Registry Interfaces

Version 1.1

IVOA Proposed Recommendation 2017-10-10

Working group
    Registry

This version
    http://www.ivoa.net/documents/RegistryInterface/20171010

Latest version
    http://www.ivoa.net/documents/RegistryInterface

Previous versions
    IVOA Registry Interfaces 1.0, IVOA Recommendation 2009-11-04

Author(s)
    Theresa Dower, Markus Demleitner, Kevin Benson, Ray Plante, Elizabeth Auden, Matthew Graham, Gretchen Greene, Martin Hill, Tony Linde, Dave Morris, Wil O’Mullane, Guy Rixon, Aurélien Stébé, Kona Andrews

Editor(s)
    Theresa Dower, Markus Demleitner

Version Control
    Revision 4542, 2017-10-20 12:56:10 -0300 (Fri, 20 Oct 2017)
    https://volute.g-vo.org/svn/trunk/projects/registry/RegistryInterface/RegistryInterface.tex
Abstract

The VO Registry provides a mechanism with which VO applications can discover and select resources that are relevant for a particular scientific problem. This specification defines the operation of this system. It is based on a general, distributed model composed of searchable and publishing registries, as introduced at the beginning of this document. The main body of the specification has three components: (a) an interface for harvesting publishing registries, which builds upon the Open Archives Initiative Protocol for Metadata Harvesting. (b) A VOResource extension for registering registry services and description of a central list of said IVOA registry services. (c) A discussion of the Registry of Registries as the root component of data discovery in the VO.

Status of This Document

This is an IVOA Proposed Recommendation made available for public review. It is appropriate to reference this document only as a recommended standard that is under review and which may be changed before it is accepted as a full Recommendation.

A list of current IVOA Recommendations and other technical documents can be found at http://www.ivoa.net/documents/.

Contents

1 Introduction 3
   1.1 Registry Architecture and Definitions ................. 4
   1.2 The Registry Interface within the VO Architecture ... 5

2 The IVOA Harvesting Interface 6
   2.1 The OAI Protocol for Metadata Harvesting ............ 7
   2.2 Metadata Formats for Resource Descriptions .......... 8
   2.3 Identifiers in OAI Messages .......................... 9
   2.4 Required Records ..................................... 10
   2.5 The Identify Operation ............................... 10
   2.6 IVOA Supported Sets ................................. 11
   2.7 Time Granularity .................................... 11

3 Registering Registries 11
   3.1 The Authority Resource Extension and the Publishing Process ... 13
   3.2 Describing Registries with the Registry Resource Extension ... 13
   3.3 The Search Capability ............................... 14
   3.4 The Harvesting Capability ............................ 14
1 Introduction

In the Virtual Observatory (VO), registries provide a means for discovering useful resources, i.e., data and services. This discovery takes place by searching within structured descriptions of resources, the resource records, authored by the data providers. In order to avoid a single point of failure for the VO, the Registry is distributed. This means that each data provider can run a service injecting resource records into the Registry (a “publishing registry” as defined below), and anyone can run services that allow global discovery (a “searchable registry” as defined below).

To enable this, common mechanisms for registry communication and interaction are required. This document therefore describes the standard interfaces that enable interoperable registries. Through these interfaces, registry builders have a common way of sharing resource descriptions with users, applications, and other registries.

This specification does not cover interfaces for global discovery, which are the subject of other IVOA standards. Also, service operators are free to build interactive, end-user interfaces in any way that best serves their target community.

While the architecture and standard processes for distributed registry search and maintenance remain similar to this document’s version 1.0 and it remains backward compatible, there is a significant philosophical change in this update.
Most importantly, a defined search interface using SOAP technology is no longer recommended, and a Table Access Protocol service using a registry data model is encouraged for search, with the understanding that new technologies will continue to be developed and adopted.

1.1 Registry Architecture and Definitions

A registry is first a repository of structured descriptions of resources. In the VO, a resource is defined by the IVOA Recommendation “Resource Metadata for the Virtual Observatory” (Hanisch and IVOA Resource Registry Working Group et al., 2007), henceforth referred to as RM, as being

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 classes of resources. The most important type of resource to applications is a service that actually does something. A registry (lower case), then, is “a service for which the response is a structured description of resources” (RM).

This specification is based on the general IVOA model for registries (Plante and Greene et al., 2004), which builds on RM’s model for resources. In this 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 partial.

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 spared from having to visit many different sites 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 allow other VO components to gather metadata on the data center’s published services. 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: storage, management, and indexing of the records can be simpler, as there is no need to support a search interface facilitating complex
discovery queries. While a searchable registry in practice will necessitate the use of a database system, a simple publishing registry may get by storing 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. Several such registries exist, run by various VO projects. A local registry, on the other hand, contains only a subset of known resources. While for publishing registries this subset usually is defined by what services are maintained by the registry’s operator, other selection criteria are conceivable. For instance, the IVOA’s Education IG is considering running a registry only containing resources manually selected for suitability for primary and secondary education.

