International

Virtual

Observatory

Alliance

IVOA Registry Interfaces Version 0.8

IVOA Working Draft 2004 June 16

This version:
0.1 http://www.ivoa.net/internal/IVOA/RegistryInterface/IVOARegistryIntervace-0.1.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

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 2005 July 25~~2005 March 16~~.

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 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.

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 et al. 2004], 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. The IVOA Registry relies on the Web Service interface version, which defines what arguments are required for harvesting and search of the resources, as well as the messaging mechanism. The IVOA Registry interface includes the following definitions:

- The meaning and behavior of three types of search and six 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 IVOA Registry collects the lists of resource descriptions that match specific criteria via the search operation. The IVOA Registry Interface consists of two search operations:

- **Search** searches the Registry in order to obtain the VO resources.
- **KeywordSearch** is a helper query based on a set of key words.

The registry can collect resource records from other registries using one of the six operations, which support resource harvesting. The operations listed below are described in more detail in 3.1.1. The most important harvesting operation is the **ListRecords**, which collects the descriptions of the resource based on constraints such as date and time period. ListRecords interface provides for the retrieval of all resources that are managed by its corresponding Registry plus resources of the type Registry are also harvestable by means of the ListRecords interface, The complete list of harvesting operations is shown below and their implementation follows the OAI standards:

- **Identify**
- **ListIdentifiers**
- **ListRecords**
- **GetRecords**
- **ListMetadataFormats**
- **ListSets**

The operations that return resource descriptions do so using the VOResource XML Schema [Plante et al. 2004] and any of its legal extensions.

2 Searching

The required search operations—**Search, KeywordSearch, GetResource** return a list of one or more resource descriptions held by the ~~registry that~~

~~match~~registry that matches the input selection criteria. The three search operations respectively support three types of searching:

- **Constraint-based Searching** for resources by means of a query using the Astronomical Data Query Language (ADQL) [ADQL],
- **Keyword-based Searching** for resources whose descriptions contain words in an input string.
- **Identifier-based Searching for returning one and only one resource based off of a uniquie identifier.**

> **Editor's 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 two operations are defined by WSDL document given in Appendix A.1. Searchable registries must implement all the operations.

All the~~e~~ operations share a common output format for the resource records that match the search criteria. These records are encoded in XML and wrapped in a root element called **VOResources**. The resource records are represented as child **Resource** elements of type **Resource** from the VOResource XML Schema (having the namespace, http://www.ivoa.net/xml/VOResource/v0.10, 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 **Resource** element MUST specify the specific type using an **xsi:type** attribute in compliance with the XML Schema standard [Schema]. The location of the schema for the vr:Resource must be always included in the output resource record identified in the xs:schemaLocation attribute. View Appendix A.1 for the WSDL to see the use of the imported Resource schema placed inside the root element of VOResources.

> **Editor's Note:**
> This document complies with the current version of the VOResource specifications. When a new VOResource schema evolves, this document should be updated accordingly.

The registry interface must implement both Search operations in order to comply with communication and interaction standards for a Web Service. Searchable registries compliant with the augmented SOAP must return a copy of the WSDL document, with a service element appropriate for the local endpoint URL appended in response to a call to the Web Service URL with the standard **"?wsdl"** argument. Additional operations may be added, however original search operations and their arguments and outputs must not be altered.

2.1 Handling large volume of data

Two of the required search interfaces –Search, KeywordSearch return a list of one or more resource descriptions which can comprise large data volumes depending on the query. These two interface methods can be run with a set of optional parameters that help reduce the amount of data. These are a paginate scheme and an identifier scheme.

A client may wish to use none, all, or a mix of these optional parameters.

The optional parameters:

- from – starting point of a returned list of Resources. If not specified the default is “1”.
- to – ending point of a list of Resources. If not specified then the end of the list of Resources or the registries implementation which ever comes first.
- identifiersOnly – Boolean to indicate return only identifiers.

2.1.1 Paginate Scheme

Clients are allowed to provide the parameter showing the starting number of a record from the selected set, as well as to supply an ending record number.

The operation is capable of returning query results incrementally. The client can view the results of an ADQL query or Keyword search starting from a certain record number to a specific record number. The output of resource records identifies the starting and the ending record numbers displayed on the page as well as a Boolean attribute showing whether more records/result pages are available further. The search interface with incremental result returns resource records in sets identified by the attributes:

from – shows the starting number of the returned resource record shown by the set

numberReturned – the total number of returned resource records.

more – a Boolean value. True shows that more results are available, false identifies the end of the returned search results.

Example:

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

The client is given an option of choosing the number of records to be returned, however, the search service allows for the implementer to establish a default setting for the result return. Therefore although the client may not specify values of from and to parameters in the search, the client may get an incremental records output depending on the implementation.

