International
Virtual
Observatory
Alliance

Discovering Data Collections Within Services

Version 1.1

Proposed Endorsed Note 2016-11-11

Working group
Registry
This version
http://www.ivoa.net/documents/discovercollections/20161111
Latest version
http://www.ivoa.net/documents/discovercollections
Previous versions
NOTE-1.0
Author(s)
Markus Demleitner, Mark Taylor
Editor(s)
Markus Demleitner
Version Control
Revision 4572, 2017-11-13 17:05:28 +0100 (Mon, 13 Nov 2017)
https://volute.g-vo.org/svn/trunk/projects/registry/discovercollections/discovercollections.tex

Abstract

This note proposes a general mechanism through which data collections exposed by services aggregating several of them can be registered and efficiently discovered in the Virtual Observatory Registry. We discuss the problem setting – in short, that the Registry must support both VO-wide discovery of services by type (“service enumeration”) and discovery by data collection (“data discovery”) – and some attempts for a solution that were found to be unsatisfactory. Based on this, a scheme employing special identifiers to annotate simple capabilities associated with data collections is described. We also detail the implications for both standard authors and Registry clients, possible shortcomings of the scheme, as well as measures required to ensure a smooth transition to the new scheme.

Status of This Document

Contents

Conformance-related definitions

The words “MUST”, “SHALL”, “SHOULD”, “MAY”, “RECOMMENDED”, and “OPTIONAL” (in upper or lower case) used in this document are to be interpreted as described in IETF standard RFC2119 (Bradner, 1997).

The *Virtual Observatory (VO)* is general term for a collection of federated resources that can be used to conduct astronomical research, education, and outreach. The International Virtual Observatory Alliance (IVOA) is a global collaboration of separately funded projects to develop standards and infrastructure that enable VO applications.

1 Introduction

1.1 The Problem Setting

In early VO services, there was generally a one-to-one relationship between a data service and a data collection – a cone search service exposing the 2MASS point source catalogue, or a Simple Spectral Access service (SSA) for the spectra of the ÉLODIE spectrograph. In actual practice, several more interfaces were usually added, for instance, a browser-based interface, which nicely meshed with the CAPABILITY model of VOResource (Plante and Benson et al., 2008); but even then, one service record in the Registry covered exactly one data collection.

Consequently, early considerations to have separate registry records for data collections and access services were shelved, and no existing Registry client supports this concept, although the Registry data model still provides the necessary building blocks (in particular through RELATIONSHIP).

With the widespread adoption of standards like TAP (Dowler and Rixon et al., 2010) and ObsTAP (Louys and Bonnarel et al., 2011), more and more services expose a large number of different data collections. VizieR's TAP service[1], for instance, at the time of writing gives access to close to 30 000 tables from almost 14 000 different data collections; as these are registered as separate cone search services, their metadata actually already is available in the Registry, but no link is made to the TAP service that opens them up for complex queries.

CADC's ObsTAP table[2] contains data sets from approximately one hundred instruments. A VO user might reasonably expect that when interrogating the registry for services publishing data originating from, say, the UH8K Mosaic Camera or from SCUBA-2, they should see CADC's ObsTAP service.

The problem of the mismatch between service records and data collection records not only occurs for TAP services. For instance, an image service collecting various observations of gravitationally lensed quasars within GAVO's data centre[3] exposes data from four different observation campaigns on four different instruments with four different data collection creators. Each of these collections should be discoverable independently of each other.

The class of problems just outlined is referred to as the "data discovery" use case in the following. In short: How can the Registry support discovering the VizieR tables, the data collections that contributed to CADC's ObsTAP service or to the SIAP service that is collecting data from different observatories and groups joined only by the common topic?

1.2 Non-solutions

1.2.1 The Full-Capability Approach

An obvious solution might be to simply register each data collection as if it were a standalone service. While for the four SIAP services in the example of the Lens Archive this proliferation of services of a certain type might appear tolerable, the examples of CADC's 100 separate instrument data collections

[1] `ivo://cds.vizier/tap`
[2] `ivo://cadc.nrc.ca/tap`
[3] `ivo://org.gavo.dc/lensunion/q/im`

and VizieR's 14 000 separate TAP services, each declaring the full capability metadata, already suggest that such a design is questionable.