As mentioned above, harvesting is the mechanism by which a registry can collect resource records from other registries. It is used by full registries to aggregate resource records from 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 term harvester refers to the registry that wishes to receive records (usually a full searchable registry); it sends its request to the harvestee (usually a publishing registry), which responds with the records. Harvesting is a much simpler process than a fully-featured search interface, as only very few constraints need to be supported and only full records are being transmitted in responses. Consequently, different protocols are employed for the two types of registry operations.

In this text, “registry” in lower case refers to concrete services, while “Registry” (or “VO Registry”) in upper case refers to the combination of the set of all resource records and the interfaces to query and manage them.

1.2 The Registry Interface within the VO Architecture

This specification directly relates to other VO standards in the following ways:

VOResource (Plante and Benson et al., 2008)

VOResource sets the foundation for a formal definition of the data model for resource records via its schema definition.

IVOA Identifiers (Demleitner and Plante et al., 2015)

IVOA identifiers are something like the primary keys to the VO registry. Also, the notion of an authority as laid down in IVOA Identifiers plays an important role as publishing registries can be viewed as a realization of a set of authorities.
Figure 1: IVOA Architecture diagram with the Registry Interface specification (RI) and the related standards marked up.

2 The IVOA Harvesting Interface

The harvesting interface allows the retrieval of complete VOResource records from registries supporting harvesting. Publishing registries MUST support the IVOA harvesting interface, searchable registries SHOULD do so.

The IVOA harvesting interface is built on the standard Protocol for Metadata Harvesting developed by the Open Archives Initiative, OAI-PMH (Lagoze and de Sompel et al., 2002). In this section, after giving a brief introduction to OAI-PMH, we define additional constraints and requirements for OAI-PMH services to be interoperable with the VO environment.

In version 1.0 of this document, a variant of the OAI-PMH protocol was defined using SOAP in the exchange of messages. Version 1.1 no longer defines it (although of course there is no requirement to remove it from running services); since OAI-PMH over SOAP has never been in active use by the IVOA, we consider this still a minor specification change not warranting a new major version.
2.1 The OAI Protocol for Metadata Harvesting

While for details of OAI-PMH we refer to Lagoze and de Sompel et al. (2002), in the following we give a brief overview of OAI-PMH that should be sufficient to understand the protocol’s role within the Registry interface architecture.

The OAI-PMH v2.0 specification 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 six standard operations laid down in OAI-PMH are:

**Identify**
provides a description of the registry

**ListIdentifiers**
returns a list of identifiers for the resource records held by the registry, possibly restricted to records changed within a certain time span or to those belonging to a certain set.

**ListRecords**
returns complete resource records in the registry, possibly restricted to records changed within a certain time span or to those belonging to a certain set.

**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**
returns a list of set 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 (with the 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 existence of the ListMetadataFormats operation, a harvester can request the formats available for encoding returned resource descriptions.
• Harvesting by date. The ListIdentifiers and ListRecords operations both support from and until date arguments which restrict the response to records changed within the given, possibly half-open, interval.

• 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 sets.

• Marking records as deleted. Registries may mark records as deleted so that harvesters will be notified that a resource has become unavailable even if only performing incremental harvests.

• 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.

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 a dedicated search interface on a searchable registry (cf. sect. 5).

In addition to basic OAI-PMH compliance, this specification defines a set of OAI-PMH-compliant requirements and recommendations special to OAI-PMH’s use within the VO that are described in the remaining subsections.

2.2 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 (Plante and Benson et al., 2008).

The VOResource metadata format has the metadata prefix name ivo_vor, which can be used wherever Lagoze and de Sompel et al. (2002) allows a metadata prefix name. The format uses the VOResource core XML Schema with the namespace http://www.ivoa.net/xml/VOResource/v1.0 (recommended namespace prefix vr:) along with any legal extension of this schema to encode the resource descriptions within the OAI-PMH metadata tag from the OAI XML Schema (namespace http://www.openarchives.org/OAI/2.0, recommended namespace prefix oai:).

As VOResource and its extensions do not define global elements, the child element within oai:metadata needs to be separately defined. This specification does this by providing the ri:Resource element. It is defined in a schema with the target namespace http://www.ivoa.net/xml/RegistryInterface/v1.0, which is given in appendix A.

The ri:Resource element MUST include an xsi:type attribute that assigns the element’s type to vr:Resource or one of its legal extensions.
It is strongly recommended that all QName values of xsi:type attributes within the VOResource record use XML namespace prefixes as recommended in VOResource or the VOResource extensions. Minor version changes are not in general reflected in the recommended prefixes – e.g., both VODataService 1.0 and VODataService 1.1 use vs:. Registry operators who must deliver OAI-PMH documents containing resource records written to different versions of a registry extension are advised to override the prefix bindings on the element level if at all possible.

