International

Virtual

Observatory

Alliance

IVOA Registry Interfaces
Version 0.8.4

IVOA Working Draft 2004 June 16

This version:
0.8.1 http://www.ivoa.net/internal/IVOA/RegistryInterface/IVOARegistryIntervace-0.8.1.pdf
Latest version:
LatestVersion
Previous versions:
PreviousVersion(s)-YYYYMMDD
Authors:
Kevin Benson, Elizabeth Auden, Matthew Graham, Gretchen Greene, Martin Hill, Tony Linde, Dave Morris, Wil O'Mullane, Ray Plante, Guy Rixon, Kona Andrews

---

Abstract

Registries provide a mechanism with which VO applications can discover and select resources—e.g. data and services—that are relevant for a particular scientific problem. This specification defines the interfaces that support interactions between applications and registries as well as between the registries themselves. It is based on a general, distributed model composed of so-called *searchable* and *publishing* registries. The specification has two main components: an interface for searching and an interface for *harvesting*. All interfaces are defined by a standard Web Service Description Language (WSDL) document; however, harvesting is also supported through the existing Open Archives Initiative Protocol for Metadata Harvesting, defined as an HTTP GET interface. Finally, this specification details the metadata used to describe registries themselves as resources using an extension of the VOResource metadata schema.

Status of This Document

This is a Working Draft. The first release of this document was 2006 June 20.

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/Documents/.

Acknowledgements

This document has been developed with support from the National Science Foundation's Information Technology Research Program under Cooperative Agreement AST0122449 with The Johns Hopkins University, from the UK Particle Physics and Astronomy Research Council (PPARC), and from the European Commission's Sixth Framework Program via the Optical Infrared Coordination Network (OPTICON).

Conformance-related definitions

The words "MUST", "SHALL", "SHOULD", "MAY", "RECOMMENDED", and "OPTIONAL" (in upper or lower case) used in this document are to be interpreted as described in IETF standard, RFC 2119 [RFC 2119].

The **Virtual Observatory (VO)** is a general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The **International Virtual Observatory Alliance (IVOA)** is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

A **Web Service** (when capitalized as it is here) refers to a service that is in actuality described by a Web Service Description Language (WSDL) [WSDLv1.1] document.

Editor's Note:
This document contains two types of boxed comments like this one. Those marked "Editor's Note" represents comments intended for the standard editors and for reviewers; these comments would be removed when the issues they discuss are addressed. Those marked simply as "Note" are intended for those who will implement the standard, and are intended to provide tips and further explanation of how the standard is expected to be used. These notes are expected to remain embedded in the final version of the document

Contents

1 Introduction

In the Virtual Observatory (VO), registries provide a means for discovering useful data and services. To make discovery efficient, a registry typically represents to some extent a centralized warehouse of resource descriptions; however, the source of this information—the resources themselves and the data providers that maintain them—are distributed. Furthermore, there need not be a single registry that serves the entire international VO community. Given the inherent distributed nature of the information used for resource discovery, there is clearly a need for common mechanisms for registry communication and interaction.

This document describes the standard interfaces that enable interoperable registries. These interfaces are based in large part on a Web Service definition in the form of a WSDL document [WSDLv1.1], which is included in this specification. Through these interfaces, registry builders have a common way of sharing resource descriptions with users, applications, and other registries. Client applications can be built according to this specification and be able to discover and retrieve descriptions from any compliant registry.

This specification does not preclude a registry builder from providing additional value-added interfaces and capabilities. In particular, they are free to build interactive, end-user interfaces in any way that best serves their target community. In a similar spirit, this specification does not intend to enforce completely identical behavior of required operations across all compliant implementations. In particular, this specification does not require that identical search queries sent to different compliant registries return identical results. Implementations are free to support different strategies for evaluating an ambiguous query (such as a keyword search) and ordering the results in a way that best serves the target community.

1.1 Registry Architecture and Definitions

A **registry** is first a repository of structured descriptions of *resources*, building on concept of a VO **resource** defined by the IVOA Recommendation, “Resource Metadata for the Virtual Observatory” (RM) [Hanisch 2004]:

> A *resource* is a general term referring to a VO element that can be described in terms of who curates or maintains it and which can be given a name and a unique identifier. Just about anything can be a resource: it can be an abstract idea, such as sky coverage or an instrumental setup, or it can be fairly concrete, like an organization or a data collection.

Organizations, data collections, and services can be considered as classes of resources. The most important type of resource to applications is a service that actually does something. What is available at a particular resource is described

through the content of metadata, whereas the service metadata describes how to access it. The RM describes a registry, then, as "a service for which the response is a structured description of resources" [Hanisch 2004]. Each resource description it returns is referred to as a **resource record**.

This specification is based on the general IVOA model for registries [Plante 2003], which builds on the RM model for resources. In the registry model, the VO environment features different types of registries that serve different functions. The primary distinction is between *publishing* registries and *searchable* ones. A secondary distinction is *full* versus *local*.

A **searchable registry** is one that allows users and client applications to search for resource records using selection criteria against the metadata contained in the records. The purpose of this type of registry is to aggregate descriptions of many resources distributed across the network. By providing a single place to locate data and services, applications are saved from having to visit many different sites to just to determine which ones are relevant to the scientific problem at hand. A searchable registry gathers its descriptions from across the network through a process called *harvesting*.

A **publishing registry** is one that simply exposes its resource descriptions to the VO environment in a way that allows those descriptions to be harvested. The contents of these registries tend to be limited to resources maintained by one or a few providers and thus are local in nature; for example, a data center will run its own publishing registry to expose all the resources it maintains to the VO environment. Since the purpose is simply publishing and not to serve users and applications directly, it is not necessary to support full searching capabilities. This simplifies the requirements for a publishing registry: not only does it not need to support the general search interface, the storage and management of the records can be simpler. While a searchable registry in practice will necessitate the use of a database system, a publishing registry can easily store its records as flat files on disk.

Note that some registries can play both roles; that is, a searchable registry may also publish its own resource descriptions.

A secondary distinction is *full* versus *local*. A **full registry** is one that attempts to contain records of all resources known to the VO. In practice, this attribute is associated only with searchable registries, as in the so-called **full searchable registry**. It is expected that there will be several such registries available, perhaps each run by a major VO project; this not only avoids the single point of failure, but allows some specialization to serve the particular needs of the project that maintains it. A **local registry**, on the other hand, contains only a subset of known resources. In practice, all publishing registries are local; however, we expect that there may be **local searchable registries** that specialize in particular types of resources, perhaps oriented toward a scientific topic.

