



International
Virtual
Observatory
Alliance

Lessons Learned Using the VOResource XML Schemas in the NVO Version 1.0

IVOA Note 5 March 2004

This version:

<http://www.ivoa.net/Documents/Notes/RMExp/RMExp-20040305.html>

Latest version:

<http://www.ivoa.net/Documents/latest/RMExp.html>

Previous versions:

none

Authors:

[Raymond Plante](#) (NCSA)

[Matthew Graham](#) (Caltech)

[Gretchen Greene](#) (STScI)

[Bob Hanisch](#) (STScI)

Jeongin Lee (HEASARC)

[Wil O'Mullane](#) (JHU)

[Ramon Williamson](#) (NCSA)

Abstract

During 2003, the [National Virtual Observatory \(NVO\)](#) project in the US set out to prototype the VO Registry framework as described by the [IVOA Registry Working Group](#). A key goal was to demonstrate use of the [VOResource XML schemas](#) as a format for exchanging descriptions of resources in registries. While we feel that the effort was overall a successful validation of the framework and the XML schemas, it was particularly useful for understanding where the challenges lie. This document looks at our lessons learned in using the [VOResource XML schemas](#), focusing on what made using

the schemas difficult. This document avoids drawing too many conclusions about what *should* be done to improve the schemas; rather, it is meant to serve as input to a discussion of a possible future revision. We recognize that some of the complexity inherent in the schemas may be necessary to adequately address the needs of the various IVOA projects. However, we do outline broad options for addressing these concerns.

Status of this document

This is a Note. The [first release of this document](#) was 5 March 2004.

This is an IVOA Note expressing suggestions from and opinions of the authors. It is intended to share best practices, possible approaches, or other perspectives on interoperability with the Virtual Observatory. It should not be referenced or otherwise interpreted as a standard specification. A list of [current IVOA Recommendations and other technical documents](#) can be found at <http://www.ivoa.net/Documents/>.

Acknowledgements

This document has been developed with support from the [National Science Foundation's](#) Information Technology Research Program under Cooperative Agreement AST0122449 with The Johns Hopkins University.

Contents

- [Abstract](#)
- [Status of this document](#)
- [Acknowledgments](#)
- [1. Introduction](#)
 - [1.1. A Review of Successes](#)
 - [1.2. The VOResource Data Model](#)
 - [1.3. Purpose of this Document](#)
- [2. Challenges](#)
 - [2.1. The Learning Curve](#)
 - [2.2. Hierarchy vs. Flatness](#)
 - [2.3. Size and Complexity](#)
 - [2.4. Referencing Multiple Namespaces and Schema Files](#)
- [3. The Use of Software Binding Tools](#)
- [4. Addressing the Difficulties](#)
- [Appendix A. Changes from previous versions](#)
- [References](#)

1. Introduction

During 2003, the [National Virtual Observatory \(NVO\)](#) project in the US set out to prototype the VO Registry framework as described by the [IVOA Registry Working Group \[IVOA-RWG\]](#). An overview of this effort is described in our ADASS XIII paper [[Plante et al. 2004](#)]. In summary, descriptions of resources were created and published in so-called *publishing registries* at NCSA, Caltech, and HEASARC. A *searchable registry* at Johns Hopkins/Space Telescope collected these descriptions via a process called *harvesting* and loaded them into a searchable database. Harvesting was enabled using the [Protocol for Metadata Harvesting \(PMH\) \[OAI-PMH\]](#) developed by the [Open Archives Initiative \(OAI\)](#). Applications, such as the [Data Inventory Service](#), are able, then, to query the searchable registry to discover data and services that match desired constraints.

Descriptions of the resources were encoded using the [Resource Metadata XML schemas \[VOResource 2003\]](#) developed by the [IVOA Registry Working Group](#). Some of the registries stored these descriptions internally in this format as well; this allowed them to use an off-the-shelf OAI tool to expose the descriptions via an PMH interface. Thus, the main challenge for the publishing registries was creating the XML documents; this was enabled at all of the registries via a web-based form that a data provider can fill out to describe a resource. The main challenge for the searchable registry developers were to creating harvester that could retrieve the OAI resource records, extract the VOResource metadata from the OAI envelope, and parse the metadata for loading into the database. In principle, the searchable registry would also emit VOResource-compliant metadata in response to search queries; however, in lieu of a standard to do this, the search interface was optimized for our sole client, the [NVO Data Inventory Service \[NVO-DIS\]](#).