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.

Harvestable registries may support other metadata formats. Responses to the ListMetadataFormats operation must list all names for formats supported by the registry; even though they are mandatory, this list must include ivo_vor and oai_dc.

2.3 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 as given in its vr:identifier child element.

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 (apart from possible additions of vr:validationLevel elements), 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 becomes 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.
2.4 Required Records

This section describes the records that a harvestable VO 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 also included in the Identify operation response. When encoded using the ivo_vor format, the returned ri:Resource element must be of the type \textit{vg:Registry} from the VORegistry schema (see sect. 3.4). The record MUST include a \textit{vg:managedAuthority} for every authority identifier that originates at that registry.

Additions to the list of a registry’s managed authorities must follow the protocol outlined in sect. 3.1.

The harvestable registry must be able to return exactly one record in ivo_vor for each authority identifier listed as a \textit{vg:managedAuthority} in the \textit{vg:Registry} record that describes that registry. When encoded in the ivo_vor format, the type of these elements must be \textit{vg:Authority}.

2.5 The Identify Operation

The \textit{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 an \textit{oai:baseURL} element that must refer to the base URL to the harvesting interface endpoint. The \textit{Identify} response must include an \textit{oai:description} element containing a single \textit{ri:Resource} element with an \textit{xsi:type} attribute that sets the element’s type to \textit{vg:Registry}. The content of \textit{vg:Registry} type must be the registry description of the harvestable registry itself.

In its \textit{Identify} response, an OAI-PMH-compliant registry must declare its support for deleted records. This can be one of

\textbf{no} – the registry will never notify harvesters of records that have become unavailable. In an environment like the VO, where searchable registries frequently harvest publishing registries, this is severely discouraged, as without deleted records, harvesters need to perform full harvests every time or risk delivering stale records.

\textbf{transient} – the registry will notify harvesters of records that have become unavailable, but the deleted records will entirely vanish after some time. This specification adds to the OAI-PMH requirements that registries declaring \textbf{transient} support MUST keep their deleted records for at least six months (after which they may discard them).

\textbf{persistent} – the registry promises to indefinitely keep deleted records.
2.6 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 Lagoze and de Sompel et al. (2002), 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 specification defines one reserved set name with a special meaning; future versions of this specification may define additional set names. These reserved set names will all start with the characters *ivo_*; implementors should not define their own set names that begin with this string. While support for sets is optional in the OAI-PMH standard, a VO registry MUST support the set with the reserved name *ivo_managed* to be compliant with this specification.

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 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 while harvesting other registries to avoid getting duplicate records.

2.7 Time Granularity

Datestamps in the OAI-PMH 2.0 standard are encoded using ISO8601 and expressed in UTC, with the UTC designator “Z” appended to seconds-based granularity where supplied, i.e. *YYYY-MM-DDThh:mm:ssZ*. In general OAI-PMH registries, granularity at seconds scale is optional. Harvestable IVOA registries MUST report datestamps at the granularity of seconds and accept *from* and *until* arguments in the same format. This simplifies the incremental harvesting process in the multi-registry IVOA environment.

3 Registering Registries

Harvesting registries must be able to locate remote registry resources relevant to them, and both harvesting registries and clients need access to metadata for the registry service itself. We address both of these issues by providing a schema for describing registries themselves, and a repository for indexing them.

The resource specification for registries themselves is defined by an *ri:Resource* extension *vg:Registry*, which describes metadata of the registry itself and its support for interfaces described in this document or elsewhere. These resources are themselves stored as records in registries as described in 2.4. From each identifier, further IVOA identifiers for authority information, services, and other records belonging under that publishing umbrella may be created. A publishing registry is said to exclusively manage a naming authority on behalf of the owning publisher; this means that within the IVOA registry network,
only that specific registry may publish records having identifiers which begin
with that authority identifier.

The XML namespace URI of this schema is http://www.ivoa.net/xml/VORegistry/v1.0. It has been chosen to allow it to be resolved as a URL to the
XML Schema document, which is also given in appendix B. The recommended
prefix for this namespace is vg:

The schema has not been changed from the one used in version 1.0, although
the standard contents have somewhat changed. The rationale for keeping the
schema unchanged is that the presence of schema features no longer relevant
has no detrimental consequences for Registry operations, whereas changing the
schema could break already operational clients.

