IVOA Registry Interfaces
Version 0.1

IVOA Working Draft
2004-01-27

This version:
0.1 http://www.ivoa.net/internal/IVOA/NotYetPlaced

Previous versions:

Editors:
Tony Linde

Authors:
Kevin Benson
Wil O'Mullane
Gretchen Greene
Martin Hill
Elizabeth Auden
Ray Plante
Alex Szalay

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

Abstract

This document describes the minimum required interface to participate in the IVOA Registry 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.

Contents

Abstract 1
Status of this document 1
Acknowledgments 2
Contents 2
1 Introduction 2
1.1 About the document 3
1.2 IVOA Registry Interface 3
2 References 3
3 Standard Query 3
4 Helper Queries 3
4.1 Keyword Search Query 3
4.2 Finding Other Registries 4
4.3 Harvesting 4
4.3.1 Harvesting By Date 4
4.3.2 Harvest All (Replicate) 4
4.3.3 Harvest Resource 4
4.3.4 Recommendations on Harvesting 4
5 Paging ..... **Error! Bookmark not defined.**
6 Best Practices 5
7 Changes from previous versions 5

1 Introduction

For interoperability purposes it will be beneficial for Registries of different organizations and institutions to share a standard way of interaction and communication. This document defines web service interfaces that constitute a part of a standard Registry implementation and attempts to create this standard way of interaction between the communities. This document does not describe the standards of the registry related to the communication protocols, for example: web browser (http)

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 RI-N where N is a running number in the document.

1.2 IVOA Registry Interface

2 References

- Standard Interface Document
- Reference to IVOA Web Site
- Resource Metadata Specifications
- OAI documents if needed

3 Standard Query

RI-1 IVOA searchable registries shall support the 'Search' Interface, which takes one parameter corresponding to the “QueryResource” Schema See Appendix A.

4 Helper Queries

4.1 Keyword Search Query

This is the simplest query interface of the three types and the main objective of this query is to make a quick keyword query that automatically searches against the supported names.

RI-2 Query Registry shall have a “KeyWordSearch” interface, which has two parameters. The first parameter is a string parameter that will be word(s) separated by spaces; or more specific joined alphanumeric character(s) separated by spaces. The second parameter is a Boolean to determine an “OR” operation in dealing with more than one word. If it is true then “any” of the words matched shall cause a return of that resource entry. If the second parameter is not true then the default action of “AND” occurs resulting in all words must match. The service will return resources with any of the words in it.

RI-3 This interface shall at a minimum search the required elements/names in the given Resource schema dealing with common text. It is expected that most registries will support other elements to their Keyword searching.

Required Elements:

- Identifier (AuthorityID and ResourceKey)
- Description (Summary/Description)
- Title
- ResourceType (usually the xsi:type)
- Subject - Optional in schema, but expected.
- Type – Optional in schema, but expected.

4.2 Finding Other Registries

RI-4 Searchable Registries shall implement the 'GetRegistries' interface, which returns all registry resources contained in this registry. This interface shall return entries in the form of resources. The above requirement is a helpful query when acquiring knowledge of other registries for harvesting data.

4.3 Harvesting

By means of Harvesting a registry (let us call it A) queries another registry (let us call it B) for the purpose of an update to occur on the source registry (A). In general terms registry updating/inserting information from another external registry. Again this document does not describe the updating mechanism. This section describes the harvest methods for the Query interface dealing with harvesting. Results from harvest query interfaces should be returned following the Resource Metadata Schema for the version of this registry.

RI-5 A harvesting query should only return resources for which it manages the authorityid.

4.3.1 Harvesting By Date

Deals with querying the registry based on a Date or to be more precise a Timestamp. How the registry keeps track of all Resource records is not discussed, but it is assumed some type of modification date is kept for all Resource records.

RI-6 Registry shall implement a “HarvestFrom” interface taking in a timestamp parameter. “from” gets results based on the “>=” (Greater-Than-Equal) Operation rule for dates.

4.3.2 Harvest All (Replicate)

RI-7 Registry shall implement a “Harvest” interface with no parameters that returns everything in the registry. Primary use is for replication.

4.3.3 Harvest Resource

A general question is “How does a Registry tell about itself to all the other registries?” This interface is for that particular purpose to register a registry and even register any resource entry.

RI-8 Registry shall implement a “HarvestResource” interface that will take in a Resource confined by the Resource Schema that is of type “Service” or an extension of “Service”. This interface will then store or register this entry in its Registry. The implementation should begin a harvest on the resource entry.

RI-9 If the Resource entry is not of a type “Registry”, then use the “Metadata” interface from the “Standard Interface Document” to harvest the metadata on this particular service.

