a document including an appropriate IVOAResult, which is the result of processing the query. IVOAResult shall be an abstract class with multiple subclasses, one for each format. This will be tied down in the WSDL file.SkyNodes shall support at least VOTable format.

QI-9 SkyNodes should accept the VOQL equivalent of Standard SQL-92 Metadata [SQL92] queries . These queries include (expressed here as SQL but would be ADQL/XML coming to the node):

```
Select * from tables
Select * from colums
```

We need to decide how much of the entire set is relevant and required. Since most Dbs handle this we could just say all of it. The syntax may be a little different in each database i.e. in SQL-Server this is

```
select * from information_schema.tables
```

While ORACLE does not support Information Schema (as far as the author can tell).

5.2.2 Full SkyNode requirements

QI-10 SkyNodes shall implement QueryCost() port which takes a simple ADQL query to return the object density per square degree for a set of criteria.

A particular survey with uniform density may choose to not run this as a full query but rather perform the query on a much smaller area or use statistics or heuristics. Basically this does not have to be 100% accurate it is an indication only. Better that it completes quickly.
On the other hand, theoretically, in some databases if the full investigative query were run then it will cause the data for the real query to be cached.

QI-11 SkyNodes shall accept complex Shapes in their queries as defined in the ADQL syntax (using the region spec).

QI-12 SkyNodes shall be able to perform cross matching between their survey and table of data provided in VOTABLE format. The method XMatch() takes an ADQL structure and a set VOTables.

An example of the SkyQL (which would be expected in ADQL format):

```
select ...
  from SDSS:PhotoObj o, EXT:0 my1, EXT:1 my2
  where XMatch(o,my1,my2) < 3
```

Note that this allows you to cross match VOTables without referring to any database table, i.e. it is a general cross matching tool.

Description of XMatch

The cross-matching algorithm is a probabilistic calculation that minimizes the chisquare parameter as defined by:

$$\chi^2 = \frac{1}{2}\sum_n \alpha_n \left[(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2\right] - \frac{1}{2}\lambda\left[x^2 + y^2 + z^2 - 1\right]$$

where x,y,z are the Cartesian coordinates corresponding to the ra and dec specified by the user, **a** is a weighting parameter calculated from the astrometric precision of the survey, and **l** is the Langrange multiplier in the minimization to ensure that the (**x,y,z**) is a unit vector. The code for spGetMatch is included in the code listings.

We compute four cumulative quantities at each cross-identification step – these are

$$a = \sum \frac{1}{\sigma_i^2}, \quad a_x = \sum \frac{x_i}{\sigma_i^2}, a_y = \sum \frac{y_i}{\sigma_i^2}, a_z = \sum \frac{z_i}{\sigma_i^2}.$$

The best position is given by the direction of (a_x, a_y, a_z). The log-likelihood at that point is given by

$$\chi^2 = a - \sqrt{a_x^2 + a_y^2 + a_z^2}$$

This is divided by the number of surveys considered up to that point, and compared to the tolerance. If a tuple's log-likelihood exceeds this threshold, it is killed. This cross-identification process is fully symmetric, the particular order of matching does not matter. The cross-matching is applied to each node recursively by the portal when it runs the query execution plan.

QI-13 SkyNodes shall implement an ExecutePlan() web method, which takes an ExecPlan and passes the relevant part of the plan to the next node. From that node it shall receive a VOTABLE of results. If there are no more nodes in the plan it simply executes the ADQL query and returns the resulting IVOAResult. The return type shall be specified in the plan at each step. The logical node and physical replicas shall be sent with the plan. The ExecPlan structure containing the portalURL the plan came from, the format the output is required in, and an ordered array of PlanElements. The PlanElement strcture shall contain the statement (ADQL document) the Target for this element (a logical node name) and an ordered array of hosts (physical nodes which should behave as the logical name – there may be only one). The order of the hosts should be in most preferable first, this is of course form the portal perspective – a node may decide to rerank the nodes for itself or simply send a quick query to all mirrors and take the first which responds.

It is not clear how clever the nodes need to be for parsing. Ideally the portal will prepare ADQL, which is good for the node to execute.

QI-14 SkyNodes should implement the footprint service. This would take a region specified in the region XML and return a new region which is the intersection of the survey an the given region. The Region specification has the ability to deal with Spectral and Temporal footprints also, this implies a node should be able to deal with these entities.

Currently we know how to deal with regions (just about) but have not really dealt with Temporal nor Spectral overlaps.