Much more serious is that the Registry must support the enumeration of all services of a certain type with minimal repetition (the "service enumeration" use case; note that this essentially is what the current Registry does). The most common user-facing task for which that is relevant are all-VO queries as performed by, for instance, clients like Aladin (Bonnarel and Fernique et al., 2000) or Splat (Draper, 2014). In a scenario where a service containing N resources would have N resource records with the respective capability, the clients would have to make sure they filter out duplicate access URLs on their service enumeration queries; for VizieR, they would retrieve 14 000 records to end up with one access URL. Legacy clients not written to de-duplicate access URLs would, in addition, potentially be overwhelmed by the large number of services to query or present to the user.

While more powerful Registry interrogation interfaces might mitigate this particular problem, hiding the fact that there is indeed a collective service with metadata of its own leads to various problems in the user interface even of clients aware of a de-duplication requirement. Additional use cases for unique service enumeration – e.g., validators – also require the services to exist as first-class Registry objects, not just as something implied by capabilities on other records.

1.2.2 The Split-Metadata Approach

As mentioned above, early designs of the VO Registry had envisioned separate registry records for data collections and services. Such designs would clearly solve the problem rather elegantly, as clients doing data discovery would only search for data collection records and only locate services serving those in a second step. Registry clients doing service enumeration could, analogously, directly query for service records.

While attractive for its conceptual clarity, a number of concerns caused us to abandon the split-metadata approach:

1. It is hard to migrate from the current state to the split-metadata model non-disruptively. Even if the major operators of publishing registries could be moved to update their records, old-style records would still need to be maintained for a considerable time.

2. In the still-typical case of a 1:1 relation between services and data collections, this model doubles the number of registry records, which is non-trivial given that the VO currently has more than 15 000 records, quite a few of which are maintained manually.

3. The distribution of query-relevant metadata over two separate resource records that are linked through RELATIONSHIP makes for clumsy, non-obvious query patterns in common Registry access protocols; for instance, a query for data from the CALIFA project created by J. Doe in a datalink service taking a parameter Wavelength would have to look for the project and creator names in the data record, whereas information on capability type and parameter semantics would have to be searched in the service record (see also Demleitner and Harrison et al. (2015), where Fig. 4 illustrates the split between resource-bound, capability-bound, and ambiguous

parts of VOResource metadata). Translating this to RegTAP, great care must be exercised when writing the JOIN expressions to ensure the semantics is as intended. In addition, getting the JOINs wrong will not result in error messages, just in result sets not matching the intended constraints.

2 A Proposal for Discovering Data Collections

2.1 Auxiliary Capabilities

This section contains the main normative content endorsed: The form of the standard identifiers for the auxiliary capabilities (sect. 2.1.2) and the requirement for an IsServedBy relationship to the main service (sect. 2.1.3).

2.1.1 Background

There is no obvious way to avoid the problems of the split-metadata approach without a redesign of major parts of the Registry ecosystem. The main deficiency of the full-capability approach – very limited support for service enumeration –, however, can be repaired relatively straightforwardly by essentially marking capabilities as “main” or “auxiliary”, where for service enumeration clients would use the main capabilities, whereas data discovery would look at metadata for services having either one of main or auxiliary capabilities.

Marking resources as primary and auxiliary can be easily effected using the *vr:Capability* type’s *standardID* attribute. Consider the following snippet from an XML-serialised registry record:

```
<capability standardID="ivo://ivoa.net/std/SIA"
    xsi:type="sia:SimpleImageAccess">
  <interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base">http://example.com/q/im/siap.xml?</accessURL>
    <queryType>GET</queryType>
    <resultType>application/x-votable+xml</resultType>
    <param std="true">
      <name>POS</name>
        [...]
    </param>
      [...]
  </interface>
  <imageServiceType>Pointed</imageServiceType>
  <maxRecords>1000000</maxRecords>
   [...]
</capability>
<capability standardID="ivo://ivoa.net/std/VOSI#availability">
  <interface xsi:type="vs:ParamHTTP">
    <accessURL use="full">http://example.com/q/im/availability</accessURL>
  </interface>
</capability>
```

Shown here are two capabilities; based on their *standardID*s, clients can work out that the first one is a service speaking the simple image access protocol, whereas the second provides an endpoint for VOSI’s availability information. The first capability, additionally, overrides the type of the capability element to give extra metadata. In the case of *sia:SimpleImageAccess*, that is information like the service type, certain resource limits, a test query, etc.

