Astronomy and Computing 7–8 (2014) 101–107

Contents lists available at ScienceDirect

Astronomy and Computing

journal homepage: www.elsevier.com/locate/ascom

Full length article

The virtual observatory registry

M. Demleitner [a,*], G. Greene [b], P. Le Sidaner [c], R.L. Plante [d]

[a] Universität Heidelberg, Astronomisches Rechen-Institut, Mönchhofstraße 12-14, 69120 Heidelberg, Germany
[b] Space Telescope Science Institute, 3700 San Martin Dr, Baltimore, MD 21218, USA
[c] VOParis/Observatoire de Paris, 61 Av de l'Observatoire, 74014 Paris, France
[d] National Center for Supercomputing Applications, University of Illinois, 1205 W. Clark St., Urbana, IL 61821, USA

ARTICLE INFO

Article history:
Received 30 April 2014
Received in revised form
7 July 2014
Accepted 7 July 2014
Available online 19 July 2014

Keywords:
Virtual observatory
Registry
Standards

ABSTRACT

In the Virtual Observatory (VO), the Registry provides the mechanism with which users and applications discover and select resources – typically, data and services – that are relevant for a particular scientific problem. Even though the VO adopted technologies in particular from the bibliographic community where available, building the Registry system involved a major standardisation effort, involving about a dozen interdependent standard texts. This paper discusses the server-side aspects of the standards and their application, as regards the functional components (registries), the resource records in both format and content, the exchange of resource records between registries (harvesting), as well as the creation and management of the identifiers used in the system based on the notion of authorities. Registry record authors, registry operators or even advanced users thus receive a big picture serving as a guideline through the body of relevant standard texts. To complete this picture, we also mention common usage patterns and open issues as appropriate.

1. Introduction

The Virtual Observatory (VO) is a distributed system—by design, there is no central node either running services, delivering data, or even just a single link list-style directory. In order to still maintain the appearance of a single, integrated information system, users and clients must have a means of discovering metadata of VO-compliant resources (in the sense discussed in Section 3). This means is provided by the VO Registry.[1]

Following the VO philosophy, the VO Registry is not a single, central system but rather a network of several types of services, some of which host and publish metadata collections, while others provide capabilities for querying such collections. All follow standard protocols for exchanging information between them and between them and client software.

The VO Registry is governed by a fairly large set of standards; one of the goals of this paper is to review this body of text and discuss how each standard fits into the architecture. Anticipating some terms that will be explained later, let us collect and arrange the relevant standards already in the introduction.[2] Where the standards have short names in common use in the VO community, we introduce these here and refer to the standards by their mnemonic names in the following.

- *IVOA Identifiers* (Plante et al., 2007) lays out how resources and resource records in the VO are referenced.
- *Resource Metadata for the Virtual Observatory* (*RM* for short; Hanisch, 2007) specifies what entities need descriptions in the VO and what pieces of metadata these should contain to satisfy the VO's use cases.
- *VOResource* (Plante et al., 2008) lays out the basics of encoding resource metadata information as specified in *RM* in XML and defines the basic types. When we talk about *VOResource* in the following, we usually mean not only (Plante et al., 2008) but also the registry extensions introduced next.
- Several *Registry extensions* apply the building blocks from *VOResource* to more specialised types of services or interfaces. All of these combine a definition of the metadata as well as its XML serialisation.

* Corresponding author. Tel.: +49 6221541837.
E-mail address: msdemlei@ari.uni-heidelberg.de (M. Demleitner).

[1] Written in upper case in the following, the term "Registry" refers to the entire system, as opposed to the lower-case "registry", which denotes a concrete service.

[2] For an even bigger picture of the VO and its components, see Arviset and Gaudet (2010).

http://dx.doi.org/10.1016/j.ascom.2014.07.001
2213-1337/

  - *VODataService* (Plante et al., 2010) defines extra metadata to describe data collections and services exposing them; in particular, this concerns table and column metadata as well as metadata on service parameters.
  - *SimpleDALRegExt* (Plante et al., 2012) defines what extra metadata applies to services implementing several "simple" protocols of the VO's Data Access Layer (DAL).
  - *TAPRegExt* (Demleitner et al., 2012) defines what extra metadata applies to services implementing the Table Access Protocol TAP.
  - *StandardsRegExt* (Harrison et al., 2012) contains resource types for standard texts and thus defines how standards can be referenced, e.g., when declaring protocol support.
- *Registry Interfaces* (Benson et al., 2009) specifies how registries exchange the XML records defined in *VOResource* and extensions. It also contains a Registry extension for the services implementing Registry services themselves. Furthermore, its current version also defines two APIs for registry clients; in a forthcoming version, these APIs will be dropped.
- *Registry Interfaces* re-uses the non-VO *OAI-PMH* (Various, 2002) standard. This Protocol for Metadata Harvesting defined by the Open Archives Initiative governs the interactions of the registries among themselves. Its use by the VO is subject to several idiosyncrasies laid out in *Registry Interfaces*.
- *RegTAP* (Demleitner et al., 2014) defines how registry users can query the Registry's data content using IVOA's Table Access Protocol. An alternative, parameter-based API is currently being designed. We defer the discussion of the client APIs to a forthcoming article.