2.1.2 Identifiers scheme

The Search and KeywordSearch interface now supports an option that allows returning only the identifiers of the selected records whereby decreasing the search process and the output space. To take advantage of this option the client must supply a true value to the parameter identifiersOnly.

~~2.1~~2.2 Constraint-based Search Query

The **Search** method allows clients to retrieve a list of resource descriptions that match constraints of values corresponding to specific metadata in VOResource schema (and its legal extensions).

IVOA searchable registries must implement the **Search** interface~~method~~, which takes one required parameter, a **`Where`** element of type **`whereType`** from the ADQL XML Schema [ADQL] (having the namespace, http://www.ivoa.net/xml/ADQL/v0.8, 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 2.1.1. 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 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.

~~2.1.1~~2.2.1 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 definied 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 simply by their name, and attributes are referred 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, '[…]').
- All element nodes in the xpathname must have an associated prefix.
- All prefixes in the xpathname must have an associated namespace in the 'Where' element.

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:	
Title	the resource's title
vr:curation/vr:publisher	the resource publisher's name
vr:curation/vr:publisher/@ivo-id	the publisher's IVOA identifier
@xsi-type	the specific type of resource
vr: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

Editor's Note:
A rule regarding the handling of namespaces is needed.

~~2.2~~2.3 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 output of the operation is a set of matched resource descriptions in the same format as from the **Search** operation and specified in section 2.

IVOA searchable registries must implement the **KeywordSearch(String words, Boolean orValue)** method, which has two required parameters:

- **String words:** The first parameter is a parameter of type **`xs:string`** that consists of at least 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).
- **Boolean orValues:** The second parameter is of type **`xs:boolean`** which determines the logical operand to be applied. Either "AND" or "OR" operand can be applied when querying with more than one word. If this parameter has a TRUE value, then *any* of the words must appear in the resource description in order for the resource to be returned. If the second parameter is FALSE, then *all* of the words must appear in the resource record in order for the record to be returned.

The **KeywordSearch** implementation forms a query by, in effect, creating a search constraint for each word in the **words** parameter. Words are extracted from the **words** parameter after a normalization that ignores leading and trailing whitespaces and treats consecutive whitespaces as a single space. For each resource record, each word is compared against every value for a selected set of resource metadata that includes at minimum the following:

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

- **`vr:type`**: the general type of resource

What other values the word is compared against (which may include non-string values) is a choice of the implementer. It is legal to compare the word with all simple type values in the record. If the word 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 related forms of the word. 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 definitions of the VOResources.

2.4 Single Resource Search Query

The purpose of the **getResource** operation is to provide a simple way to select a single resources based on the string value of its identifier. The output of the operation is a single record matched to the resource identifier.

- IVOA searchable registries can optionally implement the **getResource(String identifier)** method, which has one parameter of type **`xs:string`** that has the identifier of the resource record in order for the record to be returned.

During the search operation the resource record metadata is compared against the value fof the IVOA resource identifier vr:identifier. The result of the single resource search query is the selected resource metadata.

2.5 Xquery Search

The purpose of the XquerySearch operation is to provide a more convenient way of searching the hierarchal xml schema and provide the client a way of obtaining the subelement(s) they need and not the full Resource each time. The output of this operation is determined by the XQuery input.

To determine if a registry supports the XQuery interface a client must inspect the WSDL of the searchable registry or the Registry type "xsi:type='vg:Registry' for a <hasXQuery> element.

IVOA searchable registries may implement the **XquerySearch(String xquery)** method which has one parameter:

String xquery: The first and only parameter is a string that conforms to xquery syntax, see information on xquery here: http://www.w3.org/XML/Query

~~2.3Finding Other Registries~~

~~**Editor's Note:**~~
~~Would this operation be more useful if it were defined to return only those registries that it harvests from? Simply finding all registries can be accomplished with a simple ADQL query, as shown below.~~

~~The implementation must return the same response as the **Search** operation (section 2.1) given the following ADQL where clause (expressed here in ADQL/s format [ADQL]):~~

```
   ~~@xsi:type='vg:Registry'~~
```

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.

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 a set of resource descriptions. It returns all resources managed by the corresponding registry, as well as the resources of the Registry type. This operation is intended to be used by other searchable registries to locate harvestable registries.

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~~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 straight-forward 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/v0.10].

The VOResource metadata format will have the metadata prefix name "ivo_vor" which can be used where ever 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/v0.10 (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:VOResource** element as the sole child of the **oai:metadata** element. In compliance with the VOResource schema, the child of the **vr:VOResource** element may be any legal extension of the **vr:Resource** element (i.e. that is, an element that is in the same substitution group as **vr:Resource**), except where otherwise restricted by this document.