The second capability does not override the type, which means that essentially the capability just gives zero or more interfaces, in this case one of the type *vs:ParamHTTP* (a type suitable for all endpoints for current VO protocols).

2.1.2 StandardIDs for Auxiliary Capabilities

This note proposes to mark the auxiliary capabilities by a special value in the *standardID* attribute of a *capability* element. This will in general be of type *vr:Capability* (i.e., not contain extra metadata and hence not override *xsi:type*), but that is not important to this proposal; later standards might furnish such auxiliary capabilities with additional metadata without impact on the functioning of the mechanism proposed here.

New-style standard identifiers as defined in version 2 of the Identifiers Standard (Demleitner and Plante et al., 2015) should have the form

`ivo://.../stdname#tag-version`;

for a synchronous SIAv2 service, this would be

`ivo://ivoa.net/std/sia#query-2.0`.

For such new-style standard ids, the ids of the auxiliary capabilities are simply generated by inserting `aux-` in front of the version number, for instance

`ivo://ivoa.net/std/sia#query-aux-2.0`.

This particular form for the protocol identifier is chosen to work well with the intended way to discover services with such new-style standard identifiers. As discussed in sect. 4.2 of the Identifiers standard, services supporting the major version n of the protocol *std* would query for, using SQL patterns,

`ivo://ivoa.net/std/`*stdname#tag-n.*%,

where the wildcard allows the simultaneous discovery of all minor versions. This is the intended behaviour, since by IVOA versioning policies a client for version n should be able to operate all of n.0, n.1, and so on.

If auxiliary ids were built by appending the `-aux`, this pattern would match auxiliary capabilities as well, thus making the enumeration use case extremely cumbersome to cover. Prepending the `aux-`, on the other hand, would require embedded wildcards in the pattern of the data discovery use case – where both primary and auxiliary capabilities should be investigated while protocol versions are typically a secondary concern –, which at the very least is an efficiency concern. See sect. 2.2 for the recommended discovery patterns.

Data collections not otherwise already registered that are part of another service (that already is in the Registry) would be registered as a *vs:CatalogService*, where record authors are strongly encouraged to provide a *tableset* element. After the next revision of VODataService, *vs:DataCollection*-typed records will admit capabilities, too, at which point they would be preferred for this purpose.

In the special case of TAP, we recommend that services that have auxiliary resources should not repeate the respective metadata in the main TAP record;

apart from the unnecessary information duplication (that then will have to be filtered by clients), such registry records can grow to several megabytes and more. Also, each time any piece of this large body of metadata changes, the entire information would have to be re-retrieved and reprocessed by searchable registries. On the other hand, having table-sets in the data collection records with the auxiliary capabilities allows targeted and efficient updates through the standard OAI-PMH protocol. Obviously, this consideration does not necessarily apply to TAP services only serving one or a few tables; for them, having the tableset in the TAP record remains a viable option.

2.1.3 Relationship to the Main Service

Prototyping has shown that for common user interfaces, a reference to the main service is highly desirable in addition to the access URL as given in the auxiliary capability's interface. Ideally, the relationship between the auxiliary and the main capability would be expressed as a relationship between capabilities. The current VOResource model, however, does not allow declaring such a relationship. It does allow, however, relationships between full resources. While in the future it might be desirable to allow relationships between capabilities (see below), we believe for the moment service-to-service relationships are sufficient for the purpose of linking auxiliary and main capabilities.

Therefore, a record declaring support for an auxiliary capability MUST declare an *IsServedBy* relationship to the main service's resource record, and the respective ***relatedResource*** element must give the main service's IVOID. In legacy VOResource 1.0 records, the RELATIONSHIPTYPE must be *served-by*, and clients must accept both terms.