In the remainder of this paper, we will first delineate the Registry's role in the VO and outline its scope (Section 2), before establishing some basic notions on the relation between resources and their descriptions as the VO treats it in Section 3. Having thus introduced the concept of a resource record, in Section 4 we proceed to discuss how registries maintain collections of them. Section 5 explains the process of transmission and dissemination of the records and the separation of responsibilities in this process, as well as a common implementation error that has long plagued the Registry. The VO's way to generate globally unique identifiers as required by the harvesting protocol is then considered in Section 6.

With the basic architecture described, we proceed to discuss the current Registry content in Section 7, in particular as regards what resource records are contained. This provides some insight into the data model underlying the Registry. For the most relevant case where the resources described are services, special care must be taken in the description of "capabilities", i.e., facilities that operate on a client's behalf. We give an overview of these capabilities in Section 8. Finally, we briefly touch the issue of the validation of services and their descriptions in Section 9.

2. Scope

The Registry's role in the VO primarily is resource discovery. Hence, it must collect data sufficient to answer requests at least of the following types (or their combination):

- Resources of type X (as in: image service, database service, etc.),
- Resources on topic X (defined through keywords or via a full text search in the resource descriptions),
- Resources with physics X (defined through waveband, observables, queriable phenomena, etc.),
- Resources by author(s) X,
- Resources suitable for use X,
- Resources with spatial or temporal coverage X.

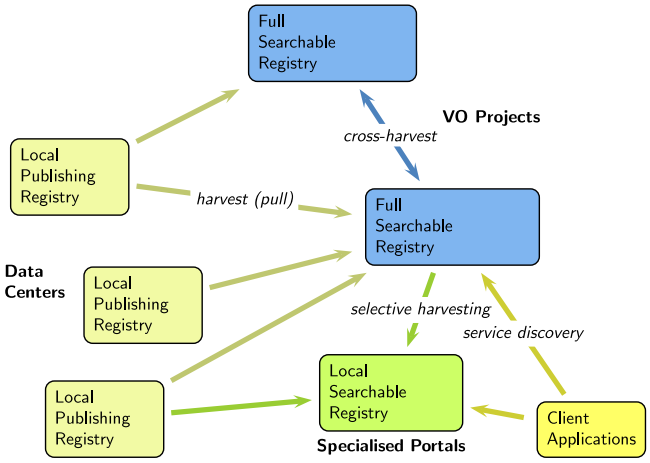

Fig. 1. A sketch of the registry system in the Virtual Observatory as laid out by Plante and Greene (2008): searchable registries harvest from publishing registries operated by the data providers. Users and client applications can then discover VO resources through queries to a searchable registry, either a full searchable registry that contains everything known to the VO, or a specialised one focused on a particular subset.

Once a resource record has been located by any of these constraints, it provides sufficient information at least to let users

- Assess suitability of the resource for purpose X,
- Access the resource,
- Identify who to credit for results obtained using the resource,
- Contact technical support for the resource.

The VO Registry is also used to monitor the health and functionality of the VO. The registries themselves are routinely validated and curated to ensure consistency with IVOA standards, which uncovers errors in the metadata supplied by the service operators. Even more importantly, services within the Registry are validated to comply to the standards they claim to implement, and registry records, where necessary, contain test input parameters suitable for exercising a service.

The Registry as such is *not* a mechanism of data preservation, and it does not provide persistent identifiers. The identifiers within the VO Registry, the IVORNs, are simple URIs with a scheme of `ivo`, an authority part as discussed in Section 6, and a local part governed by some reasonable restrictions on which characters are allowed to occur.

They can be resolved to resource records and, if applicable, access URLs by searchable registry and thus, not unlike DOIs (ISO Technical Committee 46, 2012), introduce a level of indirection between a service identifier and its access URLs. However, the indirection in the Registry mainly is a side effect of the requirement to provide rich, structured metadata for the services.

Unlike with DOIs, an operator is free at any time to discard identifiers, and the current VO infrastructure would stop resolving it on a short timescale. The conceptual reason why IVORNs as such are not suitable as persistent identifiers is that, as laid out in Section 3, they are in the first place identifiers of the resource records. Although the VO Registry could be exploited as a basis for (external) data preservation services and persistent identifiers for resources–Accomazzi (2011) reports on one such effort –, it does not in itself provide such facilities.

3. Resources and resource records

The Virtual Observatory can be seen as a collection of *resources*. Hanisch (2007) defines a VO resource as 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". He goes on to name sky

coverages, instrumental setups, organisations, or data collections as examples. In practice, over 95% of resources in the current VO are data services.

From the outset, it was clear that a common way of describing these resources would be required as a very basic building block for interoperability. For instance, VO enabled client programs need to be able to find out what protocols a service supports and at what "endpoints" – typically, HTTP URLs – these are available. Scientists should have reliable and standardised ways to work out who to reference, who to consult in case of malfunctions, and so on. Of course, having a standardised structure for content metadata (like keywords, a title, description) helps writing more focused data discovery queries as well.