As mentioned above, **harvesting** is the mechanism by which a registry can collect resource records from other registries. This mechanism is used by full searchable registries to aggregate resource records from many publishing registries. It can also be used to synchronize two registries to ensure that they have the same contents. Harvesting, in this specification, is modeled as a "pull" operation between two registries. The **harvester** refers to the registry that wishes to receive records (usually a searchable registry); it sends its request to the **harvestee** (usually the publishing registry), which responds with the records. Harvesting is intended to be a much simpler process than search and retrieval; nevertheless, there are at least two kinds of filtering that a harvestee needs to support:

- **Filtering by date:** this allows the harvester to return to the harvestee periodically to retrieve only new and updated records.
- **Filtering by ownership:** by harvesting only those records that originated with the harvestee (as opposed to those that may have been harvested from other registries) prevents a harvester from receiving duplicate records from multiple registries.

Other kinds of filtering can be useful as well (such as filtering on resource type). Note, however, that filtering is not intended to be an equivalent to arbitrary searching; rather, it is a gross selection that can be easily implemented without having to process the contents of each record.

1.2 Specification Summary

The purpose of the *registry* is to be used by other applications to provide access to various types of resources. At the programmatic level, connectivity of the registry and other applications is ensured through the registry interface as defined by this document. Much of the interface is defined as a SOAP-based Web Service and is described the WSDL documents included in Appendix A.1, while the harvesting interface (section 3) is specifically defined by the Open Archives Initiative standard called the Protocol for Metadata Harvesting (OAI-PMH) [OAI]. To define the registry interface, this document includes the following definitions:

- The meaning and behavior of four required searching operations, one optional search operation, and six required harvesting operations.
- The required input arguments for each operation.
- The XML Schema used to encode response messages.
- The meaning of the output for each operation.

The registry interface is composed of two independent parts. The harvesting interface provides the mechanism for registries to talk to each other and share information. The searching interface is used by clients that want to discover

resources to use as part of a VO application. A registry may implement either interface or both, depending on the roles it intends to play.

The searching interface can return XML descriptions of resources, or resource records, and it consists of four required operations, described in more detail in section 2:

- **Search** returns resource records that match a specific set of constraints.
- **KeywordSearch** returns resource records containing specified keywords.
- **GetResource** returns a single resource records identified by its unique IVOA identifier.
- **GetIdentity** returns the resource record describing the searchable registry itself.

The searching interface also includes an optional, alternative search option based on XQuery.

The harvesting interace, which allows one registry to collect resource records from other registries, leverages the OAI-PMH standard [OAI]. Described in more detail in section 3, the OAI-PMH interface is composed of six operations. The most commonly used harvesting operation is the **ListRecords**, which can return the descriptions of the resources of a particular category, or *set*,.that have been created or updated since a specified date. Normally the harvester will request records from the set that originate from that registry (as opposed to those harvested from another registry). This set is said to be *managed* by the registry. The complete list of OAI-PMH harvesting operations are:

- **Identify** returns the resource record describing the harvestable registry itself.
- **ListIdentifiers** lists the identifiers of records that have changed since a given date
- **ListRecords** lists the full descriptions of resources that have changed since a given date.
- **GetRecord** returns a single record identified by its identifier.
- **ListMetadataFormats** lists the available output description formats.
- **ListSets** lists the categories of records that can be requested.

The searching and harvesting operations that return resource descriptions do so using the VOResource XML Schema and any of its legal extensions [VOResource].

2 Searching

The four required operations that make up the searching interface fall into two groups: *search* and *resolve*. The *search* operations—**Search** and **KeywordSearch**—return a list of one or more resource descriptions held by the

registry that matches the input selection criteria. The **Search** operation supports constraint-based searching for resources by means of a query using the Astronomical Data Query Language (ADQL) [ADQL], **KeywordSearch** provides keyword-based searching for resources whose descriptions contain words in an input string.

The resolve operations---**GetResource** and **GetIdentity**---each return one and only one resource description. The **GetResource** resolves a unique IVOA Identifier [Identifier] to the description of the associated resource. The **GetIdentity** returns the resource record of the searchable registry itself.

> **Note:**
> It is important to note that search operations do not support resource harvesting described in section 3. Normally, an end-user would use search to retrieve resource descriptions, but not to selectively harvest information between registries.

These four operations are defined by the WSDL document given in Appendix A.1. Searchable registries must implement all four operations.

The searchable interface allows additional, optional search operations. Currently, this specification defines only one optional operation. **XQuerySearch** returns selected information extracted from resource descriptions that match a set of constraints expressed using the XQuery syntax [XQuery].

These operations are implemented to accept input parameters and return output results as SOAP messages in compliance with the WSDL document given in Appendix A.1. Compliant searchable registries must return a copy of the WSDL document whenever a client invokes the service endpoint URL with the **"?wsdl**" argument appended to it. The returned WSDL may contain additional operations beyond those specified in this section; however, the original searching operations, their arguments and their outputs must not be altered.

2.1 Required Search Operations

The two search operations—**Search** and **KeywordSearch**—return resource records that match a set of selection criteria.

2.1.1 Output Format

The two search operations share a common output format for the resource records that match the search criteria. The response is a SOAP message in compliance with the WSDL document given in Appendix A.1. This message is defined to have a single part: a **`SearchResponse`** element from the http://www.ivoa.net/wsdl/RegistrySearch/v1.0 namespace. This element in turn wraps a single **`VOResources`** element which contains each of the matching records and conforms to the following XML Schema definition:

VOResources Element Definition

```
<xs:element name="VOResources">
   <xs:complexType>
      <xs:sequence>
         <xs:choice>
            <xs:element name="Resource" type="vr:Resource"
                        minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="identifier" type="vr:IdentifierURI"
                        minOccurs="0" maxOccurs="unbounded"/>
         </xs:choice>
      </xs:sequence>
      <xs:attribute name="from" type="xs:positiveInteger" />
      <xs:attribute name="numberReturned" type="xs:positiveInteger" />
      <xs:attribute name="more" type="xs:boolean" />
   </xs:complexType>
</xs:element>
```

The [required] **`VOResources`** attributes allow the results of the query to be "paged" over several calls which can be controlled via the operation input parameters, **from** and **max**. They assume that over some limited amount of time multiple calls to a search operation on a single registry with the same search constraints returns the same results and in the same order.