The reverse relationship, the *IsServiceFor* (VOResource 1.0: *service-for*) relationship from the main service to the data collection records, seems less useful. Although on theoretical grounds, maintaining a symmetry here would seem desirable, the price of declaring thousands of little-used relationships for big services would appear too significant a liability given the uncertain benefit.

Note that records may have multiple auxiliary capabilities[4], and therefore not every *IsServedBy* record declared by a resource necessarily corresponds to the main service for a given auxiliary capability. We believe that in most relevant use cases, the clients will know about the IVOIDs of possible main services, such that they can easily filter out spurious main services.

For clients that do not have an enumeration of the main services for a given standard id, the main capability (or capabilities) can be found through registry queries based on the data collection's IVOID and the main capability's standard id (which is known to the client as it knows what protocol it wants to use). In RegTAP, the query would look like this:

```
SELECT c.ivoid, cap_index
FROM rr.relationship AS r
        JOIN rr.capability AS c
        ON (r.related_id=c.ivoid)
WHERE
        r.ivoid='ivo://org.gavo.dc/apo/res/apo/frames'
```

[4]An example suitable for testing client behaviour is `ivo://org.gavo.dc/apo/res/apo/frames` with auxiliary capabilities for both SIAP and TAP

```
AND standard_id='ivo://ivoa.net/std/tap'
AND relationship_type IN ('served-by', 'IsServedBy')
```

2.1.4 Passing Extra Information in the Access URL

It is not required that the access URL(s) of the auxiliary capability's interface(s) are identical to those of the main capability. Clients should therefore in general use the access URL discovered in the data discovery query rather than the access URL(s) given in the main capability.

This is intended to allow passing extra constraints to restrict service results to the discovered data collection when the protocol allows passing suitable parameters in the URL. A prototypical example is SIAv2 with its COLLECTION parameter. For instance, a service `sia2-site` could publish data from the *northerntel* and *southerntel* data collections. To ensure that from the auxiliary SIAv2 capabilities only data from the respective facility is returned, the COLLECTION constraint can be hardcoded into the access URLs. In sketch form, the three records involved might contain:

```
<Record> <!-- main SIAv2 service -->
  <identifier>ivo://example/sia2-site</identifier>
  <capability standardID="ivo://ivoa.net/std/SIA#query-2.0">
    <accessURL>http://example.com/sia.xml?</accessURL>
  </capability>
<Record>

<Record> <!-- data collection/service for northerntel -->
  <identifier>ivo://example/coll/northerntel</identifier>
  <capability standardID="ivo://ivoa.net/std/SIA#query-aux-2.0">
    <accessURL>
      http://example.com/sia.xml?COLLECTION=northerntel&amp;
    </accessURL>
  </capability>
</Record>

 <Record> <!-- data collection/service for southerntel -->
  <identifier>ivo://example/coll/southerntel</identifier>
  <capability standardID="ivo://ivoa.net/std/SIA#query-aux-2.0">
    <accessURL>
      http://example.com/sia.xml?COLLECTION=southerntel&amp;
    </accessURL>
  </capability>
</Record>
```

This technique is obviously not applicable for all protocols. In TAP, for instance, there is no way to pass table names in the access URLs. It is expected that extra metadata required in such cases is transmitted in other ways. For TAP, the extra metadata is the schema(s) and table(s) for the data collection discovered, which the discovering client can obtain from the VODataService table metadata. Other protocols may have to devise other means, possibly even involving special registry extensions for auxiliary capabilities.

2.2 Implications for Registry Clients

Under the new scheme, Registry client implementers will have to analyse whether their use of the Registry data is service enumeration ("all services of type X") or data discovery ("services for data with properties Y"). Depending on this, the constraints on the capabilities will be, in RegTAP terms,

service enumeration

To enumerate services implementing version 1 of `cap` defined by a hypothetical `example` standard, the constraint would look like

NATURAL JOIN rr.capability **WHERE**
standard_id **LIKE** 'ivo://ivoa.net/std/example#cap−1%'

This choice of pattern will not match auxiliary capabilities like `...#cap-aux-1.0`; it will, however, match `...#cap-1.0`, `...#cap-1.1`, etc.

