

## Recommendations for Revisions to the VOResource XML Schema



## Lessons Learned in 2003

- 2003: prototyping the Registry Framework
  - Based on framework presented in Cambridge
  - NVO prototype described at ADASS 03
- Based on VOResource v0.9 schema and its extensions
- Prototyping largely a success
  - NVO Registry supports real app: Data Inventory Service
  - Support publishing, harvesting, searching
- Support for VOResource not without difficulty
  - Challenges outlined in IVOA Note: “Lessons Learned using VOResource in the NVO”



# VOResource Model v0.9

- Family of XML Schemas: core + extensions
  - Core model: VOResource-v0.9.xsd
    - XML serialization of the "Resource Metadata" standard
    - generic Resource and Service
  - Extensions covering specific kinds of resources
    - VOCommunity - organizations and projects
    - VODataServices - data collections and service types
    - VORegistry - registries
    - Standard services: SIA, ConeSearch
  - Each schema is standardized separately
- Extension model
  - Generic <Resource> element contains core metadata
  - Specific resource types inherit from core
    - adding metadata that only applies to that type
    - Inheritance & polymorphism based on substitution groups



# Challenges Using VOResource

- Learning Curve
  - Insufficient publisher oriented documentation
  - Substitution groups as an extension mechanism
  - Departure from names and structure from RM document was a source of confusion
- Complexity
  - Cost of hierarchy (vs. flat model)
    - Software is more complex
    - Difficult to map to relational database
    - Less clear where consumer expects to find information
  - Are all features necessary?
    - Cost of elements never used
- Support for tools
  - XML binding tools still maturing
  - Substitution groups support late
  - Debate regarding use of global elements

Addressing these requires fundamental changes



## Motivation for a Revision

- Real end-user applications that depend on VOResource are now appearing
  - NVO's Data Inventory Service, Publishing registries, Sky Portal
- Second generation standards
  - DAL, Data Models
- Expansion of the VO community
  - New VO-related projects, archives
  - NVO Summer School

**Entrenchment:** the more VOResource records in use, the harder it will be to make fundamental changes

**We have narrow window of opportunity to fix real problems**



## Proposal for Revision

- Metadata Telecon of registry developers:
  - Examine difficulties of VOResource data model
  - Propose solutions that can be quickly prototyped
  - Discuss action plan to migrate registries to revised schema
  - Focus on issues that affect core VOResource model
- Recommendations
  - Move from element-based to type-based model
    - Substitution groups → `xsi:type`
  - Adopt names and structure that is closer to RM standard
- Wait and see: hiding namespace prefixes



# Substitution groups vs. xsi:type

## Using substitution groups:

```
<Resource>
  <Title>...</Title>
  ...
</Resource>

<Service>
  <Title>...</Title>
  ...
  <Interface>...</Interface>
</Service>
```

## Using xsi:type

```
<resource>
  <title>...</title>
  ...
</resource>

<resource xsi:type="Service">
  <title>...</title>
  ...
  <interface>...</interface>
</resource>
```



# “Element-based” XML model

- Each metadata concept represented by a global element
  - Semantics: name has an absolute meaning
    - FITS keywords, Dublin Core, RDF?
- Compound concepts:
  - model in a global complex type
  - model filled with existing elements via “ref=“
  - define global element of compound type
- “Subclassing” an element:
  - define derived type from element’s type
  - define global element of derived type with substitution group set to the parent element

```
<xs:element name="Resource" type="ResourceType"/>
<xs:element name="Service" type="ServiceType"
  substitutionGroup="Resource"/>
```



# “Type-based” XML Model

- Based on Gerard Lemson’s “Model-based Schema”
  - Set of patterns for mapping UML to XSD
  - PPT posted on Twiki: [VOResource010RevNotes](#)
- Reusable component: global type
  - Semantics: name-meaning association is definition of type components
- Component concepts:
  - Types reused via “type=”
  - Elements are locally defined only
  - Application defines a (single) global element as necessary
- Using subclasses
  - Standard type derivation
  - Use xsi:type mechanism in instance document
    - When necessary



# Substitution groups vs. xsi:type

## Using substitution groups:

```
<Resource>  
  <Title>...</Title>  
  ...  
</Resource>
```

```
<Service>  
  <Title>...</Title>  
  ...  
<Interface>...</Interface>  
</Service>
```

## Using xsi:type

```
<resource>  
  <title>...</title>  
  ...  
</resource>
```

```
<resource xsi:type="Service">  
  <title>...</title>  
  ...  
<interface>...</interface>  
</resource xsi:type="Service">
```





## Type-based model: Advantages

- xsi:type better supported by tools
- Clearer to understand:
  - Avoid mystery of substitution groups
  - Instance documents make clear type-subtype relationship
- One global element
- Straight-forward mapping from UML-based model supported by tools
- Captures context-specific semantics



## Reducing complexity

- Reducing the hierarchy
  - V0.9 attempts to take advantage of XML structures
    - Provide potentially reusable nodes of related metadata
    - Support references to other resources that may be registered with an ID
  - Recommendation: flatten to match hierarchy suggested by RM standard
  - Simpler resource reference model:

```
<Publisher ivo-id="ivo://rai.ncsa/rai">
  NCSA Radio Astronomy Imaging Group
</Publisher>
```
- Names that match RM
  - V0.9 altered names to accommodate altered structure
  - Recommendation: return to RM names



# Strawman Revision: v0.10

- Schemas and examples available via Twiki:  
[VOResource010RevNotes](#)  
Examples:
  - [organisation.xml](#)
  - [registry.xml](#)
  - [authority.xml](#)
  - [collection.xml](#)
  - [webform.xml](#)
  - [conesearch.xml](#)
  - [sia.xml](#)
- Call for endorsement to incorporate recommendations into revision to VOResource
  - Settle v0.10 by June
  - NVO: desire to update registries & applications by end of summer



# Other questions

- Where to identify standard services:
  - Currently: specialization of Capability element  
[sia.xml](#)
    - Rational: element to collect service-specific metadata
  - Would people prefer to associate std service type as a sub-class of Resource?

```
<resource xsi:type="SimpleImageAccess">  
...  
</resource>
```

    - Lose Capability element (or not enforce its existence)



## Other questions

- Move Organisation Resource type into core VOResource; drop VOCommunity schema
  - “Project” resource type currently not used
  - Organisation is a “core” concept
    - Referred to in RM document
    - Registering Organisations is critical to tracing responsibility
      - IVOA ID is only unambiguous way to uniquely identify a Publisher  
`<Publisher>HST</Publisher>`
      - Organisation-specific metadata draws from RM  
Facility, Instrument