1.1. A Review of Successes

Although this document focuses on the difficulties using the VOResource schemas, it is worth noting that we feel that overall our prototyping was a successful validation of the IVOA registry framework. In particular, we demonstrated:

- Form-based creation of resource descriptions that hid the details of VOResource from the user.
- Easy and natural integration of the VOResource metadata into the OAI protocol.
- Rapid development of OAI interfaces leveraging existing tools.
- Successful harvesting of VOResource records between registries.
- Successful exchange of VOResource descriptions across IVOA projects (via the AstroGrid and CDS registries).
- Parsing and loading VOResource metadata into a relational database.
- Discovery of resources via searching based on metadata constraints through a web service.
- Use of resource metadata to support a real end-user application.

It is also important to point out the success of the IVOA community process that not only developed the framework design but also the prototype standards for resource identifiers [[PR-Identifiers](#)] and the [VOResource XML schemas](#). Even though these were *prototype standards* (i.e. still in the process of development and approval), they served as a vital

rudder for assembling our prototype application. The fact that we were able to make effective use of these standards is a validation of the process itself.

1.2. The VOResource Data Model

In the VOResource data model, metadata to describe resources are segregated into a core schema (<http://www.ivoa.net/xml/VOResource/v0.9>), which applies to all resources, and various extensions that specialize in specific types of resources. This allows extensions to evolve independently from the core. Each extension schema is defined in a separate file and has its own namespace. All metadata concepts that appear in the core should be based on those defined in the general Resource Metadata document (RM, [Hanish et al. 2003]). The extension schemas should draw on RM concepts where there is overlap, but in general the extensions introduce new concepts.

The primary object in the model is the *Resource*, represented by the generic `<Resource>` element. The contents of this element constitutes the core metadata that describes all types of resources. This element is *extended* to describe specific types of resources; specific resource elements currently defined include `<Organisation>`, `<Project>`, `<Registry>`, `<Authority>`, and `<Service>`. Each resource extension element can add additional metadata that is specific to that resource. Given that `<Resource>` is so generic, it is more common to describe a resource using one of the extension elements.

In specific terms of XML Schema, the `<Resource>` element is extended in a two step process. First, one extends the `ResourceType` XML type using the XML Schema type extension mechanism. Next, one defines the extension element (e.g. `<Service>`) as having this extended type *and* declaring as a member of the substitution group, "Resource." This use of the substitution group means that the extension element can be used anywhere the `<Resource>` element is used.

1.3. Purpose of this Document

The remainder of this document focuses on what, in our experience, made using the VOResource schemas difficult. The purpose is to provide input to a discussion of a possible future revision. This document avoids making too many conclusions about what *should* be done to improve the schemas; rather, it attempts to assemble the major issues to be considered. We recognize that some of the complexity inherent in the schemas may be necessary to adequately address the needs of the various IVOA projects. Thus, a prudent revision need not address all of the challenges described here.

2. Challenges

In this section, we examine some of the challenges and difficulties that we, as on-the-ground implementors and data providers, have encountered while creating resource descriptions using the VOResource schema and fill our registries.

2.1. The Learning Curve

For much of our deployment period, the only practical documentation that existed on VOResource helpful for creating compliant descriptions was [an example file \[adil-v0.9.xml\]](#) posted to the Registry Working Group Twiki. While it contained examples of several types of resource descriptions, it was not targeted specifically to use of descriptions with registries. General XML tools, such as XMLSpy and [xs3p](#), provided "reference manual"-type descriptions of the schemas; these are somewhat helpful because the schemas themselves were fully documented, and these tools incorporated this documentation into their human-readable format. However, even for those who know where to consult these documents, it's unclear how helpful they were to people in the absence of a concept of the big picture.

What was clearly needed was a general tutorial document that stepped through a few annotated examples, targeted specifically to the registry applications. Such a document was in the works; however, in the time leading up to deployment, priority was given to tracking down various bugs in the schemas and their publishing. An important companion to that document is a [metadata dictionary \[Plante 2003\]](#) (which Plante developed as an adaptation to the xs3p-generated reference manual); using the documentation in the schema, this provides users with a dictionary for looking up the specific meaning and syntax of any VOResource element. This dictionary was published late in the deployment stage, so it is unclear if it, on its own, was useful to developers.