<ri:Resource status="active" xsi:type="vg:Authority"
  updated="2006-07-01T09:00:00" created="2006-07-01T09:00:00">
  <title>IVOA Naming Authority</title>
  <shortName>IVOA</shortName>
  <identifier>ivo://ivoa.net</identifier>
  <curation>
    <publisher ivo-id="ivo://ivoa.net/IVOA">International Virtual
    Observatory Alliance</publisher>
    <creator>
      <name>Raymond Plante</name>
      <logo>http://www.ivoa.net/icons/ivoa_logo_small.jpg</logo>
    </creator>
    <date>2006−07−01</date>
    <contact>
      <name>IVOA Resource Registry Working Group</name>
      <email>registry@ivoa.net</email>
    </contact>
  </curation>
  <content>
    <subject>virtual observatory</subject>
    <description>This registers the IVOA as the owner of the ivoa.net
    authority identifier.</description>
    <referenceURL>http://rofr.ivoa.net</referenceURL>
  </content>
  <managingOrg>International Virtual Observatory Alliance</managingOrg>
</ri:Resource>

Figure 2: A sample vg:Authority-typed resource record as it would be delivered
within oai:metadata. XML namespace declarations for the prefixes ri:, xsi:,
and vg: are assumed on enclosing elements.
3.1 The Authority Resource Extension and the Publishing Process

The \texttt{vg:Authority} type extends the core \texttt{vr:Resource} type to specifically describe the ownership of an authority identifier by a publishing organization.

The IVOA identifier of a \texttt{vg:Authority} record provided via the \texttt{vr:identifier} element must have an empty resource key component as defined in Plante and Linde et al. (2007).

The meaning of a \texttt{vg:Authority} record is that the organization referenced in the \texttt{vg:managingOrg} element has the sole right to create (in collaboration with a publishing registry) and register resource descriptions using the authority identifier given by the \texttt{vr:identifier} element.

Before a publisher can create resource descriptions using a new authority identifier, it must first register its claim to the authority identifier by creating a \texttt{vg:Authority} record. Before the publishing registry commits the record for export, it must first search a full registry to determine if a \texttt{vg:Authority} with this identifier already exists; if it does, the publication of the new \texttt{vg:Authority} record must fail.

When a registry creates a \texttt{vg:Authority} record, it is said that the registry manages the associated authority identifier (on behalf of the owning publisher) because only that registry may create records with identifiers beginning with that authority identifier. The registry must also document this ownership by adding a corresponding \texttt{vg:managedAuthority} element to the registry's own resource record.

The mechanism outlined here is not free of potential conflicts in the distributed environment of the VO Registry. The IVOA Registry Working group periodically monitors the registry-authority graph to ensure each authority in the Registry is claimed by exactly one registry.

3.2 Describing Registries with the Registry Resource Extension

The \texttt{vg:Registry} type extends the core \texttt{vr:Service} type to specifically describe registries in order to support discovering them and collecting their metadata; in addition, the extension type also defines the VO-specific metadata in the response to an OAI-PMH \textit{Identify} request.

As a subclass of \texttt{vr:Service}, the \texttt{vg:Registry} type uses \texttt{vr:capability} elements to describe its support for network interfaces to the services. The specific types defined here derive from an intermediate restriction on \texttt{vr:Capability} called \texttt{vg:RegCapRestriction} to force the value of the \texttt{standardID} attribute to be \texttt{ivo://ivoa.net/std/Registry}. In particular, OAI-PMH endpoints as specified here are identified by \texttt{ivo://ivoa.net/std/Registry}. Client should discover registries by looking for records with capabilities declaring this \texttt{standardID}.  

If the \textit{vg:full} element in an \textit{vg:Registry} instance is set to \texttt{true}, it indicates the registry’s intent to accept all valid resource records it harvests from other registries in accordance with the OAI-PMH specification. This will typically be searchable registries implementing some Registry search interface, but there are also use cases for full registries only implementing OAI-PMH (and thus only providing an \textit{vg:Harvest} capability).

The \textit{vg:managedAuthority} is used by publishing registries to claim an authority identifier (see also sect. 2.4). Note that for each managed authority claimed, the registry MUST provide a \textit{vg:Authority}-typed resource record for that authority identifier within its \texttt{ivo_managed} set.

As of version 1.1 of this specification, VO registry records must provide the three mandatory VOSI capabilities: availability, a listing of service capabilities, and a listing of tables if relevant, i.e. if a RegTAP or other tabular interface is available (Grid and Web Services Working Group, 2011).

### 3.3 The Search Capability

Version 1.0 of this standard defined a search interface, and such interfaces are described by capabilities of the type \textit{vg:Search}. Since in this version, search interfaces are specified by external standards, such external standards may define differing ways of discovering them\(^1\). The search capability nevertheless is not removed from the schema for backward compatibility, and is available in appendix D.

### 3.4 The Harvesting Capability

A registry declares itself to be a harvestable registry by including a \textit{vr:capability} element with an \texttt{xsi:type} attribute set to \textit{vg:Harvest}. An example capability for this type is provided in the appendix C.