Given the use cases for service enumeration it seems unlikely that queries of this type not constraining the version will be useful. If ever necessary, we suggest the pattern `cap-_._`, which will work at least until an IVOA protocol reaches version 10; we consider it unlikely that this will happen while current Registry infrastructure remains relevant.

data discovery

To discover data collections accessible through any version of `cap` defined by a hypothetical `example` standard, the constraint would look like

NATURAL JOIN rr.capability **WHERE**
standard_id **LIKE** 'ivo://ivoa.net/std/example#cap−%'

This pattern matches everything that the pattern in the enumeration case matches; that is intended, as it is still perfectly legal to maintain the entire metadata in a single resource record. It will, however, also match auxiliary capability identifiers like `...#cap-aux-1.0`. The downside is that this does not constrain the version of the capability.

If the query should additionally constrain the protocol version on the capability, the standard id pattern could look like

`ivo://ivoa.net/std/example#cap-%1.%`.

2.3 Implications for Registry Operators

To operators of searchable registries, the current proposal is transparent; no changes to harvesters, ingestors, or the data organisation should be necessary, except perhaps enabling indices on the capabilities' standard ids that are useful for queries involving SQL patterns.

Operators of publishing registries, on the other hand, will have to change or create new records if they publish multi-collection services. Where metadata of the data collections themselves already has registry records, it suffices to add two basically constant XML elements to the records. The typical case are services that already offer the majority of their holdings through SCS, with records properly declaring their tablesets.

To declare the availability of such a piece of data through TAP 1.0 in this case, each cone search record would receive an extra capability

```
<capability standardID=
    "ivo://ivoa.net/std/TAP#aux">
  <interface xsi:type="vs:ParamHTTP" role="std">
    <accessURL use="base">(service access URL)<accessURL>
  </interface>
</capability>
```

The necessary relationship declaration then is

```
<relationship>
  <relationshipType>IsServedBy</relationshipType>
  <relatedResource ivo-id="(main service IVOID)"
    >(terse name of the main service)</relatedResource>
</relationship>
```

(replace *IsServedBy* with *served-by* for VOResource 1.0). In case the resource already declares another *IsServedBy* relationship, only the additional *relatedResource* element has to be added to the *relationship* element.

Operators who have not yet registered their data collections will have to create new records including the usual VOResource metadata. The two fragment templates above apply to these, too. See appendix A for examples of such records.

2.4 Potential Issues

Scalability The SQL patterns given in the sample discovery queries given in sect. 2.2 are index-friendly as they have long constant prefixes. Sequential scans of large parts of the capability table might, however, become necessary when many auxiliary services in multiple major versions are present. However, in such cases, search terms for the data collections themselves (e.g., key words, author names, etc.) would typically provide sufficient constraints to enable efficient query plans. In the remaining cases, it does not appear likely that tables of VO capabilities will be too large for the occasional sequential scan any time soon.

Data model discovery When a TAP service supports data models like Obscore or the Relational Registry, it declares so by adding a *dataModel* element in a *tr:TAPCapability*-typed capability element. The auxiliary capabilities as proposed by this note do not have the type *tr:TAPCapability* and thus will not expose the extra TAP metadata. In consequence, a query of the type “Give me all ObsTAP services with data from Instrument X” will not work as expected. This deficiency could be alleviated by recommending or requiring the use of the full, typed capabilities with an auxiliary standard id for such records and keep using *capability/dataModel* for data model discovery. However, we believe the anomaly actually is the result of a modelling error. Adherence to a data model is not a property of a service, which potentially contains many data collections conforming to different data models. The early examples (ObsTAP and RegTAP) have suggested the contrary only because they described singletons. In reality, data model adherence need to be declared where a data model is realised, i.e., at the level of table or schema. We defer the details of discovering such data models to the respective specifications, mentioning EPN-TAP (?) as an example how it can be effected.

3 Considerations for the Transition Phase

Most Registry clients reviewed in Demleitner and Harrison et al. (2015) essentially do service enumeration and are hence unaffected by this proposal. Their authors might consider adding data discovery capabilities to them; an attractive alternative would be to simply add support for the SAMP MTypes allowing exchange of resource records and leave data discovery to external components like WIRR[5].