VOResources Attributes

Attribute	Definition	
`from`	*Value Type:*	integer
	Semantic Meaning:	the 1-relative position of the first record returned among the total set of matched elements.
`numberReturned`	*Value Type:*	integer
	Semantic Meaning:	the number of records returned in this response.
`more`	*Value Type:*	boolean
	Semantic Meaning:	If true, additional results are available beginning with a position of `from` + `numberReturned`. If false, no more results are available beyond this set.

If all records matched by the query are returned in a single response the value of **`more`** must be set to false and **`from`** must be set to 1.

Example:
<VOResource from="100" numberReturned="200" more="true">
...
</VOResourced>

The contents of the **VOResources** element depends on the value of the operation input parameter, **identifiersOnly**. If this parameter is set to true, then the **VOResources** element must contain a list of **identifier** elements that contain the IVOA identifiers of resources that match the input query. If **identifiersOnly** is false, then the **VOResources** element must contain a list of **Resource** elements containing the full VOResource descriptions of resources that matches the query. Each child **Resource** element is of type **Resource** from the VOResource XML Schema (having the namespace, http://www.ivoa.net/xml/VOResource/v1.0, hitherto referred to using the "**vr:**" prefix), or a legal extension of the **vr:Resource** type. If the type of the **Resource** element is actually an extension of the **vr:Resource** type, then the **Resource** element MUST specify the specific type using an **xsi:type** attribute (where the **xsi** prefix refers to the http://www.w3.org/2001/XMLSchema-instance namespace) in compliance with the XML Schema standard [Schema].

The search responses must include the **xsi:schemaLocation** attribute (regardless of the value of **identifiersOnly**) in compliance with the XML Schema standard [Schema] to indicate a URL location for the VOResource schema and all of the legal extensions of VOResource that are employed in the response. This **xsi:schemaLocation** attribute must appear either as an attribute of the **VOResources** element or as an attribute of each child **Resource** element (when **identifiersOnly** is false) or both. When **xsi:schemaLocation** appears as an attribute of **Resource**, locations need only be given for the schemas employed within that resource. The URL location for the VOResource core schema (http://www.ivoa.net/xml/VOResource/v1.0) must be set to "http://www.ivoa.net/xml/VOResource/v1.0". For those legal extensions that are standard schemas recognized by the IVOA, the location should be set to the standard location in the IVOA Document repository whose URL begins with "http://www.ivoa.net/xml/".

Example: This illustrates the use of **xsi:type** and **xsi:schemaLocation** attributes in the search output results. the **xsi:schemaLocation** attribute contains pairs of values where the first value is the schema namespace and the second value is the URL location of that schema. For IVOA standard schemas, the namespace can be used as the URL location.

```
<VOResources
   xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.0"
   xmlns:vg="http://www.ivoa.net/xml/Registry/v1.0"
```

```
   xsi:schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0
                       http://www.ivoa.net/xml/VOResource/v1.0
                       http://www.ivoa.net/xml/VODataService/v1.0
                       http://www.ivoa.net/xml/VODataService/v1.0
                       http://www.ivoa.net/xml/SIA/v1.0
                       http://www.ivoa.net/xml/SIA/v1.0
                       http://www.ivoa.net/xml/Registry/v1.0
                       http://www.ivoa.net/xml/Registry/v1.0">
   <Resource xsi:type="vs:CatalogService" status="active" …>
     …
   </Resource>
   <Resource xsi:type="vs:Registry" status="active" …>
     …
   </Resource>
</VOResources>
```

Example: In this example, the **`xsi:schemaLocation`** attribute is attached to each **`Resource`** element.

```
<VOResources>
   <Resource xsi:type="vs:CatalogService"
      xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.0"
      xsi:schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0
                          http://www.ivoa.net/xml/VOResource/v1.0
                          http://www.ivoa.net/xml/VODataService/v1.0
                          http://www.ivoa.net/xml/VODataService/v1.0
                          http://www.ivoa.net/xml/SIA/v1.0
                          http://www.ivoa.net/xml/SIA/v1.0"
      status="active" …>
     …
   </Resource>
   <Resource xsi:type="vs:Registry"
      xmlns:vg="http://www.ivoa.net/xml/Registry/v1.0"
      xsi:schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0
                          http://www.ivoa.net/xml/VOResource/v1.0
                          http://www.ivoa.net/xml/Registry/v1.0
                          http://www.ivoa.net/xml/Registry/v1.0"
      status="active" …>
     …
   </Resource>
</VOResources>
```

View Appendix A.1 for the searching interface WSDL to see the use of the imported VOResource schema placed inside the root element of VOResources.

If a legal search query does not match any resource records, the **`VOResources`** element must contain no **`Resource`** elements. If the input search query is illegal in its syntax or the operation encounters any other errors that prevents returning the requested records, the operation must return an **ErrorResponse** fault, represented by a **`ErrorResponse`** element:

ErrorResponse Element Definition

```
<xs:element name="ErrorResponse">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="errorMessage" minOccurs="1" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

A **`ErrorResponse`** element must include a human-oriented error message describing the nature of the error.

2.1.2 Constraint-based Search Query

The **Search** operation allows clients to retrieve a list of resource descriptions that match constraints of values corresponding to specific metadata from the VOResource schema (and its legal extensions). The operation's input message is defined to have a single part, a **`Search`** element, which contains four child elements that serve as the four input parameters:

Search Element Definition

```
<xs:element name="Search">
   <xs:complexType>
      <xs:sequence>
         <xs:element ref="ri:Where" minOccurs="1" maxOccurs="1" />
         <xs:element name="from" type="xs:positiveInteger"
                     minOccurs="0" maxOccurs="1" />
         <xs:element name="to" type="xs:positiveInteger"
                     minOccurs="0" maxOccurs="1" />
         <xs:element name="identifiersOnly" type="xs:boolean"
                     minOccurs="0" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Search Parameter Elements		
Element	**Definition**	
`Where`	*Value Type:*	an ADQL/x Where clause
	Semantic Meaning:	the constraints to use for selecting matched resource records
	Occurances:	required

from	*Value Type:*	integer
	Semantic Meaning:	the minimum position in the complete set of matched records of the range of records to be returned
	Occurances:	optional; default: 1
max	*Value Type:*	integer
	Semantic Meaning:	the maximum number of matched records to return. The service may choose to return fewer.
	Occurances:	optional; default: the last record in the set
identifiersOnly	*Value Type:*	boolean
	Semantic Meaning:	If true, return the results as a list of identifiers; if false, return as a list of complete resource descriptions.
	Occurances:	optional; default: false