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. See section 3. for details about how a harvester may use sets—particularly the "ivo_standard" set—to guarantee the return of records that can guarantee support for.

Note:
Use of a **vr:Resource** extension where a IVOA standard resource extension exists is strongly discouraged for records in the "ivo_vor" format. Implementers should consider defining a custom metadata format name to encode using non-standard **vr:Resource** extensions.

Editor's Note:
A "standard resource extension" will be defined as an element in the **vr:Resource** substitution group in a schema that has been approved as an IVOA Recommendation. At this writing, no VOResource schemas have not reached this state, so for the purposes of prototyping, a "standard resource extension" will refer to any element in the **vr:Resource** substitution group from the following schemas:

- VOResource: http://www.ivoa.net/xml/VOResource/v0.10
- VOCommunity: http://www.ivoa.net/xml/VOCommunity/v0.3
- VORegistry: http://www.ivoa.net/xml/VORegistry/v0.3
- VODataService: http://www.ivoa.net/xml/VODataService/v0.5

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 sole child of the **vr:VOResource** element must be a **Registry** element from the VORegistry schema (namespace http://www.ivoa.net/xml/VORegistry/v0.3; 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.

> **Editor’s Note:**
> The registry description record will also need to support additional metadata which is not currently defined in the VORegisry schema. An explanation of the required metadata should go here.
>
> Among the needed metadata is the base URL to use as the service access point. Another is an indication of whether the harvestable registry supports the Web Service or original HTTP GET versions of OAI-PMH.

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 sole child of the **vr:VOResource** element must be an **vg:Authority** element.

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”).

> **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 **vg:Registry** element. This element must contain the proper namespace definitions for the record. The content of **vg:Registry** element must be the registry description of the harvestable registry itself. Other **oai:description** elements are allowed; however, there may only be one containing the **vg:Registry** element.

3.1.6 IVOA Supported Sets

Sets, as defined in the OAI-PMH standard, “is 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 two of the reserved sets, “ivo_standard” and “ivo_managed,” are required by this specification; thus, when applied to IVOA-compliant harvestable registries, support for sets is not optional.

This specification implicitly defines a set for each of the IVOA standard extensions to the vr:Resource element 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** element as the child of the **vr:VOResource** element.) Requests for the “ivo_Resource” set (if supported) should return records whose “ivo_vor” form has a **vr:Resource** element as the child of the **vr:VOResource** element. 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” 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” 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 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 two required sets, 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 are referred to as a **harvester registry.** A registry that operates in this mode should implement the Harvester Interface which provides a way for harvestable registries to request to be harvested from (e.g. because updates have recently occurred). A registry that conforms to this interface should indicate so within its registry description using the metadata provided in the VORegistry schema (namespace, http://www.ivoa.net/xml/VORegistry/v0.3). A harvester registry that does not support this interface is understood has supporting some other mechanism for deciding which registries to harvest from and when to harvest.

The Harvester Interface is defined by the single operation WSDL listed in Appendix A.4. The operation called "harvest" is called to request that the harvestable registry referred to in the inputs be harvested from at the next earliest convenience of the harvester. The harvester, upon receipt of this request, has several options regarding when the harvesting will begin; it may choose:

1. to harvest immediately,
2. to postpone harvesting to a later time (e.g. to synchronize with its own update cycle), or
3. to not harvest at all (e.g. because the inputs do not meet its criteria for harvesting).

The operation's input arguments have the following meaning:

ivo-id: the IVOA Identifier for the harvestable registry requesting a harvest.

harvestingType: This indicates whether the harvesting registry supports the Web Serivce or the traditional HTTP Get version of the OAI protocol.

baseURL: the base URL for the service. Whether this is to interpreted as a Web Service endpoint or the base URL in the sense of the traditional "HTTP GET" version of OAI-PMH depends on the value of harvestingType.

lastUpdate: the date of the last update made to any of the records held by the caller (optional).

Note:
It is recommended that a harvester registry limit how frequently it re-harvests from a

harvestable registry; that is, if the harvester has harvested from a registry before, it should choose option (2) above. This will prevent multiple calls to the harvest operation in quick succession from triggering multiple, unnecessary harvesting processes. Instead, it should queue the request and ignore subsequent requests until the initial harvesting is complete.

4 Registering Registries Harvesting

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

Appendix A.2 Web Services Definition Language Document for OAI-PHM

Appendix A.3 Recommended Descriptions for IVOA Reserved Sets

Appendix A.4 Web Services Definition Language Document for the Harvesters Interface

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`

[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