This proposal builds on the form of standard identifiers specified by the Identifiers 2 standard made a recommendation in May 2016. The transition to identifiers formed in this way is going to happen as the respective standards are reviewed.

In order to offer a solution for legacy standards and standard identifiers already in use in the VO, we propose to form auxiliary identifiers for legacy standard ids by just appending an `aux` fragment identifier. With the endorsement of this note, the respective StandardsRegExt records are amended with the necessary keys; the description of the keys will be, modulo editorial changes, "An auxiliary capability for (protocol label), i.e., a collective service making available, among others, data from this resource. See (link to the endorsed note) for details.".

The following keys are defined by this document:

- `ivo://ivoa.net/std/conesearch#aux` (SCS 1.03)
- `ivo://ivoa.net/std/sia#aux` (SIAP 1.0)
- `ivo://ivoa.net/std/sia#query-aux-2.0` (SIAP 2.0)
- `ivo://ivoa.net/std/ssa#aux` (SSA 1.*)
- `ivo://ivoa.net/std/tap#aux` (TAP 1.0)

The query patterns from sect. 2.2 still apply for those.

Future DAL protocols, as well as updates to existing ones, should define their own identifiers for their auxiliary capabilities.

A Examples

A.1 A TAP-accessible Data Collection

A resource record for a TAP-accessible data collection consisting of two tables looks like this, using the transitional identifier for TAP 1.0 auxiliary services (look for the capability element with a *`standardID`* attribute containing `#aux` to see the mechanisms proposed here in context):

```
<ri:Resource
  created="2011-05-06T11:10:00Z"
  status="active"
  updated="2015-01-16T11:43:32"
  xsi:type="vs:CatalogService"
  xmlns:ri="http://www.ivoa.net/xml/RegistryInterface/v1.0"
```

[5] http://dc.g-vo.org/WIRR

```
 xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
 xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <title>ARIGFH object catalogue</title>
 <identifier>ivo://org.gavo.dc/arigfh/q/gfhtables</identifier>
 <curation>
   <publisher>The GAVO DC team</publisher>
   <creator>
     <name>Schwan, H.; Demleitner, M.; Wielen, R.; et al</name>
   </creator>
   <date role="updated">2015-01-16T11:43:32</date>
   <contact>
     <name>Markus Demleitner</name>
     <address>M&#xF6;nchhofstrasse 12-14, D-69120 Heidelberg</address>
     <email>msdemlei@ari.uni-heidelberg.de</email>
     <telephone>++49 6221 54 1837</telephone>
   </contact>
 </curation>
 <content>
   <subject>Stars: Positions</subject>
   <subject>History and philosophy of astronomy</subject>
   <subject>Catalogues</subject>
   <subject>Astrometry</subject>
   <description>The "Geschichte des Fixsternhimmels" is an attempt to
    collect  all  [...]
   </description>
   <source format="bibcode">1989AGAb...2...33W</source>
   <referenceURL>http://dc.g-vo.org/browse/arigfh/q</referenceURL>
   <relationship>
     <relationshipType>IsServedBy</relationshipType>
     <relatedResource ivo-id="ivo://org.gavo.dc/tap"
       >GAVO TAP service</relatedResource>
   </relationship>
 </content>
 <capability standardID="ivo://ivoa.net/std/TAP#aux">
   <interface role="std" xsi:type="vs:ParamHTTP">
     <accessURL use="base"
       >http://dc.g-vo.org/tap</accessURL>
   </interface>
</capability>
 <coverage>
   <waveband>Optical</waveband>
 </coverage>
 <tableset>
   <schema>
     <name>arigfh</name>
     <title>ARIGFH object catalog</title>
     <description>"Geschichte des Fixsternhimmels" is an attempt to
      [...]
     </description>
     <table>
       <name>arigfh.nid</name>
       <description> The stars from the gfh table that could not be
         matched with objects in the master catalog.</description>
       <column>
         <name>catid</name>
         <description>Catalog identifier as t( teleki  no)p(part)(version)
         </description>
         <ucd>meta.ref</ucd>
         <dataType arraysize="*" xsi:type="vs:VOTableType">char</dataType>
         <flag>nullable</flag>
       </column>
      <!-- more columns elided for brevity -->
    </table>
    <table>
      <name>arigfh.id</name>
      <description>The stars from the gfh table having counterparts in
      the master together with those counterparts
      .</description>
      <column>
        <name>masterno</name>
        <description>Identification number in the ARIGFH master catalog
        </description>
        <ucd>meta.id;meta.main</ucd>
        <dataType arraysize="1" xsi:type="vs:VOTableType">int</dataType>
      </column>
      <!-- more columns elided for brevity -->
    </table>
  </schema>
</tableset>
</ri:Resource>
</oai:OAI-PMH>
```