The one required parameter, the **Where** element, is of type **whereType** from the ADQL XML Schema [ADQL] (having the namespace, http://www.ivoa.net/xml/ADQL/v1.0, hitherto referred to using the "**adql:**" prefix; see Appendix A.1) which contains the constraints that specific components of the resource metadata must satisfy. The specific components are named using **adql:Column** elements subject to the following restrictions:

- The **Table** attribute, which is required by the ADQL Schema, should be set to an empty string and must be ignored by the **Search** method implementation.
- The **Name** attribute, which is required by the ADQL Schema, may be set to an empty string or to a short name to serve as an alias for the resource metadatum referred to. This value must be ignored by the **Search** method.
- The **xpathName** attribute must be set to a restricted XPath string, subject to the rules in section 2.2.1. This XPath string identifies the specific VOResource element (or legal extension) within the resource record that is to be constrained.

The **Search** implementation selects matching active or inactive resources as if the ADQL query were being applied to a single table in which each row is a single resource record and the columns include the resource metadata components referred by **xpathName** XML attributes. Matched resource records are then encoded using the VOResource XML Schema (and its legal extensions) according to the specifications given in the Search WSDL and described in

Section 2, and they should include all information available to the registry that is compliant with the VOResources definitions.

2.1.2.1 Restrictions on the use of XPath in ADQL

The value of the **`xpathName`** attribute in any **`adql:Column`** element used within the input to the Search method must be a legal XPath [XPath] string that is restricted in form by the following rules:

- The path points to an element or attribute value within a resource description encoded with the VOResource schema and/or any of its legal extensions.
- When the path points to a specific element, that element must be of a simple type as defined by the XML Schema standard [Schema]
- The path is relative and assumes that the context node is the element that forms the parent of a single resource description (e.g. a **`Resource`** element) and is of type **`vr:Resource`** or one of its legal extensions.
- The path must be composed only of location steps with child axes expressed using the abbreviated syntax for child elements and attributes: elements are referred to simply by their name, and attributes are referred to by their name preceded by an '@' character. Unabbreviated location steps—i.e., those that require the double colon ('::') syntax—are not allowed. All other types of abbreviated axes, including use of double slashes ('//'), single and double periods ('.' and '..'), and wildcards ('*'), are not allowed.
- The path must not include any predicates (i.e., qualifiers expressed using square brackets, '[…]').
- When "xsi" is used as an attribute prefix, it is implicitly assumed to refer to the http://www.w3.org/2001/XMLSchema-instance namespace.

Note:
Because VOResource schema and its legal extensions set `elementFormDefault` and `attributeFormDefault` to both be 'unqualified', prefixes are not normally required to qualify elements and attributes. xsi:type is an exception because it is technically a global attribute, and `attributeFormDefault` does not apply; thus, the prefix would be required in a standard XPath, which is why the last rule is needed. An known exception at this time is the case of the Space-Time Coordinates schema (STC, http://www.ivoa.net/xml/STC/stc-v1.30.xsd) which defines many global elements; thus, technically, these elements would require prefixes, too. This document does not address the problem of querying against these element because it specifies no additional rules for understanding the mapping of other prefixes used in the context of a ADQL query. Because of the complexity of STC, it is not likely that normal ADQL (or XQuery) queries will be particularly useful. Thus, the problems of invoking other prefixes within ADQL and generally querying against STC is left to be solved in a future version of this document.

This restricted form of XPath is intended to make it straight forward to transform the ADQL Where clause to a string-based query—namely SQL and XQuery—through a static mapping from an XPath to an attribute in a local database without parsing the internal content of the path.

Legal Examples:	
curation/publisher	the resource publisher's name
curation/publisher/@ivo-id	the publisher's IVOA identifier
@xsi-type	the specific type of resource
interface/@xsi-type	the specific type of interface
Illegal Examples:	
Resource/title	wrong context node
content	not an element with a simple type
curation/child::publisher	"child::" syntax not allowed
curation//@ivo-id	"//" syntax not allowed
Interface[@xsi-type="vs:WebService"]/accessURL	"[…]" syntax not allowed

2.1.3 Keyword Search Query

The purpose of the **KeywordSearch** operation is to provide a simple way to select resources based on the string values in their resource descriptions. The operation's input message is defined to have a single part, a **`KeywordSearch`** element, which contains five child elements that serve as the five input parameters:

KeywordSearch Element Definition

```
<xs:element name="KeywordSearch">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="keywords" type="xs:string"
                     minOccurs="1" maxOccurs="1" />
         <xs:element name="orValues" type="xs:boolean"
                     minOccurs="0" maxOccurs="1" default="true"/>
         <xs:element name="from" type="xs:positiveInteger"
                     minOccurs="0" maxOccurs="1" />
         <xs:element name="to" type="xs:positiveInteger"
                     minOccurs="0" maxOccurs="1" />
         <xs:element name="identifiersOnly" type="xs:boolean"
                     minOccurs="0" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

KeywordSearch Parameter Elements		
Element	**Definition**	
keywords	*Value Type:*	text string
	Semantic Meaning:	the list of words or phrases to search for within resource descriptions
	Occurances:	Required
orValues	*Value Type:*	boolean
	Semantic Meaning:	if true, apply multiple word/phrase constraints with a logical OR; if fase, apply with a logical AND
	Occurances:	optional; default: true
from	*Value Type:*	integer
	Semantic Meaning:	the minimum position in the complete set of matched records of the range of records to be returned
	Occurances:	optional; default: 1
max	*Value Type:*	integer
	Semantic Meaning:	the maximum number of matched records to return. The service may choose to return fewer.
	Occurances:	optional; default: the last record in the set
identifiersOnly	*Value Type:*	boolean
	Semantic Meaning:	If true, return the results as a list of identifiers; if false, return as a list of complete resource descriptions.
	Occurances:	optional; default: false

The meaning of the last three parameters above and their effect on the output is the same as for the **Search** operation.

The **keywords** parameter is a string that consists of one or more words separated by whitespace characters. The characters that qualify as whitespace are the same as in XML: space (x20), tab (x9), line feed (xA), and carriage return (xD). A phrase is a portion of **keywords** parameter that is enclosed in double quotation marks (e.g. “black hole”); when the phrase is extracted from the parameter, the quotation marks are removed, and a string normalization is applied that ignores leading and trailing whitespaces and treats consecutive whitespaces as a single space. Non-quoted words are extracted from the **keywords** parameter after applying the same string normalization.

The **KeywordSearch** implementation forms a query by, in effect, creating a search constraint for each word or phrase in this parameter. For each active or inactive resource record, each word or phrase is compared against every value for a selected set of resource metadata that includes at minimum the following (drawn from the VOResource schema):

- **`identifier`**: the resource's IVOA identifier
- **`content/description`**: the descriptive summary of the resource
- **`title`**: the resource title
- **`@xsi:type`**: the specific type of resource specified as an extension of the **`Resource`** type
- **`content/subject`**: the subject topics associated with the resource
- **`content/type`**: the general type of resource

The implementer may include additional metadata values in the comparison as they choose (which may include non-string values). It is legal to compare the word with all simple type values in the record. If the word or phrase is contained within one of the selected set of resource metadatum values, the constraint evaluates as TRUE. It is up to the implementer to decide what it means for a word to be considered "contained;" for example, the implementation may also test for related forms of the word. The implementer *may* further parse the phrase into words and include comparison constraints on those individual words. The results of all of the constraint tests (one for each word) are combined logically according to the value of **orValues**: if **orValues** is TRUE, then the resource record is returned when any of the constraints are TRUE, and if it FALSE, then all constraints must be TRUE in order for the record to be returned.

Matched resource records are then encoded using the VOResource XML Schema (and its legal extensions) and should include all information available to the registry that complies with the VOResource standard.

> **Note:**
> This specification provides wide latitude in how the **KeywordSearch** is implemented; thus, different registries may return different results to the same set of input keywords. If precision and consistency in results is important regardless of which registry is queried, users should favor the Search or XQuerySearch operations.

2.2 Resolve Operations

The two resolve operations—**GetResource** and **GetIdentity**—each select and return a single resource record.

2.2.1 Output Format

The two resolve operations share a common output format for returning a single resource record. The response is a SOAP message in compliance with the WSDL document given in Appendix A.1. This message is defined to have a single part: a **`ResolveResponse`** element from the http://www.ivoa.net/wsdl/RegistrySearch/v1.0 namespace. This element in turn wraps a **`Resource`** element of type **`vr:Resource`** or one of its legal extensions. As in with the search operations when the type of the **`Resource`** element is actually an extension of the **`vr:Resource`** type, then the **`Resource`** element MUST specify the specific type using an **`xsi:type`** attribute.

The **`Resource`** element must include a **`xsi:schemaLocation`** attribute in compliance with the XML Schema standard [Schema] to indicate a URL location for the VOResource schema and all of the legal extensions of VOResource that are employed in the response. As with the search operation responses, the URL location for the VOResource core schema (http://www.ivoa.net/xml/VOResource/v1.0) must be set to "http://www.ivoa.net/xml/VOResource/v1.0". For those legal extensions that are standard schemas recognized by the IVOA, the location should be set to the standard location in the IVOA Document repository whose URL begins with "http://www.ivoa.net/xml/".

2.2.2 Identifier Resolution

The purpose of the **GetResource** operation is to provide a simple way to resolve a unique IVOA Identifier to a full resource description. The input message is defined to have a single part, a **`GetResource`** element. This element contains the operation's one input parameter, **`identifier`**, of type **`vr:IdentifierURI`**, encodes an IVOA identifier. The output message contains a single VOResource record whose **`identifier`** element matches the input identifier.

If the registry does not have a resource record (or otherwise cannot access one) with an identifier matching the input parameter, the **GetResource** operation should return a **NotFound** fault message, represented by a **`NotFound`** element:

NotFound Element Definition

```
<xs:element name="NotFound">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="errorMessage" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Including an error message in the fault response is optional but recommended.

If the operation encounters any error that prevents it from determining whether the identifier can be resolved to a description or otherwise prevents the delivery of that description, the operation must return an **ErrorResponse** fault as described in section 2.1.1.

2.2.3 Identity Query

The purpose of the **GetIdentity** operation is to provide a simple way to get the VOResource record that describes the implementing registry itself. A client may then inspect this VOResource record to discover various information about the implemented registry (See section 2.7).

The **GetIdentity** operation takes no parameters. The result is a single VOResource record whose format conforms to the format described in section 2.2.1; however, with this operation, the **`Resource`** element must include an **`xsi:type`** attribute set to indicate the **`Registry`** resource extension type from the VORegistry extension schema (having the namespace http://www.ivoa.net/xml/VORegistry/v1.0, hitherto referred to using the "**`vg:`**" prefix). The recommended value to express this type is "**`vg:Registry`**". The VORegistry schema is described in section 4; see Appendix A.3 for the full XML Schema definition.

2.3 XQuery Search

XQuerySearch is an optional operation of the searching interface which allows clients to form constraint-based queries with greater control than the required **Search** operation. It also allows the client to control the format of the query output; in particular, the client can obtain only the metadata needed rather than the full Resource record. The client can determine if a searchable registry supports this operation by consulting the registry's resource description (see Section 4).

The operation's input message is defined to have a single part, an **`XQuerySearch`** element. This element contains the operation's one input string parameter, **`xquery`**. The value of the **`xquery`** element is a string that states the query and must conform to the XQuery syntax [XQuery]. The XPath strings [XPath] used in the query must be written as if each resource record is stored as a separate document under a root element called **`RootResource`**. The operation implementation may translate the XPath as necessary to reflect the actual storage of the records within the registry.

The operation's output message is also defined to have a single part, an **`XQuerySearchResponse`** element having the following definition:

XQuerySearch Element Definition

```
<xs:element name="XQuerySearch">
   <xs:complexType>
      <xs:sequence>
         <xs:any minOccurs="0" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

The specific XML content of the **`XQuerySearchResponse`** must comply with the format requested by the XQuery input query.

If a registry does not support XQuery–based queries, the **XQuerySearch** operation must be implemented to always return an **UnsupportedOperation** fault message. This message has a single part in the form of an **`UnsupportedOperation`** element:

OperationUnsupported Element Definition

```
<xs:element name="UnsupportedOperation">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="errorMessage" minOccurs="0" maxOccurs="1" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

Including an error message in the fault response is optional but recommended.

