



IVOA SkyNode Interface Version 0.7.4

IVOA Working Draft
2004-05-17

This version:

0.7.4 <http://www.ivoa.net/internal/IVOA/IvoaVOQL/SkyNodeInterface-0.7.4.pdf>

Previous versions:

0.7 <http://www.ivoa.net/internal/IVOA/IvoaVOQL/SkyNodeInterface-0.7.pdf>

0.6 <http://skyservice.pha.jhu.edu/develop/vo/adql/SkyNodeInterface-0.6.pdf>

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.

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](http://www.ivoa.net/docs/) can be found at <http://www.ivoa.net/docs/>.

Acknowledgments

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

Contents

Abstract.....	1
Status of this document.....	1
Acknowledgments.....	2
Contents.....	2
1 Introduction	2
1.1 OpenSkyQuery.....	3
2 Open SkyQuery Portal, IVOASkyNode, ADQL and VOQL – brief architectural discussion.....	3
3 Error Handling.....	4
4 Standard Interface.....	4
5 IVOA SkyNode Interface	4
5.1 Feature Matrix	4
5.2 IVOA SkyNode Interface	5
5.2.1 Basic Sky Node.....	5
5.2.2 Full SkyNode requirements	6
6 Changes from previous versions.....	8
7 References.....	8

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. Any IVO site implementing the Query Interface would be considered a “Basic SkyNode”. More advanced nodes would participate in distributed queries through the portal at OpenSkyQuery.NET

1.1 OpenSkyQuery

SkyQuery (www.skyquery.net) is a successful implantation of distributed astronomical query system. This is implemented using some proprietary Microsoft Technology but for the most part uses SOAP Services. Several people have expressed 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.

OpenSkyQuery (<http://www.openskyquery.net/>) opens up the SkyQuery protocol to enable other databases and servers to become “Full SkyNodes”.

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.