A.2 An Auxiliary Image Service

A data collection published through a SIAP service also publishing other data, which is also accessible through TAP looks like this (look for the capability elements with *standardID* attributes containing `#aux` to see the mechanisms proposed here in context):

```
<ri:Resource
  created="2007-06-06T12:00:00Z"
  status="active"
  updated="2015-05-27T09:13:39"
  xsi:type="vs:CatalogService"
  xmlns:ri="http://www.ivoa.net/xml/RegistryInterface/v1.0"
  xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
  xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <title>Maidanak Observatory Lens Images</title>
  <identifier
    >ivo://org.gavo.dc/maidanak/res/rawframes/rawframes</identifier>
  <curation>
    <publisher>The GAVO DC team</publisher>
    <creator>
      <name>GAVO Data Centre</name>
      <logo>http://vo.ari.uni−heidelberg.de/docs/GavoTiny.png</logo>
    </creator>
    <date role="updated">2015−05−27T09:13:39</date>
    <contact>
      <name>Markus Demleitner</name>
      <address>M&#xF6;nchhofstrasse 12−14, D−69120 Heidelberg</address>
      <email>msdemlei@ari.uni−heidelberg.de</email>
      <telephone>++49 6221 54 1837</telephone>
    </contact>
  </curation>
  <content>
    <subject>Quasars: general</subject>
    <subject>Strong gravitational lensing</subject>
    <subject>Pointed observations</subject>
    <description>Observations of (mainly) lensed quasars from
      Maidanak Observatory, Uzbekistan</description>
    <referenceURL>http://dc.g−vo.org/tableinfo/maidanak.rawframes
  </referenceURL>
  <relationship>
    <relationshipType>IsServedBy</relationshipType>
    <relatedResource ivo-id="ivo://org.gavo.dc/lensunion/q/im"
      >Lens Image Archive</relatedResource>
    <relatedResource ivo-id="ivo://org.gavo.dc/tap"
      >GAVO TAP service</relatedResource>
  </relationship>
</content>
<capability standardID="ivo://ivoa.net/std/SIA#aux">
  <interface role="std" xsi:type="vs:ParamHTTP">
    <accessURL use="base"
      >http://dc.g-vo.org/lensunion/q/im/siap.xml?</accessURL>
    <queryType>GET</queryType>
    <resultType>application/x-votable+xml</resultType>
    <param std="true">
      <name>POS</name>
      <description>ICRS Position, RA,DEC decimal degrees
        (e.g., 234.234,-32.46)</description>
      <unit>deg</unit>
      <ucd>pos.eq</ucd>
      <dataType>string</dataType>
    </param>
    <param std="true">
      <name>SIZE</name>
      <description>Size in decimal degrees (e.g., 0.2
        or 1,0.1)</description>
      <unit>deg</unit>
      <dataType>string</dataType>
    </param>
    <param std="true">
      <name>INTERSECT</name>
      <description>Relation of image and specified Region
        of Interest.</description>
      <dataType>string</dataType>
    </param>
    <param std="true">
      <name>FORMAT</name>
      <description>Requested format of the image data</description>
      <dataType>string</dataType>
    </param>
    <param std="false">
      <name>dateObs</name>
      <description>Epoch at midpoint of observation</description>
      <unit>d</unit>
      <ucd>VOX:Image_MJDateObs</ucd>
      <dataType>string</dataType>
    </param>
    <param std="false">
      <name>bandpassId</name>
      <description>Freeform name of the bandpass used</description>
      <ucd>VOX:BandPass_ID</ucd>
      <dataType>string</dataType>
    </param>
    <param std="false">
      <name>object</name>
      <description>Object being observed, Simbad-resolvable
        form</description>
      <ucd>meta.name</ucd>
      <dataType>string</dataType>
    </param>
  </interface>
  </capability>
  <capability standardID="ivo://ivoa.net/std/TAP#aux">
    <interface role="std" xsi:type="vs:ParamHTTP">
      <accessURL use="base"
        >http://dc.g-vo.org/tap</accessURL>
    </interface>
  </capability>
  <coverage>
    <waveband>Optical</waveband>
  </coverage>
  <tableset>
    <schema>
      <name>maidanak</name>
      <title>Maidanak Observatory Lens Images</title>
      <description>Observations of (mainly) lensed quasars from
      Maidanak Observatory, </description>
      <table>
        <name>maidanak.rawframes</name>
        <description>Observations of (mainly) lensed quasars from
        Maidanak Observatory, Uzbekistan</description>
        <column>
          <name>accref</name>
          <description>Access key for the data</description>
          <utype>Access.Reference</utype>
          <dataType arraysize="*" xsi:type="vs:VOTableType">char</dataType>
          <flag>nullable</flag>
        </column>
      <!-- additional columns elided for brevity -->
      </table>
    </schema>
  </tableset>
</ri:Resource>
```