Fortunately, the VO did not have to develop the technology to support such descriptions itself, as library sciences have worked on very comparable problems for centuries already. The VO's registry architecture in particular re-uses the Open Archives Initiative's protocol for metadata harvesting (OAI-PMH; Various, 2002) for a conceptual framework and the metadata exchange protocol, and Dublin Core (Kunze and Baker, 2007) for a basis on which to build the metadata model.

Central to *OAI-PMH* is the notion of a *unique identifier*, which "unambiguously identifies an item within" the set of resource records. Other than that these should be URIs (Berners-Lee et al., 1998), *OAI-PMH* does not state details on how they should be formed. For the VO, *IVOA Identifiers* prescribes the use of IVORNs as introduced in Section 2.

A somewhat subtle but nevertheless important distinction made in *OAI-PMH* is between a resource and a *resource record* containing its description. To see that this distinction has actual consequences, say the data collection X contains spectra obtained using the spectrograph S, and the resource record R describes X. Now, during the lifetime of the instrument, S will add new data to X on every clear night, which means the resource changes. Nevertheless, in the current VO R will not generally change (though it is conceivable that it will be updated now and then, e.g., as the description might contain rough estimates on the number of datasets contained in X).

For the converse scenario of a changing resource record with a constant resource, suppose S is now decommissioned, while the standard defining the content of the resource record is updated to include the spatial coverage of the data collection. Now, R needs an update without X changing.

As stated above *OAI-PMH* defines that its unique identifiers – and hence the VO's IVORNs – always reference resource records. As to how the resources themselves should be referenced, *OAI-PMH* declares that the "nature of a resource identifier is outside the scope" (Various, 2002). This reservation is motivated by the library use case, where a single book might be described by different libraries and hence have multiple resource records.

In the VO, it was expected that such complications would not arise as the resource records would almost always come from the resource publishers themselves, and no need for multiple resource records for a single resource was foreseen. It was therefore decided that the IVORN of a resource record should also identify the resource itself, which simplifies identifier generation and management significantly.

This also explains why, in *OAI-PMH* messages with the metadata prefix `ivo_vor` (see Section 4 for details), the IVORN is repeated in both the header and the metadata of a resource record. Awareness of the distinction is relevant to registry users to understand the meaning of the creation or update times in the resource record (which refer to the record itself) and the dates and times given in the curation/date child of the resource record, which pertain to the resource.

4. Registries

Having a set of resource records alone is not enough to build a useful system, even if they already are in a standard format. There must also be ways in which users can locate records of the resources relevant to them within this set. Therefore, systems are required enabling service operators to feed their resource records into the set. Also, users must have a way to execute queries against the set. Both requirements are covered by *registries* within the VO.

Injection of resource records is performed by *publishing registries*. These typically are run by service operators and deliver resource records of a specific operator's set of services. In addition, for service operators who choose not to run publishing registries of their own, both the registry at STScI[3] and the registry at ESAC[4] run publishing registries accepting third-party resource records which are typically created using web interfaces provided by the institutions.

Conceivably, a user looking for a resource matching some constraints could now query each publisher's publishing registry in turn to obtain a list of all matching VO resources. This architecture obviously will not scale well with the number of publishers. It also introduces many points of failure into the system, as all publishers would have to keep their registries highly available to avoid a severe degradation of the whole system.

To avoid these issues, retrieving resource records from the publishing registries, joining the sets of resource records thus obtained, and offering a means of querying this joined set to VO users are the tasks of specialised agents, the *searchable registries*. The process of retrieval of resource records from publishing registries by a searchable registry is known as *harvesting*. To allow this harvesting, publishing and searchable registries must agree on a common protocol. As mentioned in Section 3, the adoption of *OAI-PMH* already defined such a protocol for the VO.

A secondary distinction between searchable registries is between *full registries* (the term "searchable" is usually implied in this case) which strive to harvest all publishing registries in the VO and *local searchable registries* which only carry a selection of records. An example for the second kind that is currently in discussion is an "educational" registry that contains a manually curated subset of services delivering data suitable for classroom use (i.e., data of moderate size, with easily understood data types, etc.).

The actual application of *OAI-PMH* within the VO is described in *Registry Interfaces*, which in particular defines that the VO's own resource record format is selected in *OAI-PMH* using a metadata prefix of `ivo_vor`. Requesting this will make a VO-compliant registry embed *VOResource* records as discussed in Section 7 in the *OAI-PMH* record's metadata child. VO registries are also required to emit the much simpler Dublin core metadata records on request and are thus interoperable with bibliographic services outside of the VO; within the VO the much richer *VOResource* metadata is used exclusively.

One additional building block needs to be mentioned, the Registry of Registries or RofR for short (Plante, 2007). This is a special publishing registry from which searchable registries can harvest the set of available publishing registries to initialise or update their internal list of registries to work on. As such, it is a single point of failure, as there is only one such service globally. On the other hand, no client code directly accesses the RofR, which means that an outage of the RofR does not impair the user-visible functionality of the VO. The main impact would be that no new publishing registries could be added to the VO's registry system,

[3] http://vao.stsci.edu/directory/.
[4] http://registry.euro-vo.org/.

and existing registries' endpoints would have to be discovered from searchable registries (which in user tools they usually are anyway).

In the current VO, the RofR also doubles as the publishing registry for standards and other resources managed by IVOA, and it operates a service for validating the content of publishing registries (cf. Section 9).

