



## IVOA SkyNode Interface Version 0.4

### IVOA Working Draft 2003-08-26

**This version:**

**0.3** <http://skyservice.pha.jhu.edu/develop/vo/adql/QueryInterface-2003Aug.pdf>

**Previous versions:**

**0.2** <http://skyservice.pha.jhu.edu/develop/vo/adql/QueryInterface-2003July.pdf>

**Editors:**

Masatoshi Ohishi, Alex Szalay

**Authors:**

Jim Gray, Tamas Budavari, Vivek Haridas, Tanu Malik, Bob Mann, William O'Mullane, Roy Williams, Naoki Yasuda

**Please send comments to:** [openskyquery@eta.pha.jhu.edu](mailto:openskyquery@eta.pha.jhu.edu)

## Abstract

This document describes the minimum required interface to participate in the IVOA as a queryable SkyNode as well as requirements to be 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 the Cambridge IVOA meeting.

## Contents

Abstract .....	1
Status of this document.....	1
Acknowledgments .....	2
Contents.....	2
1 Introduction .....	2
2 Open SkyQuery Portal, IVOASkyNode, ADQL and VOQL – brief architectural discussion. ....	3
3 Error Handling .....	4
4 Standard VO Interface .....	4
5 Astronomical Data Query Language(ADQL) .....	4
6 Open SkyQuery Portal .....	5
6.1 Duplicate/Replicated nodes.....	6
7 IVOA SkyNode Interface .....	7
7.1 Feature Matrix .....	7
7.2 IVOA SkyNode Interface .....	7
7.2.1 Level 1 Sky Node.....	7
7.2.2 Higher order SkyNode requirements .....	8
8 Changes from previous versions.....	9

## 1 Introduction

At the Cambridge IVOA meeting there was general agreement to the ADQL 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.

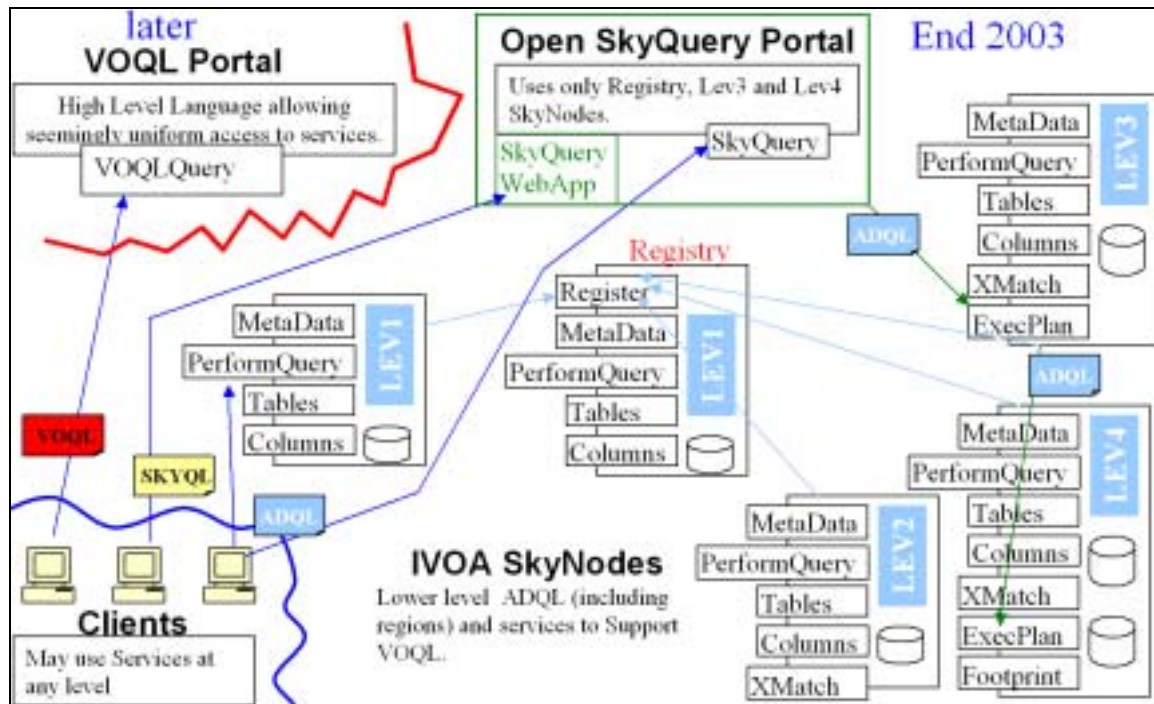
We have now demonstrated that a single WSDL (Web Services Definition Language) file may be implemented in at least Java and C#, i.e. on Linux as well as windows. Also we have demonstrated that a single client may interact with both implementations of the service.

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.

Next would like to Open up the SkyQuery protocol to enable other databases and servers to become SkyNodes. 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.

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](http://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 SkyQL : 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/adqlarchive/archiveservice.asmx>.

- QI-1** All VO WebServices shall implement the “MetaData” service. This service 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.