B Changes from Previous Versions

B.1 Changes from PEN-2016-11-11

- Now taking a clear stance that discovery of TAP-published data models need to be discovered not via capability/dataModel.

B.2 Changes from NOTE-1.0

- Now encouraging extra parameters in access URLs where appropriate.
- No longer requiring auxiliary capabilities to be plain *vr:Capability*.
- Updates for deprecation of *served-by* in VOResource 1.1.
- Removed section on GloTS.

References

Bonnarel, F., Fernique, P., Bienaymé, O., Egret, D., Genova, F., Louys, M., Ochsenbein, F., Wenger, M. and Bartlett, J. G. (2000), 'The ALADIN interactive sky atlas. A reference tool for identification of astronomical sources',

Astronomy and Astrophysics Supplement **143**, 33–40.
http://ads.ari.uni-heidelberg.de/abs/2000A%26AS..143...33B

Bradner, S. (1997), ‘Key words for use in RFCs to indicate requirement levels’, RFC 2119.
http://www.ietf.org/rfc/rfc2119.txt

Demleitner, M., Harrison, P., Taylor, M. and Normand, J. (2015), ‘Client Interfaces to the Virtual Observatory Registry’, *ArXiv e-prints* , arXiv:1502.01186.
http://ads.ari.uni-heidelberg.de/abs/2015arXiv150201186D

Demleitner, M., Plante, R., Linde, T., Williams, R. and Noddle, K. (2015), ‘IVOA identifiers, version 2’, IVOA Recommendation.
http://www.ivoa.net/documents/IVOAIdentifiers/20160523/

Dowler, P., Rixon, G. and Tody, D. (2010), ‘Table access protocol version 1.0’, IVOA Recommendation.
http://www.ivoa.net/documents/TAP

Draper, P. W. (2014), ‘SPLAT: Spectral Analysis Tool’, Astrophysics Source Code Library, ascl:1402.007.
http://adsabs.harvard.edu/abs/2014ascl.soft02007D

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. and Skoda, P. (2011), ‘Observation data model core components and its implementation in the Table Access Protocol, version 1.0’, IVOA Recommendation.
http://www.ivoa.net/documents/ObsCore/20111028/REC-ObsCore-v1.0-20111028.pdf

Plante, R., Benson, K., Graham, M., Greene, G., Harrison, P., Lemson, G., Linde, T., Rixon, G., Stébé, A. and IVOA Registry Working Group (2008), ‘VOResource: an XML Encoding Schema for Resource Metadata Version 1.03’, IVOA Recommendation 22 February 2008, arXiv:1110.0515.
http://adsabs.harvard.edu/abs/2008ivoa.spec.0222P