5. Harvesting

The VO registry system is de-centralised in both directions: a given publishing registry does not know which searchable registries will eventually carry its records. An implication of this is that it cannot notify the searchable registries when a resource record changes. This, in turn, implies that the searchable registries will have to poll the publishing registries it harvests. This is not entirely trivial, as the largest publishing registry in the VO currently emits more than 100 MB of resource records, and due to paging and other delays the transfer takes about 10 min.

On the other hand, to keep up to date, searchable registries should poll the publishing registries with a fairly high frequency. Most active searchable registries today poll once or twice a day. To nevertheless keep network and CPU load low, *OAI-PMH* supports *incremental harvesting*. This allows searchable registries to query publishing registries for records updated since some point in time.

A common harvesting strategy is that searchable registries persist the date and time of the last harvest and, on re-harvesting, query the publishing registry for records updated since then. Together with a very natural-seeming (but incorrect) implementation on the part of the publishing registry, this can lead to a loss of records with incremental harvesting.

To see how this happens, consider a publishing registry P that, as is usual, keeps the updated dates of its resources in a database table to facilitate quick responses to *OAI-PMH* queries. The race condition now can be exposed as follows:

1. A new resource record R is created at t_1 and its updated attribute accordingly is set to t_1 in the record itself. For one reason or another, the program that ingests the updated dates for the record into the database table does not run immediately.
2. At $t_2 > t_1$, a searchable registry S harvests P and memorises t_2 as the date of the last harvest. As the database table does not contain R yet, R is not harvested.
3. At $t_3 > t_2$, the program ingesting R into the database table is finally run, but the timestamp is taken from the resource record, i.e., it is t_1.
4. S comes back for an incremental harvest at $t_4 > t_3$ and asks for records updated after t_2. As $t_1 < t_2$, R is not in the set of resources delivered.

Hence, the record will be missed by S, which then will not contain R. An analogous problem exists for updates and deletions of records.

What might seem like a fairly exotic scenario is not uncommon at all with current registry implementations and regularly causes user-visible differences between the content of different registries. Some mitigation is possible if harvesters use the time of the last-but-one harvest to constrain their incremental queries. The correct solution, though, is that publishing registries set the ingestion time as the updated timestamp for their records. As the condition outlined above is the result of a straightforward implementation, however, we believe in the medium term a more robust method for incremental harvesting, presumably based on monotonously incrementing IDs, should be put in place within *OAI-PMH*, to make the straightforward implementation also a race-free one.

Another sometimes misunderstood feature has to do with *sets*. These are a feature of *OAI-PMH* that lets archive operators define subsets of their data holdings sharing some property. The VO's registry interface standard defines one such set that must be supported by all registries, `ivo_managed`. This set is defined to comprise all records that originate from the registry *and* should be visible in a searchable VO registry. The idea behind this is that a harvesting registry can constrain its queries to `ivo_managed` and will not see records from other registries even for registries harvesting other registries. Note that set membership is a property of a registry, not of a record, so information on set membership is lost at harvesting time.

6. Authorities

When, as in the VO, the creation of identifiers is distributed, there needs to be a mechanism ensuring uniqueness, which in the case of the VO Registry means making sure that no identifier is assigned to two different resources. In the VO, this mechanism is founded on the notion of *authorities*, which are entities creating IVORNs. As such, they are akin to DOI's prefixes.

As with DOI registrants owning prefixes, each authority is assigned a namespace, within which the authority is free to create new names, as long as some basic syntactic rules are followed. Full identifiers are then a combination of the authority identifier and the local part. As long as the IVOA makes sure that authority identifiers are unique and each authority ensures uniqueness *within their namespace*, the system yields globally unique identifiers.

Technically, authority identifiers are IVORNs (as introduced in Section 2) that just consist of the scheme and the URI authority part, for instance, ivo://ivoa.net. By *Registry Interfaces*, this must already be a valid IVORN, i.e., refer to a resource record, which in this case must be of the type `vg:Authority`. Resource records of this type ("authority records" in the following) are an "assertion of control over a namespace represented by an authority identifier" (Benson et al., 2009). In practice, the metadata should describe what organisational detail suggests the creation of a new authority. In consequence, the contact would be the person responsible for ensuring the uniqueness of the local parts.

In addition to the usual *VOResource* pieces of metadata – discussed in detail in Section 7 – authority records have exactly one `managingOrg`. This is the organisation that is responsible for an authority, and the distinction from the authority itself is somewhat subtle and best illustrated by an example: an observatory with an infrared unit and an ultraviolet unit that want to avoid having to negotiate before minting identifiers could claim the authorities `infrared.sample`, `ultraviolet.sample`, and `sample`. The observatory itself would then be ivo://sample/org, and it would be the managing organisation for all the authorities. All authority records would also list "The sample observatory" (or similar) as their publisher.

Note that URI authorities are opaque and unstructured, which means that clients are not supposed to infer any relationship from the fact that `sample` is contained in `infrared.sample`. There has been a recommendation to re-use DNS names as authority IDs, which has been largely ignored, probably because it tends to make IVORNs unnecessarily long. Today, we would suggest to base authority names on the names of national VO projects where available.