All other error conditions encountered by the implementation that prevent the return of the query results should be handled by returning an **ErrorResponse** fault as described in section 2.1.1.

3 Harvesting

Harvesting is the mechanism by which a registry can collect resource descriptions from other registries. This mechanism is used by full searchable registries to aggregate resource descriptions from many publishing registries. It can also be used to synchronize two registries to ensure that they have the same contents. This section defines the **IVOA Harvesting Interface**. Client applications that make use of this interface are referred to as **harvesters.** Those registries that declare themselves as harvestable (section 3.2) must comply with the specification described in this section. As Noted in the Abstract Registries may become harvestable by implementing the HTTP GET interface of OAI as an alternative to the SOAP Service interface described.

3.1 Harvesting Interface

The harvesting interface builds on the Web Service version of the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [OAI]. In particular, all IVOA Registries that support the Harvesting Interface must be compliant with the Web Service version of OAI-PMH. Compliance with this base standard allows IVOA registries to be accessed by applications from outside the IVOA community.

> **Editor's Note:**
> OAI does not currently support an official Web Services version of PMH. One of the purposes of the development of this standard is to drive the evolution of the OAI standard which has demonstrated to be a highly effective harvesting protocol across a broad continuum of communities.

In addition to OAI-PMH compliance, this specification defines an additional set of OAI-PMH-compliant requirements and recommendations which are described in sections 3.1.1 through 3.1..6 below.

3.1.1 A Summary of the OAI Web Service Interface

The Web Service version of OAI-PMH is defined by:

- The OAI-PMH v2.0 specification (http://www.openarchives.org/OAI/openarchivesprotocol.html) which defines
  - the meaning and behavior of the six harvesting operations, referred to as "verbs",
  - the meaning of the input arguments for each operation, and
  - the XML Schema used to encode response messages.
- The OAI-PMH Web Service Definition Language (WSDL) document (see Appendix A.2) which defines
  - the six "verbs" defined as Web Service operations
  - SOAP encoding of the operation input arguments and response messages, based on the OAI-PMH XML Schema.

In summary, the OAI-PMH standard defines six operations:

Identify: provides a description of the registry
ListIdentifiers: returns a list of identifiers for the resource records held by the registry.
ListRecords: returns all Resource records in the registry. Registries may use the set "ivo_managed" to get Resource records managed by this particular registry..
GetRecord: returns a single resource description matching a given identifier.

ListMetadataFormats: returns a list of supported formats that the registry can use to encode resource descriptions upon a harvester's request.

ListSets: return a list of category names supported by the registry that harvesters can request in order to get back a subset of the descriptions held by the registry.

The **ListRecords** and **GetRecord** operations return the actual resource description records held by the registry. These descriptions are encoded in XML and wrapped in a general-purpose envelope defined by the OAI-PMH XML Schema (namespace http://www.openarchives.org/OAI/2.0).

Through the operations' arguments, OAI-PMH provides a number of useful features:

- *Support for multiple return formats*. As suggested by the **ListMetadataFormats** operation, a harvester can request the format resource descriptions are encoded in.
- *Harvesting by date*. The **ListIdentifiers** and **ListRecords** operations both support "from" and "until" date arguments. The "from" argument can be used to retrieve records that have changed since the last harvest.
- *Harvesting by category*. The **ListIdentifiers** and **ListRecords** operations both support a "set" argument for retrieving resources that are grouped in a particular category. Resource records may belong to multiple groups.
- *Marking records as deleted*. Registries may mark records as deleted so that harvesters may remove access to them from their applications.
- *Support for resumption tokens.* If a request results in returning a very large number of records, the registry can choose to split the results over several calls; this is done by passing a resumption token back to the harvester. The harvester uses it to retrieve the next set of matching results.

Editor's Note:
The Web Service version of the OAI-PMH protocol has been designed to match the behavior and functionality of the original "HTTP GET"-based version as much as possible. One reason for this is to make it as straightforward as possible to build bridges between implementations of both types and to build off the existing OAI software.

Note:
It is important to note that the OAI-PMH interface is not intended to be a general search interface. The filtering capabilities described above are just enough to support intelligent harvesting between registries. Most end-user applications will use the search interface described in sections 3 and 4 to retrieve resource descriptions.

The Web Service or SOAP version of OAI-PMH augments the original specification with a standard Web Service Definition Language (WSDL) document which is listed in H.2. Harvestable registries complying to the SOAP version of OAI-PMH must emit a copy of the WSDL document, with a service element appropriate for the local endpoint URL added in, in response to a call to the Web Service URL with the standard "?wsdl" argument. All six of the standard operations must be implemented. Additional, non-standard operations may be added; however, the definition of the six standard operations, along with the definition of their inputs and outputs, must not be altered. The interface is recognized as the OAI-PMH standard when the default namespace for the WSDL matches "http://www.ivoa.net/wsdl/oai.wsdl" exactly.

Editor's Note:
The namespace for the WSDL would presumably be changed to something like http://www.openarchives.org/OAI-WS/1.0/ if and when it is accepted by the OAI community.

The subsequent sections below describe how the standard OAI-PMH features are used to support IVOA-specific functionality.

3.1.2 Metadata Formats for Resource Descriptions

All IVOA registries that support the Harvesting Interface must support two standard metadata formats: the OAI Dublin Core format (mandated by the base OAI-PMH standard) and the IVOA VOResource metadata format [http://www.ivoa.net/xml/VOResource/v1.0].

The VOResource metadata format will have the metadata prefix name "ivo_vor" which can be used wherever an OAI-PMH metadata prefix name is supported (see OAI standard, section 3.4, "metadataPrefix and Metadata Schema"). The format uses the VOResource core XML Schema with the namespace http://www.ivoa.net/xml/VOResource/v1.0 (referred hereto with the namespace prefix "vr") along with any legal extension of this schema (including the IVOA standard extensions) to encode the resource descriptions within the OAI-PMH **metadata** tag from the OAI XML Schema (namespace http://www.openarchives.org/OAI/2.0, hereto referred by the namespace prefix "oai"). The format is specifically defined as a **vr: Resource** element as the sole child of the **oai:metadata** element. In compliance with the VOResource schema and any legal extensions.