A \textit{vr:capability} element of type \textit{vg:Harvest} MUST include at least one \textit{vr:interface} element with an \texttt{xsi:type} attribute set to \textit{vg:OAIHTTP} and the \texttt{role} attribute set to \texttt{std}. If the \textit{vr:capability} element is used to simultaneously describe support for other versions of this Registry Interface standard, then the \textit{vr:interface} element describing support for this version must include the version attribute set to \texttt{1.0}. The \textit{vr:accessURL} element must be set to the base URL for the OAI-PMH interface.

The \textit{vg:OAISOAP} extension of \textit{vr:WebService} was defined in version 1.0 of this specification and is no longer part of VO Registry interfaces since it was never used.

### 4 Registry Discovery

\(^1\)For instance, RegTAP (Demleitner and Harrison et al., 2013) uses the \texttt{tre:dataModel} element from TAPRegExt as its primary discovery mechanism in its version 1.0.
4.1 The Registry of Registries

To facilitate discovery and automated harvesting of VO publishing registries, a master list of IVOA registries exists as part of the IVOA web infrastructure, hosted at http://rofr.ivoa.net. It is referred to as the Registry of Registries, or RoR (pronounced “rover”). As the RoR is itself a registry, it provides an OAI-PMH interface conforming to this document. The OAI-PMH interface is always available at http://rofr.ivoa.net/oai. The RoR includes resource records describing each currently active registry of IVOA resources, its status as a full or local registry, authorities associated with it, and its programmatic interfaces. Each record is of type `vg:Registry` as defined in section 3.2.

Once a registry provider has deployed a new publishing registry, they must enroll it in the RoR for their records to be seen by the full searchable registries, and therefore registry search clients accessing the whole IVOA registry ecosystem. The RoR provides a dedicated web-based interface for this purpose accessible from http://rofr.ivoa.net. The RoR includes a validator package, which thoroughly checks the new registry, including schema validation for the OAI interface itself and all listed resources. The registration process will only accept registries that validate successfully. Local updates within a publishing registry post-inclusion in the RoR are not necessarily automatically validated by the RoR software later: the validator tool can, and indeed should, be used independently of the initial admission process by the registry providers to periodically make sure their registries are still compliant with the relevant IVOA standards.

The Registry of Registries also contains resources describing the most recent versions of IVOA standards for resources and resource extensions themselves; these are of type `vstd:Standard`. It is not guaranteed that every standard will be represented in RoR, but for the ones that are listed, the RoR version of their document is the canonical version.

4.2 Harvesting the Registry of Registries

Given the Registry of Registries contains records for all other currently active and validated IVOA registries, a client wishing to harvest the contents of all registries should begin at the RoR. Full searchable registries wishing to include records from the other IVOA registries count among these potential clients. To harvest the entire contents of IVOA registries, it is recommended to first harvest the Registry of Registries via its OAI-PMH interface.

This first step is done by making a call to the RoR’s OAI-PMH interface with the `ListRecords` operation, with the `set` argument set to `ivo_publishers`. This will return the registry records (i.e. resources with `xsi:type=’vg:Registry’`) for the registries that successfully registered themselves as described in 4.1.

The next step in harvesting the entire distributed IVOA registry contents is to iterate over the `accessURL` of each `vg:Registry` record’s `vr:capability` of type `vg:Harvest`, and use the URL for each of those OAI-PMH interfaces to harvest the individual registries. In iterating over the OAI interface of each registry itself, to avoid harvesting duplicate records from the full searchable
registries, it is recommended to add the set parameter to that OAI query as well: records locally published by a full registry comprise that registry’s supported set ivo_managed.

The very first time the harvester executes the ListRecords operation on the RoR or any listed registry, the from argument should be not used so that all known publishing registries are returned, as well as all known resources within each discovered registry. If the harvesting client wishes to use the OAI interface for incremental updates, it can cache at least a mapping of the registry identifiers to their respective harvesting endpoints along with a timestamp for when this operation was last successfully carried out on each. Then, at the start of subsequent harvesting updates, the harvester can provide the cached date using the from argument to receive only new and updated records, and update the cached timestamp upon success.

Experience has shown that when relying on incremental harvests exclusively, minor problems eventually accumulate to severe inconsistencies even when registries declare support for deleted records. It is therefore recommended that harvesting clients occasionally (e.g., semianually) perform full updates to an empty local copy without using the from parameter, even for registries that announce deletion of records. To further provide some robustness against small operational issues in the publishing process, it is also recommended to leave an overlap in incremental harvesting requests, e.g. to request resources going back to the beginning of the day of last incremental harvest.

For example, to get a listing of registries in the IVOA ecosystem, one would first query

\[http://rofr.ivoa.net/oai?verb=ListRecords\]
\[&metadataPrefix=ivo_vor\]
\[&set=ivo_publishers.\]

Then, for each returned resource, the accessURL under a Capability with xsi:type=vg:Harvest, that URL could be called as such:

\[http://accessURLValue?verb=ListRecords&metadataPrefix=ivo_vor\]

or

\[http://accessURLValue?verb=ListRecords\]
\[&metadataPrefix=ivo_vor\]
\[&from=YYYY-MM-DDTHH:MM:SSZ\]

for return visits, with the ‘from’ date representing the last successful query to that accessURL.