In *Registry Interfaces*, the burden of ensuring the uniqueness of the authority names is put on the publishing registries: "Before the publishing registry commits the [authority] record for export, it must first search a full registry to determine if a vg:Authority with this identifier already exists; if it does, the publishing of the new vg:Authority record must fail". Given the delays involved in harvesting, this procedure obviously has very real issues with race conditions, and to our knowledge, no engine for publishing registries implements such a check.

Compared to creating and operating DOI registrants, the creation and operation of VO authorities is thus simple, cheap and quick. The downside of this is that plain IVORNs do not work as persistent identifiers as laid out in Section 2.

The construction also implies that only one registry is accepted as the source for registry records under the authority (but a given registry can manage multiple authorities). Full registries can use this mapping from authorities to their managing registries to decide whether to ingest records they harvest when harvesting full registries either complementary to evaluating `ivo_managed` or instead of it, which has in the history of the VO Registry at times been more stable.

While name clashes in authorities at the time they are created have not been a problem in practice, it has frequently happened that as authorities sometimes moved from one registry to another, the releasing registry failed to drop its declaration of managing the departing authority, or did not update the record's modification date, which meant that incremental harvests would miss the update. This then means that two or more registries claim to manage a single authority, which introduces severe inconsistencies in the Registry, in particular as regards the continual resurrection of "zombie" records long deleted at the registry rightfully managing the authority.

At this point we believe the way to ensure a bijective mapping between authorities and their managing registries is its manual curation at the RofR, as the updated resource record from the accepting registry comes in and the conflicting claims of authority can be diagnosed.

7. VO resource records

Following *RM*, VO resource records contain a fairly comprehensive set of metadata. All resource records must have a title, an identifier, and a status as well as information on its content and the curation. They also have timestamps for the creation and the last update of the resource record. Additional optional metadata includes a short name (primarily for use in cramped displays) and validation information (cf. Section 9).

Content metadata consists of subjects – keywords which are supposed to be drawn from the IVOA thesaurus (Various, 2009) –, a human-readable description, the URL of a reference page giving more information about the resource (the *reference URL*), as well as optionally a bibliographic source – this is what should be referenced if the data is used – and some additional ancillary information. Content also allows defining relationships to other resources, examples for which include "mirror-of" or "service-for", which is particularly interesting for data collections to declare services allowing access to them.

Curation metadata gives a simple provenance of a resource: who has created it – to a first approximation, this usually is the "authors" –, who has published it, who can repair it. Curation also lets publishers specify dates relevant to the history of the resource itself (as opposed to the resource record), as, for instance, major data additions, schema changes, or the application of corrections for errata.

Resource records also have types, and certain types have additional metadata. As can be seen from Table 1, the overwhelming majority of resources in the current VO registry are of type `vs:CatalogService`.[5] These are access services for entities with sky coordinates, and most VO-compliant catalogue, image, or spectral services will use this type.

[5] Following widespread practice, we abbreviate the namespaces *VOResource* types come from with their "canonical" prefixes. A review of this, including a translation from prefixes to their namespaces, is given in Section 4 of Demleitner et al. (2014).

Table 1
Distribution of resource types in April 2014, as obtained by the prototype implementation of *RegTAP* operated for GAVO in Heidelberg (see http://dc.g-vo.org/browse/rr/q). The XML prefixes are as in Section 4 of Demleitner et al. (2014). Other includes deprecated or experimental types.

res_type	*N*
vs:CatalogService	13 706
vs:DataCollection	144
vg:Authority	131
vr:Organization	76
vr:Service	48
vs:DataService	29
vg:Registry	24
vstd:Standard	7
vstd:ServiceStandard	4
Other	153

In addition to basic *VOResource* metadata, catalogue services can contain additional information on the facility and the instrument that produced the data, whether the data is public or proprietary, on the area covered by the data contained on the sky, and on the structure of the table that feeds the service. Catalogue services share this metadata with `vs:DataCollection`.

In contrast to data collections, however, catalogue services have capability metadata, which in particular lets clients work out what protocols are available at what network endpoints. Note that capability types and resource types are largely decoupled, and no rules are enforced as to what resource types are allowed for which capabilities if a resource type allows capabilities at all. As capabilities are a fairly complex part of *VOResource*, we defer their closer discussion to Section 8.

A `vs:DataService` record is like `vs:CatalogService`, but without claiming to be based on some tabular structure. In retrospect, it seems doubtful that this distinction should be reflected in the resource type, as witnessed by its low and inconsistent use.

The interplay between `vg:Authority`, `vr:Organization`, and `vg:Registry` was discussed in Section 6, and *VOResource* just follows the roles laid out there: `vg:Authority` in addition to the basic metadata just gives the organisation that manages the authority, `vr:Organization` allows the specification of the organisation's facilities and instruments, and `vg:Registry` lists the authorities it manages, whether it is a full registry, and it has capabilities. Whether a registry is searchable or publishing or both is determined by its capabilities in *Registry Interfaces*. In *RegTAP*, data model identifiers from *TAPRegExt* are used for registry API discovery instead.