### 5 Astronomical Data Query Language(ADQL)

ADQL is passed as an XML document to the Query Interface. ADQL is based on a subset of SQL plus Circle(Cone Search). The only SQL command allowed in ADQL is a “select”. The full XSD for ADQL and services for translation of SQL to ADQL and back may be found at

<http://skyservice.pha.jhu.edu/develop/vo/adql/>

- QI-2** ADQL shall be in the form of an XML document.
- QI-3** All table names in ADQL shall be followed by an alias.
- QI-4** ADQL shall support Column aliasing using the AS keyword.
- QI-5** ADQL shall support the region specification as defined by the region.xsd of the IVOA/NVO.
- QI-6** Mathematical functions shall be allowed in ADQL as follows:
- Trigonometric functions: acos, asin, atan, cos, cot, sin, tan  
Math functions: abs, ceiling, degree, exp, floor, ln, log, mod, pi, power, radians, sqrt, floor

Sample applications and tutorials for development and deployment of ADQL services is available at <http://skyservice.pha.jhu.edu/develop/vo/adql/>

## 6 SkyQL

SkyQL a string like representation of ADQL. Semantically SkyQL and ADQL are identical. Syntactically ADQL is XML and SkyQL is more human readable. This is like the current SkyQL in SkyQuery.NET .

**QI-7** SkyQL shall have SQL like Syntax.

One interesting problem to be solved here is the use of STC for region specification. It seems some simple representation would work for some cases but ultimately we will need to refer to an XML document for some region specifications in SkyQL.

We foresee 3 ways of using regions in SkyQL.

**QI-8** SkyQL shall support the Region keyword. This will be followed by a single quoted string specifying a region in a simple manner similar to the current SDSS cover specification in [HTMDLL]. This would look something like:

```
Region ('CIRCLE J2000 19.5 -36.7 0.02')
```

More complex regions may require the use of XML directly.

**QI-9** It should be possible to refer to a region specification as a url in SkyQL using the RegionURL keyword e.g.

```
RegionURL ('http://aserver.edu/aregion.xml')
```

**QI-10** It should be possible to inline a region specification as in SkyQL using the RegionXML keyword e.g. (not a valid region spec )

```
RegionXML ('<circle><coordsys>j2000</coordsys><ra>19.5</ra><dec>-36.7</dec><radius>0.02</radius></circle>')
```

## 7 Open SkyQuery Portal

There may indeed be different types of VO portal, which use the Query interface to different degrees. Even for a full SkyQuery portal it should be able to work with both LEV3 and LEV4 nodes.

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('J2000 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-11** The Open SkyQuery Portal shall accept queries in ADQL format.

**QI-12** The portal should accept queries in SkyQL.

**QI-13** The portal shall use a registry to resolve survey names. The current NVO registry prototype could already be used for this.

**QI-14** 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 overlap between surveys. We should also look at generating a dynamic footprint 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 `PerformanceQuery()` 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.

## 7.1 Duplicate/Replicated nodes

### 7.1.1 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.

### 7.1.2 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.

## 8 IVOA SkyNode Interface

### 8.1 Feature Matrix

The following matrix tries to categorize features and SkyNode/ADQL Levels.

LEV1 here is the minimum IVOA SkyNode Interface – this is useful in itself as it allows one to send queries to a system using ADQL. 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 implement LEV4 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	LEV1– ADQL	LEV2 – XMATCH	LEV3- SKYNODE	LEV4- SKYNODE Advanced	Optional
ADQL – circle	X	X	X	X	
Xmatch		X	X	X	
Performance Query			X	X	
Takes exec plan			X	X	
Footprint – Region intersect				X	X
MYDB					X
Authentication – must have with MYDB					X

### 8.2 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.

#### 8.2.1 Level 1 Sky Node

**QI-15** 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 e.g. LEVEL 3 and 4

**QI-16** SkyNodes shall implement the “Tables” service, which returns a list of all tables that may be used in ADQL passed to this endpoint.



- QI-17** SkyNodes shall implement the “Columns” service, 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, Unit, Description, and UCD.
- QI-18** SkyNodes shall implement the “PerformQuery” service, which takes an ADQL document as a parameter and returns a VOTable, which is the result of processing the query.
- QI-19** SkyNodes should accept Standard SQL-92 Metadata [SQL92] queries. These queries include

```
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).

### 8.2.2 Higher order SkyNode requirements

- QI-20** SkyNodes shall implement PerformanceQuery() method 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-21** SkyNodes shall accept complex Shapes in their queries as defined in the ADQL syntax (using the region spec).
- QI-22** 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.

- QI-23** SkyNodes shall implement an ExecutePlan() web method, which takes an ExPlan 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 VOTable

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-24** 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 which 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/adqlArchive/ArchiveService.asmx?wsdl>

## 9 Changes from previous versions

- Added 3 layers of VOQL to VOQL Section
- Tidied Duplicate/Replica section
- Added link to SQL-92 spec for metadata queries plus completed the text.
- Added footprint requirement
- Added new SKYQL section with RegionURL RegionXML Region removed Circle

## 10 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](http://skyservice.pha.jhu.edu/develop/vo/adql/htmdll_2_0.doc)