The Latest WSDL file can be located at
http://skyservice.pha.jhu.edu/devel/skynode/SkyNode.asmx?wsdl

6 Open SkyQuery Portal

There may indeed be different types of VO portal, which use the Query interface to different degrees. Arguably we should not include the portal in this specification. However it is felt that having a brief portal description in this document will help focus the direction of SkyNode development. Hence in developing a portal one would not necessarily need to follow these requirements. The OpenSkyNode portal though would be a good aim for IVOA as a demonstration project.
The ultimate aim of the Open SkyQuery portal is to allow a query such as

```
select s.ra, s.dec, s.r, s.i, s.g, s.objid, p.*
  from SDSS:PhotoObj s, Twomass:Photo t, First:Photo f,
       SDSS:Spectro p, IVOA:MyDB.MyObjList m
  where XMatch (s,t,f,m) < 3
         and  Region('Circle Cartesian 0.7 0.2 -0.1 0.02')
         and s.objid=p.objid and s.r<15 and abs(s.i-t.j_m)<2
```

This implies several things about the portal.

QI-15 The Open SkyQuery Portal shall accept queries in ADQL format.

QI-16 The portal should accept queries in SkyQL.

QI-17 The portal shall use a registry to resolve survey names. The current NVO registry prototype could already be used for this.

QI-18 The Open Sky Query portal shall perform an implicit intersection of surveys listed in the cross match command.
The current sky query requires an AREA clause for the cross match to work in. Assuming some survey provide proper footprint services we should be able to perform intersections to find the actual shape of over lap between surveys. We should also look at generating a dynamic foot print for a small list of objects i.e. the MyObjList in the example above is probably the smallest of all the surveys if we could have a footprint for that it would give us an area to search.

Execution in parallel or sequence of the individual queries has a major impact on the logic in the portal and requires extension of the nodes to accept more than one table from different surveys at the same time. The current SkyQuery works sequentially and it seems sensible to stick with this initially for Open Sky Query. Of course, there may be more than one Portal so both approaches may be attempted in parallel. Assuming a sequential plan the flow in the portal is roughly the following:

1. Parse Survey Names and convert to actual server names using the registry.

2. Select out the independent predicates e.g. the parts which are for one survey only like s.r<15 and s.objid=p.objid in the example above.
3. Pull out the Circle/Shape clause and/or perform an implicit footprint intersection using the surveys footprint services, if available. If a survey does not have the footprint intersection service then it does not decrease the area by changing the shape.
4. Create the performance queries and call the PerfomanceQuery() method on every SkyNode involved.
5. Order the surveys according to performance result to minimize network traffic.
6. Create the execution plan (ExPlan) containing the ADQL for each survey ordered by density – most dense first runs last i.e. first SkyNode in ExPlan.
7. Send plan to first survey.

6.1 Duplicate/Replicated nodes

6.1.1 What we need

We need some form of logical and physical name separation where a single logical name may refer to multiple physical names as is done in grid systems.
In SkyQL and ADQL we would refer only to logical names. The Node or Portal would then choose the most appropriate physical entity to use by requesting all physical names for the logical name and polling each, probably with a performance metric.

6.1.2 What we have

The current Registry could be used to do this by using the Short name as the Logical name for use in Queries. The Type of Registry Entry would be "IVOASkyNode" if multiple Short Names are identical it means we have mirrors of that survey. We assume all mirrors are equal for now in terms of the data they contain – if they are not they should employ different short names.
There has already been a discussion on having Short Names Unique or not and we sided with not. We would be enforcing our own rule here on a subset of the registry data – this is acceptable.

6.1.3 What we could/should have

In the new RSM and resource xsd there is a link attribute which allows relationships to be expressed between resources. One type of link is "mirror". Hence in the new scheme we could ensure that the mirror attributes are used to say which SkyNodes are mirrors of which other ones. This could be seen as a driver for having this mechanism in the registry.

7 Changes from previous versions

- ADQL in separate document
- Use port terminology for web service
- Added heartbeat
- Added Functions port
- Changed PerformanceQuery to QueryCost
- More info in execplan
- Added ucds to metadata – extended metadata description
- Added type and precission to Colums port.

- Adde Xmatch description

8 References

[SQL92] http://www.contrib.andrew.cmu.edu/%7Eshadow/sql/sql1992.txt

[HTMDLL] Alex Szalay, George Fekete, Jim Gray; SQL SERVER 2000 HTM Interface Release 2.10; July 12, 2003 http://skyservice.pha.jhu.edu/develop/vo/adql/htmdll_2_0.doc

[ADQL] IVOA VOQL Working group; IVOA Astronomical Data Query Language – get latest from www.ivoa.net/voql