While few in number, records of types `vstd:Standard` and `vstd:ServiceStandard` are nevertheless important. They serve as destinations for references to standards as required in, e.g., capability records as discussed below. Such records allow the declaration of the various versions of a standard, associated XML namespace URIs, and also the declaration of terms. This latter feature provides a relatively lightweight way to generate IVORNs for certain concepts standards might need. In the registry extension for TAP (Demleitner et al., 2012), for example, this mechanism is used to introduce identifiers for output formats not distinguishable by the MIME type. Service standard records, in addition, allow a simple specification of a standard service's interface.

We finally mention the status attribute of *VOResource* records. It is distinct from but related to *OAI-PMH*'s `status` element optionally present in OAI headers; there, `status` takes the single value `deleted`, which should cause a harvesting registry to remove a resource record with the same identifier it may have stored from previous harvests (provided it uses the `ivo_vor` metadata prefix consistently). As VOResource describes the resource rather than the resource record, its `status` attribute in addition can assume the values `active` (which for resource

records is implied by the fact that they can be harvested) and `inactive`. This latter value is intended as a measure for publishers of third-party resource records when they suspect a resource registered through them has gone unmaintained but do not want to remove the resource record entirely. It is a feature rarely used, and the upcoming Registry APIs do not expose inactive resources to clients, since to them nonresponsive services coming back from registries are an annoyance regardless of prospects for the service's restoration.

8. Capabilities

Resource types that offer endpoints for interaction (services, registry) also contain zero or more capability elements. Capabilities essentially are *VOResource*'s way to describe the possible interactions with a resource.[6]

VOResource's basic capability element consists of optional validation information, and optional human-readable description, and zero or more interfaces.

The interfaces are again typed, with most interfaces in the current VO being one of `vs:ParamHTTP` – an interface for operation by HTTP and HTTP request parameters (about 64%) – and `vr:WebBrowser` – services based on HTML forms (about 35%). The remaining interfaces are a few SOAP-based services, the special OAIHTTP type used by publishing registries, and some types from abandoned standards.

Interfaces have one or more access URLs, where we expect that the next version of *VOResource* will restrict this to exactly one. In addition, a role attribute should be set to `std` if the interface is a standard interface for the standard the capability claims to implement. In that case, a version attribute can give the version of this standard. In current VO practice, this version attribute is typically ignored, as incompatible standards are told apart by the standard identifier of the capability.

Derivations of `vr:Interface` may have additional properties. In particular, `vs:ParamHTTP` declares a result type – supposed to be a MIME type – and the input parameters with their names, UCDs, and types, expressed in a simplified type system. This is a cross-protocol way of discovering the parameter metadata which should be provided in addition to protocol-specific means. Compared to the parameter declarations emitted from metadata queries in the VO's image and spectral access protocols SIAP and SSAP (Tody and Plante, 2009; Tody et al., 2012), parameter declarations in interfaces are less expressive, since the VOTable PARAMs employed in SIAP/SSAP metadata can have VALUES children giving ranges or possible values for enumerated parameters. It is somewhat unfortunate that the same kind of information is exposed in two non-equivalent ways.

In addition to these basic capability metadata, registry extensions can define capabilities with richer metadata. For instance, *SimpleDALRegExt* defines things like test queries, limits to search and response sizes, but also the kind of data contained, which for the image access protocol SIAP declares whether the service returns cutouts, pointed observations, mosaiced images, or is an atlas-type service. The most complex capability structure so far is the one for the Table Access Protocol TAP (*TAPRegExt*), which exposes many aspects of the TAP service and the languages supported by it. In the context of a paper on the registry, TAPRegExt's `dataModel` element deserves particular attention. It contains an IVORN of a standard defining a data model, more specifically a set of relational tables. This can be used to locate TAP services having these tables. Both Obscore – a table schema for observational products, Louys et al. (2011) – and the upcoming *RegTAP* standard use this mechanism to enable service discovery.

Capabilities are not only used directly in the registry. The VOSI and DALI standards (Grid and Web Services Working Group, 2011; Dowler et al., 2013) mandate that services should also emit the capability elements on a specialised endpoint next to the science endpoints. An example for where these endpoints are already in everyday use is again TAP, where clients determine the details of a TAP service (user defined functions, support for optional features, output formats, limits, etc.) without having to consult a registry.

9. Validation

In a distributed system in which many parties operate services, partly using custom implementations, it is inevitable that not all services actually comply to the standards they claim to implement. With a complex system like the VO Registry, it is not trivial to even write correct and complete resource records, let alone follow all rules ensuring that a publishing registry fits into the whole system. Hence, validation on many levels is crucial for maintaining the integrity of the VO.

As regards the *VOResource* records themselves, their validity essentially is equivalent to their compliance to the XML schema files that accompany the pertinent standards. For a publishing registry, a large number of further properties need to be checked, for instance a correct implementation of *OAI-PMH*, the definition of the authorities managed by the registry, the support of the `ivo_managed` set, and so forth.

A service performing such a validation is operated at the RofR, and it has proven instrumental for building a working Registry system. In particular, publishing registries that try to enlist themselves in the RofR are validated and can only enter if they are valid.

Registries may become non-compliant after this initial validation due to software updates or, more commonly, invalid registry records entering the set of resource records. No automatic re-validation is taking place, and registries that become invalid are not removed from the RofR. Relying on the registry operators to re-validate and repair their services has so far proven sufficient for keeping the VO Registry operational.