In the absence of a tutorial document, several of us looked to the RM document for guidance. This led to considerable confusion because the metadata names and structure defined in the schemas do not match exactly to what is in the RM. Although, the RM does state that particular encoding may diverge in name and structure from what is defined in the RM, some feel that the divergence is greater than necessary and thus a source of unnecessary confusion.

2.2. Hierarchy vs. Flatness

One of the reasons for VOResource's deviation from the RM document is to take advantage of XML's ability to organize information hierarchically. The purpose of employing a hierarchical organization is to aggregate information into logical objects. For example, all the information about the resource's Coverage is contained within the `<Coverage>` element. This can be useful for an application that wishes to handle, say, Coverage objects independently from a resource description.

A hierarchical structure does have some disadvantages, however. One is that it can be a bit tedious programmatically accessing a piece of information deep within the hierarchy using an in-memory tree (either DOM or customized "binding" objects): this may require several function calls to traverse the levels of the hierarchy. Such calls can be particularly verbose if one or more levels of the hierarchy are optional; one needs to confirm that the layer exists before extracting its contents.

We also found SAX parsing is a bit more complicated than it might be because certain elements (e.g. `<Title>` and `<Name>`) are reused in multiple locations within the hierarchy. In these cases, the basic concept the element represents is the same but modifies different things (e.g. the title of a publisher versus the title of a resource). This requires that the parser must not only look for an element with a particular name, it must keep track of where in the hierarchy the element is found in order to grab the right one.

The biggest difficulty with the hierarchical model imposes is that it does not map easily to a relational database which was used to implement the searchable registry. If the RDBMS tables are to be normalized, one would typically store each layer (or object) from the XML document into a separate table. Repeatable elements (e.g. `<Format>`, `<Subject>`) would also need to be further segregated into separate tables. Consequently, simple queries will invariably require joins across multiple tables. A flatter model would make the mapping to an RDBMS easier.

It's worth pointing out that the question of hierarchy versus flatness primarily is an issue that effects how easily XML tools work with our schema. However, it also can affect a user's overall understanding of the model, particularly if one is familiar with the relatively straight-forward concepts in the RM. For example, the name of a publisher is not the value of the `Publisher` element, but rather `Publisher/Title`. The hierarchy, thus, can obscure the actual location of the most commonly used values. This is also an issue of general complexity.

2.3. Size and Complexity

The hierarchical nature of the schemas can be one aspect of the schema's complexity. The number of elements defined in the schema is another. Overall complexity can help determine how easy it is to work with the schema. In this section, we enumerate some ways the overall size and complexity of the schema can increase the effort necessary to use the schemas.

First, most of the elements are optional, and many of them not being used by our current applications. This creates a challenge for both providers--those that must create the resource descriptions--and application developers--those that make use of them. First, for providers, it becomes unclear which of the elements are most important to provide to be useful. For developers, more elements often require more code to support. For example, with a searchable registry, more elements means more data that must be stored, mapped into and out of a database. Because this mapping is non-trivial (as mentioned in the previous section), more hand coding is necessary.

It can be argued that the larger the schema, the harder it is to comprehend and support. It may, therefore, make sense to remove or otherwise consolidate elements that in practice are rarely used. There are several elements that might be considered in that category. For example, there are several elements of the type `ResourceReferenceType` (e.g. `Facility` and `Instrument`). This type has four child nodes, `Identifier`, `Title`, `Description`, and `ReferenceURL`, of which only `Title` is required. This type is intended for referring to

other resources that *may or may not* be described in an external, registered resource description. For **Facility** and **Instrument**, providers have *not* registered them separately (because there is no driving need at the moment). As a result, the **Identifier** value is not set; neither, typically, are **Description** and **ReferenceURL**. If we concluded that these rarely-used children are not needed we could simplify the **Facility** and **Instrument** elements greatly by defining their type to be `xsd:string` to simply hold the title. Obviously, with this example, what would be sacrificed would be flexibility, particularly to future uses (e.g. when a facility is commonly registered independently).

2.4. Referencing Multiple Namespaces and Schema Files

A typical VOResource instance draws on multiple schemas, usually one extension schema that defines a specific type of resource (e.g. VOCommunity which defines the **Organisation** resource) and the core VOResource schema. In our prototype registry deployment, a set of descriptions from a single provider would typically have to draw on all six of the standard resource schemas defined.