5 Searching Registries

Experience with version 1 of this specification suggests that it is preferable to not couple the relatively stable standards for harvesting and general registry
maintenance with client interfaces to the registry, which were found to be in much more need of experimentation. For a discussion of the history of client interfaces in the VO, see Demleitner and Harrison et al. (2015).

5.1 RI Search

A SOAP-based search capability, \texttt{vg:RISearch} defined in Registry Interfaces 1.0, exists but is no longer encouraged or required for searchable registries as technologies have moved forward. However, it is still a valid capability defined in the registry resource schema so that registry operators may continue to provide valid RI registries without having to support different versions of the VORegistry schema. The base \texttt{vg:RISearch} extension may also be useful for the description of future registry search interfaces. RISearch is described in appendix D.

5.2 Registry Table Access Protocol Services

One second-generation standard search interface to the VO Registry that has progressed to become an IVOA recommendation is RegTAP (Demleitner and Harrison et al., 2013), an interface based on a relational representation of key fields in resource descriptions and on the IVOA Table Access Protocol (Dowler and Rixon et al., 2010). RegTAP services have been made available from several registry providers listed in the Registry of Registries.

RegTAP-based registries should be located by clients as described in the RegTAP standard (which in version 1.0 happens through locating TAP services with a certain data model identifier like \texttt{ivo://ivoa.net/std/RegTAP#1.0}). To aid smart clients of the full RofR which generate lists for initial discovery, RegTAP registries must also be registered as separate resources with the appropriate tableset. These must include either a full TAP service capability according to TAPRegExt (Demleitner and Dowler et al., 2012) or an auxiliary capability referencing a TAP service as per Demleitner and Taylor (2016). An example for the latter option, preferable if the TAP service in question contains additional tables, is given in appendix C.

5.3 Announcing Local vs Full Searchable Registries

While a publishing registry may provide search capabilities for its own hosted records, this is considered a locally searchable registry, and not a full searchable one, as distinguished in the RofR listing. For a registry to be considered full searchable, it must harvest resources from the other publishing registries listed in the RofR, and implement an IVOA standard programmatic interface beyond the interface for OAI harvesting, with some method for filtering resource queries. This can be announced simply in the registry’s own self-describing resource record with a \texttt{full} tag set to true, without having to proscribe any one interface as the defining search feature.
6 Looking Forward

While the OAI-PMH harvesting interface as adopted from outside the IVOA community is stable and replacing it would require a major revision of this document, we expect that new search interfaces for registries will be continually developed, leveraging new technologies and best practices as they emerge. These search interfaces can be added without sacrificing interoperability with the IVOA registry ecosystem. Whether these emerging search technologies become formally endorsed by the IVOA as notes or new standards documents, so long as a registry supports the basic harvest interface and hosts valid \texttt{ri:Resource} documents including registry and authority records, it should be considered covered by the practices described herein and a welcome addition to the Registry of Registries listing, with all of its records also accessible through the full registries.

A The RegistryInterface Schema

The following schema defines a global element, allowing the inclusion of VOResource records into \texttt{oai:metadata} elements in OAI-PMH responses for the \texttt{ivo_vor} metadata prefix. See sect. 2.2 for details.

The schema is unchanged from version 1.0 of this specification and therefore does not change its version.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.ivoa.net/xml/RegistryInterface/v1.0"
  xmlns:ri="http://www.ivoa.net/xml/RegistryInterface/v1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
  elementFormDefault="qualified"
  version="1.0">
  <xs:import namespace="http://www.ivoa.net/xml/VOResource/v1.0"
    schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0"/>

  <xs:element name="VOResources">
    <xs:annotation>
      <xs:documentation>
        a container for one or more resource descriptions or identifier references to resources.
      </xs:documentation>
      <xs:documentation>
        This is used to transmit multiple resource descriptions resulting from a query.
      </xs:documentation>
    </xs:annotation>

    <xs:complexType>
      <xs:sequence>
        <xs:choice>
          <xs:element ref="ri:Resource"
            minOccurs="0" maxOccurs="unbounded"/>
          <xs:element name="identifier" type="vr:IdentifierURI"/>
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
B  The VORegistry Schema

The following schema defines VOResource types for describing registries in the Registry. It is unchanged from version 1.0 of this specification and therefore does not change its version.

Note that standards defining search interfaces may specify alternative or complementary methods of registering the services defined by them, and that auxiliary capabilities for these search capabilities may be listed within the registry record.