Editor's Note:
If and when the VOResource schema evolves to a new version, this standard must be updated accordingly. Thus, this definition is locked to particular version of the VOResource, so saying that a registry is compliant with vX.X of this document implies a specific version of VOResource.

Note:
It is possible that the **vr:Resource** extension returned is unrecognized by the harvester. The harvester must deal with this possible outcome by handling and storing of extensions or by ignoring vr:Resource metadata.

Editor's Note:
A "standard resource extension" will be defined as a type of **vr:Resource** in a schema that has been approved as an IVOA Recommendation. At this writing, no VOResource schemas have reached this state, so for the purposes of prototyping, a "standard resource extension" will refer to any **vr:Resource** type from the following schemas:

- VOResource: http://www.ivoa.net/xml/VOResource/v1.0
- VORegistry: http://www.ivoa.net/xml/VORegistry/v1.0
- VODataService: http://www.ivoa.net/xml/VODataService/v1.0
- ConeSearch: http://www.ivoa.net/xml/ConeSearch/v1.0
- SIA: http://www.ivoa.net/xml/SIA/v1.0

The OAI Dublin Core format, with the metadata prefix of "oai_dc", is defined by the OAI-PMH base standard and must be supported by all OAI-PMH compliant registries. This document does not specify how a record in the VOResource format maps into the OAI Dublin Core format; however, the IVOA Registry Working Group may recommend such a mapping based on the IVOA Resource Metadata standard [ref].

Harvestable registries may support other metadata formats. The **ListMetadataFormats** must list all names for formats supported by the registry; this list must include "ivo_vor" and "oai_dc".

3.1.3 Identifiers in OAI Messages

In accordance with the OAI-PMH standard, an OAI-PMH XML envelope that contains a resource description must include a globally unique URI that identifies that resource record. This identifier must be the IVOA identifier used to identify the resource being described and cited as the value of the **`vr:identifier`** resource metadatum.

Note:
This specification does not follow the recommendation of the OAI-PMH standard with regard to record identifiers. OAI-PMH makes a distinction between the resource record containing resource metadata and the resource itself; thus, it recommends that the identifier in the OAI envelope be different from the resource identifier. In particular, the former is the choice of the publishing registry. This allows one to distinguish resource descriptions of the same resource from different registries, which in principle could be different.

In the VO, because it is intended that resource descriptions of the same resource from different registries should not differ, there is not a strong need to distinguish between the resource and the resource description. By making the resource and resource record identifiers the same, it makes it much easier to retrieve the record for a single resource via **GetRecord**, regardless of

which registry is being queried. Otherwise—when the registry chooses the record identifier—a client will not *a priori* know the record identifier for a particular resource, and so it is left to call **ListRecords** and search through the metadata of all the records itself to find the one of interest. In contrast, IVOA identifiers are intended to be a cross-application way of referring to a resource, and thus when a client wants only a single specific resource record, it is very likely that it would know the resource identifier when making a call to the **GetRecord** operation.

3.1.4 Required Records

This section describes the records that a harvestable IVOA Registry must include among those it emits via the OAI-PMH operations.

The harvestable registry must return one record that describes the registry itself as a whole, and the "ivo_vor" format must be supported for this record. This record is included in the **Identify** operation response (see section 3.1.5). When encoded using the "ivo_vor" format, the vr:Resource returned must be of an extension of vg:Registry from the VORegistry schema (namespace http://www.ivoa.net/xml/VORegistry/v1.0; hereto referred by the "vg" namespace prefix). The record must include a **vg:managedAuthority** for every Authority Identifier [ref IVOA Identifiers] that originated at that registry. The registry may contain other registry records for other registries it knows about; use of **vr:Resource** elements other than **vg:Registry** to describe these other registries is strongly discouraged.

The harvestable registry must return exactly one record in "ivo_vor" format for each Authority Identifer listed as a **vg:managedAuthority** in the **vg:Registry** record that describes that registry. When encoded in the "ivo_vor" format, the type of **vr: Resource** must be an **vg:Authority** type.

3.1.5 The Identify Operation

The **Identify** operation describes the harvestable registry as a whole. The response from this operation must include all information required by the OAI-PMH standard. In particular, it must include a **oai:baseURL** element which must refer to the base URL to the Web Service endpoint (i.e. the URL used to retrieve the WSDL document via the standard URL suffix, "?wsdl") unless the HTTP-GET is implemented by the Registry see below note.

Note:
A traditional "HTTP GET" implementation of OAI-PMH that serves as a bridge to Web Service implementation must transform the value of the **oai:baseURL** element to refer to itself rather than the delegate Web Service.

The **Identify** response must include a **oai:description** element containing a single vr:Resource of type **vg:Registry**. The content of **vg:Registry** type must be the registry description of the harvestable registry itself..

3.1.6 IVOA Supported Sets

Sets, as defined in the OAI-PMH standard, "[are] an optional construct for grouping items for the purpose of selective harvesting" (see the OAI-PMH standard, section 2.6). Harvestable IVOA registries are free to define any number of custom sets for categorizing records. The OAI-PMH standard allows a record to be a member of multiple sets. This document defines a set of reserved set names with special meanings. Their names all start with the characters "ivo_"; implementers must not define their own set names that begin with this string. Support for one of the reserved sets, "ivo_managed," is required by this specification; thus, when applied to IVOA-compliant harvestable registries, support for sets is not optional.

This specification optionally defines a set for each of the IVOA standard extensions to the vr:Resource as well as the vr:Resource element itself. The set name is formed by prepending "ivo_" to the local element name for the resource extension. (For example, a set defined for **vg:Registry** is named "ivo_Registry".) A request for records in such a set will return records whose "ivo_vor" rendering features the associated resource extension. (For example, requesting the "ivo_Registry" set will return all records whose "ivo_vor" form has a **vg:Registry** type of the **vr:VOResource** element.) Requests for the "ivo_Resource" set (if supported) should return records whose "ivo_vor" form has a **vr:Resource** with no type extension hence no xsi:type defined.. Harvesting registries should support all sets associated with IVOA standard **Resource** extensions. Requests for these sets that are not supported should return an error (in accordance with the OAI-PMH standard), even if such records exist.

The "ivo_standard" optional set refers to all of the IVOA reserved sets that correspond to IVOA standard **Resource** extensions that are supported by the registry. Harvesters may request this set to guarantee getting back records it can fully parse. Harvesting registries must support this set.