The complexity introduced by the use of multiple namespaces made namespace use error-prone. When a namespace was mishandled, it was often difficult to determine the cause as the error message one usually saw was something like "schema not found" or "unexpected element." From the perspective of a data provider, who arguably should not have to be an XML expert to use the schemas, these messages are too obscure to easily track down the problem.

The difficulties we experienced with namespaces resulted primarily from three common errors:

- **Mismatched namespace URIs:** a namespace URI must match case-sensitively, letter-to-letter in every place it is used. A typo in the URI will not usually result in an error at the location of the typo, but rather where an element from that namespace is used.

There are five locations where the namespace URI is typically cited:

- the **targetNamespace** attribute within the schema (.xsd) file.
 - the **xmlns** attribute at the top of the XML document indicating the default namespace for the document.
 - the **xmlns** attribute elsewhere in the XML document used to switch to a new default namespace.
 - the **xmlns:prefix** attribute used to define namespace prefixes.
 - the **schemaLocation** attribute to indicate where to find the schema document associated with the namespace.
- **Missing or incorrect namespace prefix.** One must be aware of which schema an element is defined in.
 - **Insufficient support for locating schemas across XML tools and application contexts.**

VOResource uses a standard-but-optional mechanism for locating schemas in which the URI is a URL that resolves to the actual Schema file itself.

Applications can search this URL as a default location; however, not all XML tools support this mechanism. This avoids having to "hard-code" the schema locations in the XML documents.

However, support for this mechanism varied greatly among the tools in use.

Many of the errors we experienced related to namespaces were traced either to errors in the schema file themselves or in their deployment on the IVOA web site. (Varying support for the XML standard across XML tools often made finding these problems difficult.) These problems went away once the bugs in the schemas were fixed; thus, one might argue that these are no longer a problem for developers and users.

The prominent role that namespaces play in the resource description documents ultimately made the XML mark-up more complex and mysterious from the perspective of a resource provider than we would prefer. In particular:

- managing XML namespaces with namespace prefixes and the `xmlns` and `schemaLocation` attributes visually detracts the information being encoded. It makes the document "noisy".
- In practice, an XML instance document will often define namespace prefixes and locations for all six standard schemas, whether they were all used or not. This provides greater opportunity for error.
- In general, one has to pay close attention to which namespace an element belongs so that the correct prefix is applied.
- XML tools (silently) varied in their support for locating schemas. This included whether or not they would resolve the namespace URI as URL and whether they would search the local directory (important for development and debugging). Consequently, it was difficult to recommend an "always works" recipe for specifying namespace locations.

Some the difficulties we encountered have been fixed in the schema documents themselves, and others may constitute necessary complexity. However, life would be simpler for providers and developers if namespace management could be made less prominent and more transparent.

3. The Use of Software Binding Tools

Software binding tools, such as JAXB, Castor, and Microsoft's XSD, are a special class of tool we believe will be important for developing VO applications. Such a tool will automatically read a schema and create customized software classes for it. Getting our schemas to work with these tools, however, has been non-trivial. Our experience may simply be a reflection of the maturity of the tools and their documentation. It is perhaps not reasonable to expect, for example, our particular approach to metadata to be reflected in the examples that come with the tools' documentation, even if that approach is

considered valid and necessary. Thus, the difficulties we describe may be more useful as a "heads-up" for other developers than input into schema changes.

Using the binding tools is not straight-forward primarily for two reasons. Most important is the fact that substitution groups, which VOResource uses to support polymorphism, was not fully supported by these tools. (At the time of this writing, we believe that substitution groups are now supported by JAXB and MS-XSD. Castor can conditionally support them under certain usage patterns.) Second, it is not clear how to use these tools to create classes for an extension schema which draws on the core VOResource schema; that is, while it may be possible, this is not well documented. For example, through trial and error, we found that MS-XSD could support all the extension schemas if we generated all of the classes from all of the schemas all at once in one call to the XSD command.