<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.ivoa.net/xml/VORegistry/v1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
  xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0"
  xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.1"
  xmlns:vm="http://www.ivoa.net/xml/VOMetadata/v0.1"
  elementFormDefault="unqualified" attributeFormDefault="unqualified"
  version="1.0.1">
  <xs:annotation>
    <xs:appinfo>
      <vm:schemaName>VORegistry</vm:schemaName>
      <vm:schemaPrefix>xs</vm:schemaPrefix>
      <vm:targetPrefix>vg</vm:targetPrefix>
    </xs:appinfo>
    <xs:documentation>
      An extension to the core resource metadata (VOResource) for describing registries and authority IDs.
    </xs:documentation>
  </xs:annotation>
  <xs:import namespace="http://www.ivoa.net/xml/VOResource/v1.0"
    schemaLocation="http://www.ivoa.net/xml/VOResource/v1.0"/>
a service that provides access to descriptions of resources.

A registry is considered a publishing registry if it contains a capability element with xsi:type="vg:Harvest". It is considered a searchable registry if it contains a capability element with xsi:type="vg:Search".

If true, this registry attempts to collect all resource records known to the IVOA.

A registry typically collects everything by harvesting from all registries listed in the IVOA Registry of Registries.

an authority identifier managed by the registry.

Typically, this means the AuthorityIDs that originated (i.e. were first published by) this registry. Currently, only one registry can lay claim to an AuthorityID via this element at a time.

For registry interfaces with a user-visible table structure, tableset allows its declaration.
In case protocols implemented in different capabilities have conflicting requirements on tableset, the two capabilities should be considered belonging to separate resources.

```xml
<xs:complexType name="Harvest">
  <xs:annotation>
    <xs:documentation>
      The capabilities of the Registry Harvest implementation.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="vr:Capability">
      <xs:sequence>
        <xs:element name="maxRecords" type="xs:int">
          <xs:annotation>
            <xs:documentation>
              The largest number of records that the registry search method will return. A value greater than one implies that an OAI continuation token will be provided when the limit is reached. A value of zero or less indicates that there is no explicit limit and thus, continuation tokens are not supported.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```xml
<xs:complexType name="Search">
  <xs:annotation>
    <xs:documentation>
      The capabilities of the Registry Search implementation.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="vr:Capability">
      <xs:sequence>
        <xs:element name="maxRecords" type="xs:int">
          <xs:annotation>
            <xs:documentation>
              The largest number of records that the registry search
              </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
method will return. A value of zero or less indicates that there is no explicit limit.
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="extensionSearchSupport"
    type="vg:ExtensionSearchSupport">
<xs:annotation>
<xs:documentation>
(deprecated)
</xs:documentation>
<xs:documentation>
This was used in Registry Interfaces 1.0 to indicate what VOResource extensions a search interface supported. Modern search interfaces will indicate that through version, their tableset, or similar.
</xs:documentation>
</xs:annotation>
</xs:element>

<xs:element name="optionalProtocol"
    type="vg:OptionalProtocol"
    minOccurs="0" maxOccurs="unbounded">
<xs:annotation>
<xs:documentation>
(deprecated)
</xs:documentation>
<xs:documentation>
This was used in Registry Interfaces 1.0 to indicate search protocol extensions. In 1.1, use multiple capabilities with the appropriate standardIDs to declare special search capabilities.
</xs:documentation>
</xs:annotation>
</xs:element>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>

<xs:simpleType name="ExtensionSearchSupport">
<xs:restriction base="xs:NMTOKEN">
<xs:enumeration value="core">
<xs:annotation>
<xs:documentation>
Only searches against the core VOResource metadata are supported.
</xs:documentation>
</xs:annotation>
</xs:enumeration>
<xs:enumeration value="partial">
<xs:annotation>
<xs:documentation>
Searches against some VOResource extension metadata are supported but not necessarily all that exist in the registry.
</xs:documentation>
</xs:annotation>
</xs:enumeration>
</xs:restriction>
</xs:simpleType>
Searches against all VOResource extension metadata contained in the registry are supported.

The XQuery (http://www.w3.org/TR/xquery/) protocol as defined in the VO Registry Interface standard.

The accessURL child element is the base URL for the OAI service as defined in section 3.1.1 of the OAI PMH standard.
C Example Capabilities

The following XML fragment shows the three capability elements discussed in this document: The OAI-PMH-based publishing registry, the legacy RI 1.1 searchable registry, and an auxiliary TAP capability as used for RegTAP.