The "ivo_managed" required set refers to all records that originate from the queried registry. That is, those records that were harvested from other registries are excluded. The IVOA Resource identifiers given in the records must have an Authority Identifier that matches on one of the **vg:managedAuthority** values in the **vg:Registry** record for that registry. Full searchable registries may use this set to avoid getting duplicate records when harvesting from many registries.

All sets that are supported by the harvestable registry, including the one required set, must be listed in the response to the **ListSets** operation in compliance with the OAI-PMH standard. Appendix A.3 lists the recommended set descriptions which can be returned by the **ListSets** operation for the IVOA reserved set names.

3.2 Harvesters

A registry that collects resource descriptions from other registries through the Harvesting Interface defined above in section 3.1 is referred to as a **harvester registry.** See Section 4 for sample vg:Registry extension for deterimination of interface URL and method to which to call the registry such as HTTP-GET or SOAP. Harvesters 'must' be able to harvest via SOAP or HTTP-GET interface. If both SOAP and HTTP-GET interfaces are defined in the vg:Registry extension then the harvester has an option of which interface to call.

Note:
The Registry of Registries is hosted by IVOA for the discovery of other Registry types "vg:Registry" records. The Registry of Registries known as RofR only stores vg:Registry records and implements the Harvesting interface only. Full Registries should harvest RofR for discovery of new and/or updated Registries. See IVOA Note section on more detailed description of RofR.

4 Registering Registries

This specification defines a VOResource extension schema called VORegistry that can be used to specifically describe a registry and its support for the registry interface described in this document. These descriptions can be stored as resource records in registries. The schema is also used to register a *naming authority*—a publisher who claims ownership of an *authority identifier* from which IVOA identifiers may be created [Identifiers]. A publishing registry is said to exclusively *manage* a naming authority on behalf of the owning publisher; this means that only that registry may publish records with IVOA identifiers using that authority identifier.

4.1 Namespace and Schema Location

The

4.2 The Registry Resource Extension

The

4.3 The Authority Resource Extension

The

[Points to cover:

- *VORegistry used to register registries*
- *Definition of extension metadata: managedAuthority*
- *Restrictions on describing capabilities*

- *Example record*

]

```
<resource xsi:type="vg:Registry"
          xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
          xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0
                              VOResource-v1.0.xsd
                              http://www.ivoa.net/xml/VORegistry/v1.0
                              VORegistry-v1.0.xsd">
    <title>IVOA Registry of Registries sample entry</title>
    <shortName>RofR</shortName>
    <identifier>ivo://ivoa/registry</identifier>
    <curation>
      <publisher>
         IVOA
      </publisher>
      <creator>
        <name>Ray Plante</name>http://www.ivoa.net/
      </creator>
      <date>2006-08-08</date>
      <contact>
        <name>Ray Plante</name>
   <email>rplante@ncsa.uiuc.edu</email>
      </contact>
    </curation>
    <content>
      <subject>registry repositories</subject>
      <description>
         This registry provides information regarding other registries.
      </description>
      <referenceURL>http://www.ivoa.net</referenceURL>
      <type>Registry</type>
      <contentLevel>Research</contentLevel>
    </content>
    <capability xsi:type="vg:Harvest"
                standardID="ivo://ivoa.net/std/Registry">
       <interface xsi:type="vg:OAIHTTPGet" role="std">
          <accessURL> http://www.ivoa.net/cgi-bin/rofr/oai.pl
</accessURL>
       </interface>
             <interface xsi:type="vg:OAISOAP" role="std">
          <accessURL> http://www.ivoa.net/rofr/RegistryHarvest
</accessURL>
       </interface>
       <maxRecords>100</maxRecords>
    </capability>
   <!-- Uncomment this section for the Search Interface
    <capability xsi:type="vg:Search"
                standardID="ivo://ivoa.net/std/Registry">

       <interface xsi:type="vr:WebService" role="std">
         <accessURL> http://nvo.ncsa.uiuc.edu/cgi-bin/nvo/search.pl
</accessURL>
```

```
        </interface>
        <optionalProtocol>XQuery</optionalProtocol>
        <maxRecords>0</maxRecords>
    </capability>
    -->
    <full>false</full>
    <managedAuthority>ivoa</managedAuthority>
    <managedAuthority>ivoa.net</managedAuthority>
</resource>
```

Appendix A.1 Web Services Definition Language Document for Search Interface

RegistrySearch Interface WSDL

Appendix A.2 Web Services Definition Language Document for the Harvesting Interface

Currently See: http://www.ivoa.net/twiki/bin/view/IVOA/RegistryInterface

Appendix A.3 VORegistry: the VOResource Extension Schema for Registering Registries

Currently See: http://www.ivoa.net/twiki/bin/view/IVOA/RegistryInterface

References

[WSDLv1.1] Christensen, E., Curbera, F., Meredith, G., & Weerawarana, S. 2001, Web Services Description Language v1.1, W3C Note 15 March 2001, http://www.w3.org/TR/wsdl/

[Hanisch 2004] Hanisch, R. (ed.) et al. 2004, *Resource Metadata for the Virtual Observatory*, IVOA Recommendation, http://www.ivoa.net/Documents/latest/RM.html

[Plante 2003] Plante, R., Greene, G., Hanisch, R., McGlynn, T., O'Mullane, W., & Williamson, R. 2003, in ASP Conf. Ser., Vol. 314 Astronomical Data Analysis Software and Systems XIII, eds. F. Ochsenbein, M. Allen, & D.

Egret (San Francisco: ASP), 585, http://www.adass.org/adass/proceedings/adass03/O7-1

[VOResource] Plante, R. et al. 2006, *VOResource: an XML Encoding Schema for Resource Metadata*, v1.00, IVOA Working Draft, http://www.ivoa.net/Documents/WD/ReR/VOResource-20060530.html

[ADQL] Ohishi, M. et al. 2004, *Astronomical Dataset Query Language*, IVOA Working Draft (internal), http://www.ivoa.net/internal/IVOA/IvoaVOQL/WD_ADQL-0.9.pdf

[XPath] Clark, J. and DeRose, S. 2001, XML Path Language (XPath) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath/

[Schema] Fallside, D, and Walmsley, P. 2004, XML Schema Part 0: Primer Second Edition, W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-0/

[OAI] http://www.openarchives.org/OAI/openarchivesprotocol.html