4. Addressing the Difficulties

As stated in the [Introduction](#), the purpose of this document is to serve as input to a discussion of a revision of VOResource; that is, we hope that we can benefit from these lessons learned. To that end, we see three possible approaches that the IVOA RWG can take to address the concerns raised in this document:

1. **We do nothing.** We may decide to continue using VOResource for the foreseeable future, either because we feel that the difficulties are a necessary burden to support the functionality the current schemas provide or because we feel that the current schemas are sufficiently useful and that addressing the difficulties is not worth the effort at this time. We could potentially consider a revision again in a year's time. In the absence of a revision, we would concentrate efforts on documentaton, tools, and interfaces that make creating descriptions easier.
2. **We attempt a highly focused revision process.** The idea here is that a revision would start with existing schemas. By *highly focused*, we mean that we should only address existing concerns based on real experience (which would presumably include those described here) and not consider new functionality. The process would have to include a discussion of which concerns should be addressed.

The major motivation for this option is that by late 2004, we expect to be fairly entrenched with the VOResource schemas. We expect there to be several working registries in production mode, serving several thousands of records. The prospect of retooling these registries for a new core schema (and retraining their maintainers) may make a revision prohibitively expensive.

Given the IVOA goals for 2004, such a revision process would have to happen rapidly in the early spring, and be tested in real prototypes by spring and ready for IVOA review at the May 2004 Interoperability meeting.

3. **We start over from scratch.** Given the 2004 IVOA roadmap, this option might be chosen if it was felt that the greatly simplified schema was needed, so simplified that it a replacement could be generated and agreed upon in very short amount of time.

A stated requirement of the Virtual Observatory framework has always been low "buy-in" costs for data providers. With the current schemas, one can argue that the complexity is a real impediment for providers and application developers. It is our vision that it needs to be easier to understand and use these schemas than it is now. Regardless of which of the above paths we follow, more work on is needed on documentation and tools, which will take time. The question now is, can adoption of standard schemas be accelerated if they were revised to make them easier to use?

Appendix A: Changes from previous versions

Not applicable.

References

[adil-v0.9.xml]

[a sample VOResource instance file,](#)

<http://www.ivoa.net/internal/IVOA/IVOARegWp03/adil-v0.9.xml>

[Hanish et al. 2003]

Hanisch, Robert (ed.) 2003. [Resource Metadata for the Virtual Observatory](#), IVOA Working Draft,

<http://www.ivoa.net/Documents/PR/ResMetadata/ResMetadata.html>

[IVOA-RWG]

[The IVOA Registry Working Group Twiki Page,](#)

<http://www.ivoa.net/twiki/bin/view/IVOA/IvoaResReg>

[NVO-DIS]

[the NVO Data Inventory Service,](#) <http://heasarc.gsfc.nasa.gov/vo/data-inventory.html>

[Plante et al. 2004]

Plante, R., Green, G., Hanisch, R., McGlynn, T., O'Mullane, W., Williams, R., and Williamson, R. 2004. [Resource Registries for the Virtual Observatory](#), in ASP Conf. Ser., Astronomical Data Analysis Software and Systems XIII, in press, <http://bill.cacr.caltech.edu/usvo-pubs/files/VORegistries.pdf>

[Plante 2003]

Plante, R. 2003. [Metadata Dictionaries](#), from *XML Schemas for Resource Metadata: a Tutorial*, in preparation, with links to individual dictionaries for each schema, <http://nvo.ncsa.uiuc.edu/VO/schemas/vomdoc-v0.9/Note-RMSchemas.html#appA>

[PR-Identifiers 2003]

Plante, R., Linde, T., Williams, R. and Noddle, K. 2003. [IVOA Identifiers](#), IVOA Proposed Recommendation,

<http://www.ivoa.net/Documents/PR/Identifiers/Identifiers.html>

[OAI-PMH]

Lagoze, C., van de Sompel, H., Nelson, M. and Warner, S. 2002. [The Open Archives Initiative Protocol for Metadata Harvesting](http://www.openarchives.org/OAI/openarchivesprotocol.html), an Open Archives Initiative standard, <http://www.openarchives.org/OAI/openarchivesprotocol.html>
[VOResource 2003]
[VOResource: a Resource Metadata Schema](http://www.ivoa.net/twiki/bin/view/IVOA/IVOARegWp03#VOResource_a_Resource_Metadata_S), from the *IVOA RWG-WP3: Resource Metadata Twiki Page*,
http://www.ivoa.net/twiki/bin/view/IVOA/IVOARegWp03#VOResource_a_Resource_Metadata_S