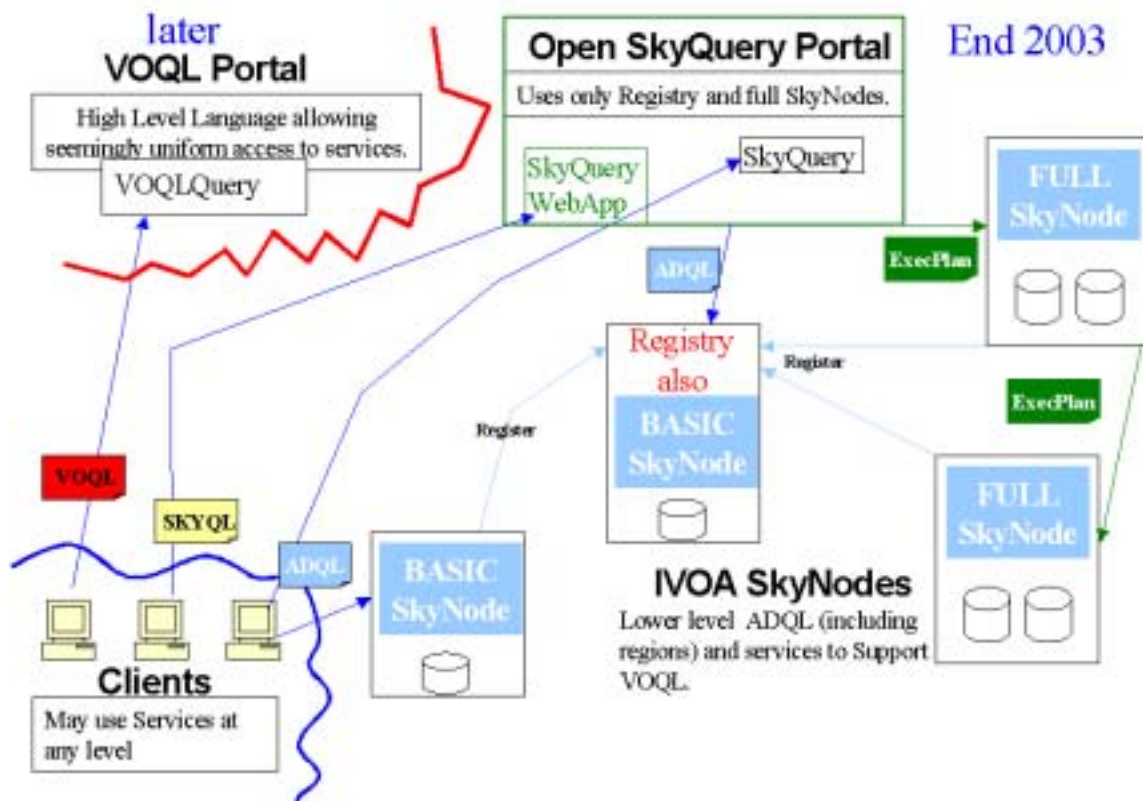


Figure 1. Architecture

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 Interface

- QI-1** SkyNodes shall implement the services outlined in the standard interfaces specification [STDIF]

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 and units 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.

Below we split the requirements in two sections – the minimum set required to be a skynode called Basic and more advanced requirements aiming toward complex portal creation. It is assumed nodes will exist in the spectrum between these extremes. The Functions interface allow the node to specify precisely what it supports.

5.2.1 Basic Sky Node

QI-2 SkyNodes shall register with the registry with type=”SkyNode”

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, Intermediate, or Full
- AcceptsUCDs : True or False

Specifying AcceptsUCDs means that a query 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 sense 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 the unit and type of the native SkyNode column to specify the query correctly.

QI-3 SkyNodes shall implement the “Tables” interface, which returns a list of all tables that may be used in ADQL passed to this endpoint.

QI-4 SkyNodes shall implement the “Columns” interface, which returns information about the columns of a given table name. This shall return an array of MetaColumns this structure shall contain the Column Name, Type (as in SQL type), Unit, Precision (number of significant digits), Description, and UCD.

- QI-5** SkyNodes shall implement the “Formats” interface. 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-6** SkyNodes shall implement the “Functions” interface, 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-7** SkyNodes shall implement the "PerformQuery" interface, which takes an XML document including an ADQL query and an optional string parameter called “format”. Fomrat must be one of the strings listed from a call to the "Formats" interface. This returns a document including a single appropriate IVOAResult, which is the result of processing the query. IVOAResult shall be an abstract class with multiple subclasses, one for each format. SkyNodes shall support at least VOTable format.
- QI-8** 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 columns
```

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-9** SkyNodes shall implement QueryCost() interface 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.

The notion here is to find out the volume of data which would be returned from the specified query. This allows nodes to be ranked in the correct order for a Xmatch.

- QI-10** SkyNodes shall accept complex Shapes in their queries as defined in the ADQL syntax (using the region spec).
- QI-11** 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:

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

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 λ is the Lagrange 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

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

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.

A particular node may decide to implement this differently however if the chisquare parameters are not returned than other nodes wishing to use the above algorithm will not function correctly.

In dealing with tables the obvious approach is to load the table into a temporary table in the database. Here again other approaches may be adopted by the implementer. The interface is all that matters.

QI-12 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 `VOData` result. If there are no more nodes in the plan it simply executes the ADQL query and returns the resulting `VOResult`. 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` structure 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 from 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.

- QI-13** 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 and 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 is based on ADQL0.7.4 and can be located at <http://openskyquery.net/nodes/sdss/nodeb.asmx?wsdl>

6 Changes from previous versions

- Put in links to openskyquery.net
- Link to new wsdl

7 References

- [SQL92] <http://www.contrib.andrew.cmu.edu/~eshadow/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
<http://www.ivoa.net/internal/IVOA/IvoaVOQL/ADQL-0.7.4.pdf>
- [STDIF] IVOA Grid & Web Services Working Group; Standard Interfaces;
<http://www.ivoa.net/internal/IVOA/IvoaGridAndWebServices/StandardInterfaces-0.1.pdf>