There is, however, a second and much larger aspect to validation: resource validation. This is another case in which the distinction between resource record and the resource itself becomes relevant—a valid resource record might very well describe a service that does not comply to the underlying standard. Validating a resource means examining as many aspects of its operation as possible. While this validation can in principle be performed by anyone, a publishing registry is a natural place for the operation of a service validator: (a) it already has the metadata available; (b) it has a means to disseminate its results.

As to (a), this metadata obviously includes the access URL and the standard implemented. However, meaningful validation typically requires additional metadata, in particular parameters that must return a non-empty response. *SimpleDALRegExt* contains elements designed for that purpose. For instance, the cone search capability has a `testQuery` element that separately lists values for the `RA`, `DEC`, and `SR` parameters that VO cone searches require. In actual use, it turned out that separating out the individual parameters of protocols did not significantly help either validators or other VO components. In the most recent simple DAL extension, the one for SSAP, `testQuery` hence admits the specification of a complete query string otherwise opaque to the validator.

As to (b), *VOResource* introduces a validation type that allows operators of validators to communicate their results. It consists of a numeric code from *RM* and a mandatory URI identifying the validating entity. The numeric code currently ranges between

[6] An exception to the interact-through-capability concept is the `accessURL` child of `vs:DataCollection`, which allows, which allows retrieval of the data and is a top-level attribute of the resource.

0 – "has a description that is stored in a registry" – and 4 – "meets additional quality criteria set by the human inspector" –, where from 2 up there is a requirement that the resource described exists and has been "demonstrated to be functionally compliant".

A resource record may contain validation information for both the full record and for a single capability. While the exact semantics of this distinction is not easy to define, the rough guideline from *VOResource* suffices for a useful interpretation. According to this, when a validation level is given for a resource, the "grade applies to the core set of metadata", whereas "capability and interface metadata, as well as the compliance of the service with the interface standard, is rated by validationLevel tag in the capability element".

Validation information is different from the rest of the resource record in that it is the only part designed to be changed by a third party on the way from the resource record author through publishing and searchable registry to the resource record consumer. It is also the only piece of information that a harvester should accept from a resource record it harvests from somewhere other than the originating registry.

As almost all other aspects of the VO, validation is distributed. Conceptually, everyone is free to offer a harvestable registry handing out validity assessments. In actual experience, validity assessments actually differ between various validating entities, for example because the feature sets exercised by the various validators are different. Several organisations in the VO operate validators, for instance, the Observatoire de Paris (Savalle and Le Sidaner, 2011), which also keeps a history of the performance of services such that it is easy to diagnose services that have been unresponsive or severely degraded for extended periods of time.

10. Conclusions

The Registry is the Virtual Observatory's answer to the need for structured, global, and detailed resource discovery. It exposes to clients a wealth of metadata while not introducing a single point of failure. This is enabled by a strictly defined metadata format, the use of standard protocols in the communication between registries, judicious use of cross-harvesting, authority management, and continuous validation.

The paper reviews how a set of standards by both the IVOA and external communities lay the foundations for the whole Registry system consisting of (cf. Fig. 1 for a graphical representation of this):

- publishing registries run by the providers of the science services (or on their behalf) that inject the resource records in a flexible and extensible metadata format,
- searchable registries that harvest the publishing registries (and potentially each other),
- a single registry of registries facilitating the initial discovery of registries (but is not important in daily operation of the Registry, as its content is also available from all full registries),
- and user interfaces and APIs provided by the searchable registries exposing the Registry contents to queries and inspection (these will be discussed in a forthcoming article).

The Registry has additional roles to play on top of resource discovery. For example, information on the publishers, creators, and maintainers of the resources is available in a standardised way. This lets client software present the VO user with information on who to credit in a study using data obtained from registered services, or to find out where to direct questions in case of technical malfunction or scientific issues.

The success of the development of a resource, and in particular service, registry within the VO may also be seen from the adoption of the underlying technologies in similar projects in other fields, for instance a VO-like effort in molecular and atomic spectroscopy called VAMDC (Walton et al., 2011).

Acknowledgement

This work was in part supported by the German Astrophysical Virtual Observatory GAVO, BMBF grant 05A11VH3.

References

Accomazzi, A., 2011. Linking Literature and Data: Status Report and Future Efforts. In: Accomazzi, A. (Ed.), Future Professional Communication in Astronomy II. In: Astrophysics and Space Science Proceedings, vol. 1. p. 135.

Arviset, C., Gaudet, S., the IVOA Technical Coordination Group, Nov. 2010. IVOA architecture. IVOA Note. URL http://www.ivoa.net/documents/Notes/IVOAArchitecture.

Benson, K., Plante, R., Auden, E., Graham, M., Greene, G., Hill, M., Linde, T., Morris, D., O'Mullane, W., Rixon, G., Stébé, A., Andrews, K., 2009. IVOA registry interfaces version 1.0. IVOA Recommendation. URL http://www.ivoa.net/Documents/RegistryInterface/.

Berners-Lee, T., Fielding, R., Masinter, L., Aug. 1998. Uniform resource identifiers (URI): Generic syntax. RFC 2396. URL http://www.ietf.org/rfc/rfc2396.txt.