4.3.4 Recommendations on Harvesting

Every Registry resource entry has elements known as ManagedAuthority these are the Authority IDs that this registry manages, and no other registry should be managing these authority IDs. A harvest query on a registry should only return Resource metadata that has changed for the authority IDs that it manages.

This document does not describe how often you harvest data from an external registry, but as stated in the best practices section it is preferred to always use the HarvestFrom interface. This means internally you are keeping track of the last date you did a harvest on an external Registry.

Also as stated in the "Best Practices" section to use the "MetaDataChangedOn" interface to see if a harvest even needs to be performed.

How do you know about other registries? There are several solutions to this, which this document does not attempt to make a standard way. The following ways are suggested below and can be used in combination if desired:

- Do a search on the "ResourceType" supported name for all the "RegistryType" types. Assuming that at least one external registry exists to do this query on. This returns other Registry resource entries to analyze and begin harvesting.
- See Standard Interface Document about the "Metadata" interface for connecting to an external Registry Resource to analyze.
- Remember other registries may have already used the "HarvestResource" interface for registering the Registry resource entry; hence it may already be an entry in your registry being harvested.

5 Best Practices

This is a general best practice session that is best to be followed when performing queries on the registry.

- Use the Helper Queries if possible instead of creating a Standard Query.
- It is desirable to keep track of where a resource was harvested from. In this way we may track failures or errors through multiple registries by following the chain. A harvestedDate field is also useful.
- Never delete anything in the registry - always make new copies and flag old ones as superseded. Likewise if a record is deleted merely flag it deleted.
- All queries then need to be modified to filter out the deleted/superseded records. Again this will be very useful if erroneous records crop up or in the case of accidental deletion.
- Try to use HarvestFrom interface instead of replicating each time with a full Harvest.
- Do not duplicate authority ids that other registries own. New registries are better to harvest first so that a lookup of other authority ids can be identified.

6 Changes from previous versions

- First draft

Appendix A – QueryResource Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.ivoa.net/xml/QueryResource/v0.1"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                    xmlns="http://www.w3.org/2001/XMLSchema"
                    xmlns:qr="http://www.ivoa.net/xml/QueryResource/v0.1"
                    elementFormDefault="qualified"
                    attributeFormDefault="unqualified"
                   version="0.1">

        <xsd:annotation>
        <xsd:documentation>
           This is the first draft version of the Query schema for querying a Resource
Registry.  The results of the query are determined by the implementation, but is
           expected to represent a schema conforming to the Resource schema.
           See: http://www.ivoa.net/xml/VOResource/v0.9
           The structure of this schema is much like a very basic criteria or predicate to
most popular query languages.
        </xsd:documentation>
     </xsd:annotation>

   <xsd:element name="RegistryQuery">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="qr:QueryResources"/>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="QueryCriteria">
        <xsd:annotation>
        <xsd:documentation>
            Deals with a particular criteria to perform on a Resource Registry.  This section
looks very much like most query languanges except it uses an XPath
            to represent on what you querying against.
        </xsd:documentation>
     </xsd:annotation>

       <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="ResourceXPath" type="xsd:string"/>
            <xsd:element name="Operand">
               <xsd:simpleType>
                  <xsd:restriction base="xsd:string">
                     <xsd:enumeration value="like"/>
```

```
                    <xsd:enumeration value="startswith"/>
                    <xsd:enumeration value="endswith"/>
                    <xsd:enumeration value="greatar-than-or-equals"/>
                    <xsd:enumeration value="less-than-or-equals"/>
                    <xsd:enumeration value="greater-than"/>
                    <xsd:enumeration value="less-than"/>
                    <xsd:enumeration value="equals"/>
                    <xsd:enumeration value="not"/>
                  </xsd:restriction>
                </xsd:simpleType>
              </xsd:element>
              <xsd:element name="value" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>

        <xsd:element name="QueryResources">
              <xsd:annotation>
           <xsd:documentation>

           </xsd:documentation>
         </xsd:annotation>

           <xsd:complexType>
             <xsd:sequence>
               <xsd:element ref="qr:QueryCriteria"/>
                  <xsd:sequence  minOccurs="0" maxOccurs="unbounded">
                  <xsd:element name="JoinOperator">
                     <xsd:simpleType>
                        <xsd:restriction base="xsd:string">
                           <xsd:enumeration value="and"/>
                           <xsd:enumeration value="or"/>
                        </xsd:restriction>
                     </xsd:simpleType>
                </xsd:element>
              <xsd:element ref="qr:QueryCriteria"/>
             </xsd:sequence>
           </xsd:sequence>
           </xsd:complexType>
        </xsd:element>

</xsd:schema>
```