```xml
<ri:Resource
xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:ri="http://www.ivoa.net/xml/RegistryInterface/v1.0">
<!-- Standard VOResource metadata omitted for brevity -->
<!-- The capability for an OAI-PMH endpoint (publishing registry) -->
<capability xsi:type="vg:Harvest" standardID="ivo://ivoa.net/std/Registry">
 <interface xsi:type="vg:OAIHTTP" version="1.0" role="std">
</interface>
</capability>
```

24
<accessURL use="base">http://registry.example.org/oai</accessURL>
</interface>
<maxRecords>100</maxRecords>
</capability>

<!-- A legacy, RI1.0 searchable registry endpoint, with an extra interface for web browsers. -->
<capability xsi:type="vg:Search" standardID="ivo://ivoa.net/std/Registry">
  <interface xsi:type="vr:WebBrowser" version="1.0" role="gui">
    <accessURL use="full">http://registry.euro-vo.org</accessURL>
  </interface>
  <interface xsi:type="vr:WebService" version="1.0" role="std">
    <accessURL use="full">
      http://registry.example.org/services/RegistrySearch
    </accessURL>
  </interface>
  <maxRecords>100</maxRecords>
  <extensionSearchSupport>core</extensionSearchSupport>
</capability>

<!-- A reference to RegTAP-enabled TAP service as an auxiliary capability -->
<capability standardID="ivo://ivoa.net/std/TAP#aux">
  <interface xsi:type="vs:ParamHTTP" role="std">
    <accessURL use="base">http://registry.example.org/tap</accessURL>
  </interface>
</capability>

<!-- A RegTAP-capable searchable registry should have a tableset with all its tables in the rr schema here -->
</ri:Resource>

D The RISearch Schema

The following schema defines the SOAP-based RISearch interface, which is discouraged as of version 1.1 but still available. It is unchanged from version 1.0 of this specification and therefore does not change its version.

vg:Search Type Schema Documentation

The capabilities of the Registry Search implementation.

vg:Search Type Schema Definition

<xs:complexType name="Search">
  <xs:complexContent>
    <xs:extension base="vr:Capability">
      <xs:sequence>
        <xs:element name="maxRecords" type="xs:int" />
        <xs:element name="extensionSearchSupport" type="vg:ExtensionSearchSupport" />
        <xs:element name="optionalProtocol" type="vg:OptionalProtocol" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
### vg:Search Extension Metadata Elements

**Element** `maxRecords`  
*Type* `xs:int`  
*Meaning* The largest number of records that the registry search method will return. A value of zero or less indicates that there is no explicit limit.  
*Occurrence* required

**Element** `extensionSearchSupport`  
*Type* `string`  
*Meaning* (deprecated)  
*Occurrence* required

**Allowed Values**  
- **core** Only searches against the core VOResource metadata are supported.  
- **partial** Searches against some VOResource extension metadata are supported but not necessarily all that exist in the registry.  
- **full** Searches against all VOResource extension metadata contained in the registry are supported.  

*Comment* This was used in Registry Interfaces 1.0 to indicate what VOResource extensions a search interface supported. Modern search interfaces will indicate that through version, their tableset, or similar.

**Element** `optionalProtocol`  
*Type* `string`  
*Meaning* (deprecated)  
*Occurrence* optional; multiple occurrences allowed

**Allowed Values**  
- **XQuery** the XQuery (http://www.w3.org/TR/xquery/) protocol as defined in the VO Registry Interface standard.  

*Comment* This was used in Registry Interfaces 1.0 to indicate search protocol extensions. In 1.1, use multiple capabilities with the appropriate standardIDs to declare special search capabilities.

### E Changes from Previous Versions

For pre-REC-1.0 changes, see Benson and Plante et al. (2009).
E.1 Changes from first 1.1 WD

- Text clarifications for harvesting the entire RofR, and exhortation to harvest from scratch occasionally as OAI announcement of record deletions are not mandatory.
- Simplified announcement of full searchable registry in the RofR and removed operational instructions which may change.

E.2 Changes from Version 1.0

- Corrected reference to OAI-PMH spec in registry interface description to v2.0.
- Added requirement for OAI-PMH interface to support seconds granularity, optional in the OAI-PMH 2.0 standard itself.
- Removed requirement for VOResource version number changes to force an update of this document.
- Removed the implementation-dependent requirement for searchable registries in section 2, specifically the SOAP-based services based on “ADQL 1.0” and XQuery.
- Dropped the requirement on registries to not deliver any records that are OAI-PMH deleted when no temporal constraint is given.
- Added a requirement to provide VOSI endpoints.
- Added support for auxiliary Registry TAP Service search interfaces.
- Clarified that the requirement to keep deleted records for six months only applies to the transient case; also discouraging registries with no support of deleted records.
- Added recommended process for discovery of registries and their resources using the Registry of Registries, based on the Registry of Registries IVOA note.
- Added conclusion describing implications of future search and publishing interface changes in the Registry environment.
- Many editorial changes across the text, mostly as a consequence of externalizing search interfaces.
References


http://ads.ari.uni-heidelberg.de/abs/2004ASPC..314..585P

http://www.ivoa.net/documents/REC/Identifiers/Identifiers-20070302.html