Demleitner, M., Dowler, P., Plante, R., Rixon, G., Taylor, M., Aug. 2012. TAPRegExt: a VOResource schema extension for describing TAP services, version 1.0. IVOA Recommendation. URL http://www.ivoa.net/Documents/TAPRegExt.

Demleitner, M., Harrison, P., Molinaro, M., Greene, G., Dower, T., Perdikeas, M., 2014. IVOA registry relational schema. IVOA Proposed Recommendation. URL http://www.ivoa.net/documents/RegTAP/.

Dowler, P., Demleitner, M., Taylor, M., Tody, D., Nov. 2013. Data access layer interface, version 1.0. IVOA Recommendation. URL http://www.ivoa.net/documents/DALI.

Grid and Web Services Working Group, 2011. IVOA support interfaces version 1.0. URL http://www.ivoa.net/Documents/VOSI/index.html.

Hanisch, R., IVOA Resource Registry Working Group, NVO Metadata Working Group, 2007. Resource metadata for the virtual observatory. IVOA Recommendation. URL http://www.ivoa.net/Documents/latest/RM.html.

Harrison, P., Burke, D., Plante, R., Rixon, G., Morris, D., May 2012. StandardsRegExt: a VOResource schema extension for describing IVOA standards, version 1.0. IVOA Recommendation. URL http://www.ivoa.net/Documents/StandardsRegExt.

ISO Technical Committee 46, 2012. ISO 26324:2012 information and documentation—digital object identifier system. URL http://www.doi.org/idf-members/ISO_Standard/ISO_263234_English_Final.pdf.

Kunze, J., Baker, T., Aug. 2007. The Dublin Core metadata element set. RFC 5013. URL http://www.ietf.org/rfc/rfc5013.txt.

Louys, M., Bonnarel, F., Schade, D., Dowler, P., Micol, A., Durand, D., Tody, D., Michel, L., Salgado, J., Chilingarian, I., Rino, B., de Dios Santander, J., Skoda, P., 2011. Observation data model core components and its implementation in the Table Access Protocol, version 1.0. IVOA Recommendation. URL http://www.ivoa.net/Documents/ObsCore/.

Plante, R., Jun. 2007. The registry of registries. IVOA Note. URL http://www.ivoa.net/Documents/latest/RegistryOfRegistries.html.

Plante, R., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T., Rixon, G., Stébé, A., Feb. 2008. VOResource: an XML encoding schema for resource metadata version 1.03. IVOA Recommendation. URL http://www.ivoa.net/documents/latest/VOResource.html.

Plante, R., Delago, J., Harrison, P., Tody, D., May 2012. SimpleDALRegExt: Describing simple data access services, version 1.0. IVOA Proposed Recommendation. URL http://www.ivoa.net/Documents/SimpleDALRegExt.

Plante, R.L., Greene, G., 2008. Chapter 41: An Overview of the Registry Framework. In: Graham, M.J., Fitzpatrick, M.J., McGlynn, T.A. (Eds.), The National Virtual Observatory: Tools and Techniques for Astronomical Research. In: Astronomical Society of the Pacific Conference Series, vol. 382. p. 445.

Plante, R., Linde, T., Williams, R., Noddle, K., Mar. 2007. IVOA identifiers, version 1.03. IVOA Recommendation. URL http://www.ivoa.net/documents/latest/IDs.html.

Plante, R., Stébé, A., Benson, K., Dowler, P., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T., Rixon, G., Dec. 2010. VODataService: a VOResource schema extension for describing collections and services version 1.1. IVOA Recommendation. URL http://www.ivoa.net/Documents/VODataService/.

Savalle, R., Le Sidaner, P., 2011. A Services Validator for IVOA registries. Talk given at the May 2011 IVOA Interop Napoli. URL http://wiki.ivoa.net/internal/IVOA/InterOpMay2011Registry/ServicesValidatorForIVOARegistries.pdf.

Tody, D., Dolensky, M., McDowell, J., Bonnarel, F., Budavari, T., Busko, I., Micol, A., Osuna, P., Salgado, J., Skoda, P., Thompson, R., Valdes, F., 2012. Simple spectral access protocol version 1.1. IVOA Recommendation. URL http://www.ivoa.net/documents/SSA/20120210/REC-SSA-1.1-20120210.htm.

Tody, D., Plante, R., 2009. Simple image access specification. IVOA Recommendation. URL http://www.ivoa.net/Documents/latest/SIA.html.

Various, 2002. The open archives initiative protocol for metadata harvesting, version 2.0. URL http://www.openarchives.org/OAI/openarchivesprotocol.html.

Various, 2009. IVOAT Thesaurus. URL http://www.ivoa.net/rdf/Vocabularies/vocabularies-20091007/IVOAT/IVOAT.html.

Walton, N.A., Dubernet, M.L., Mason, N.J., Piskunov, N., Rixon, G.T., 2011. VAMDC: The Virtual Atomic and Molecular Data Center. In: Evans, I.N., Accomazzi, A., Mink, D.J., Rots, A.H. (Eds.), Astronomical Data Analysis Software and Systems XX. In: Astronomical Society of the Pacific Conference Series, vol. 442. p. 89.