Heliophysics Integrated Observatory

Project No.: 238969
Call: FP7-INFRA-2008-2

Helio Registry Service User Manual

Draft/Under Review/Released

Title:	**Helio Registry Service – User Manual**
Document No.:	HELIO_UCL_S2_014_UM_HRS
Date:	07 June 2012
Editor:	**Kevin Benson**, UCL
Contributors:	
Distribution:	Project

CAPACITIES

e-infrastructure

Revision History

Version	Date	Released by	Detail

Note: Any notes here.

1. Introduction

The International Virtual Observatory Alliance (IVOA) registry allows astronomers to search, obtain details of, and leverage any of the resources located anywhere in the IVO space, namely in any Virtual Observatory. The IVOA defines the protocols and standards whereby different registry services are able to interoperate and thereby realise this goal. IVOA registry defines interfaces on how to query and share resources. Software is written to conform to standard interfaces in order to assist scientific utilities to access particular resource. A resource in this context is represented in XML form and is stored in the registry. A resource may describe anything about an observatory, particular instrument, another registry, and services such as catalogue or table type services, cone searches. Extensions can be made if necessary and this functionality is made available for Helio.

1.1. *Suggested Reading*

To build the service	
Helio Interface Specification	Service Interface Specification.docx
Helio API	Helio API
Java (compile service)	http://www.oracle.com/technetwork/java/javase/overview/index.html
Tomcat (web container to host the service)	http://tomcat.apache.org/
Maven (build system)	http://maven.apache.org/ (or use a plug-in to your development environment)
To extend the service	
SOAP + WSDL (web service definition)	http://www.w3.org/TR/wsdl (or use a plug-in to your development environment)
ILS Database Design	Helio_ICS_ILSDBStructure.doc

1.2. *Resource Definition*

Resources conform to a standard schema and every XML request is validated to the schema before it can be submitted to the registry for querying. More information on IVOA schemas can be found here: http://www.ivoa.net/xml/index.html

1.3. *VOResource and Dublin Core*

XML resources derive from a common top layer schema titled 'VOResource'. The VOResource may also be referred to as 'Core' or 'Dublin Core' as it contains the complete set of the necessary core data. More information on VOResource documentation can be found here:

1.4. *Identifier*

Every resource in the registry must have an identifier (similar a primary key), which is URI based. A sample: ivo://helio-vo.eu/ils/table

Identifier must be in the following form: ivo://{authorityid}/{resourcekey}

Registry manages authoritIDs. Any other registry cannot duplicate an authorityID, it is owned by one registry only. For the purposes of Helio only one authority id helio-vo.eu is managed at present. ResourceKey is a localised name and is unique in respect of the authorityID.

Though an identifier can be of any form, it is widely accepted that authorityId is a domain name or a subsection of an institute, such as mssl.ucl.ac.uk or climatephysics.mssl.ucl.ac.uk. Currently the assumption has been made that Helio only needs one main registry and will use the authority ID helio-vo.eu. A ResourceKey is typically a name with reference to the registered resource.

More information about registry identifiers can be obtained below
http://www.ivoa.net/Documents/REC/Identifiers/Identifiers-20070302.html

1.5. *VOSI*

The Support interface is required by all IVOA compliant services and defines common interfaces for its services. The registry uses common support interfaces to help populate resources in the registry.
Capability - All services define capability metadata, which comprises of XML formatted metadata that describes a particular capability and location of this particular service. The capability also describes what standards this service conforms to. Certain capabilities will be to other VOSI. Registry uses this VOSI location of the capability metadata to property fill out the resource in the registry. If other VOSI locations are present such as Table and Application metadata it additionally harvests that data.
Table Metadata - Another VOSI interface in XML form to describe table metadata for Catalogue services.
Application Metadata - Not part of VOSI, an extension created to have a piece of XML VOSI for application description.
Availability - Not used by the registry, but is provided as a Support interface to make retrieve information of uptime and other availability information concerning the service.

2. Web Administration

The End User does not have the capability to access registry via this Web interface. Only Scientist and other Technical users of Helio can use the registry to add or update resources in the Helio registry. End Users use other client programs such as the Astrogrid VODesktop to query on the resources located inside the registry.

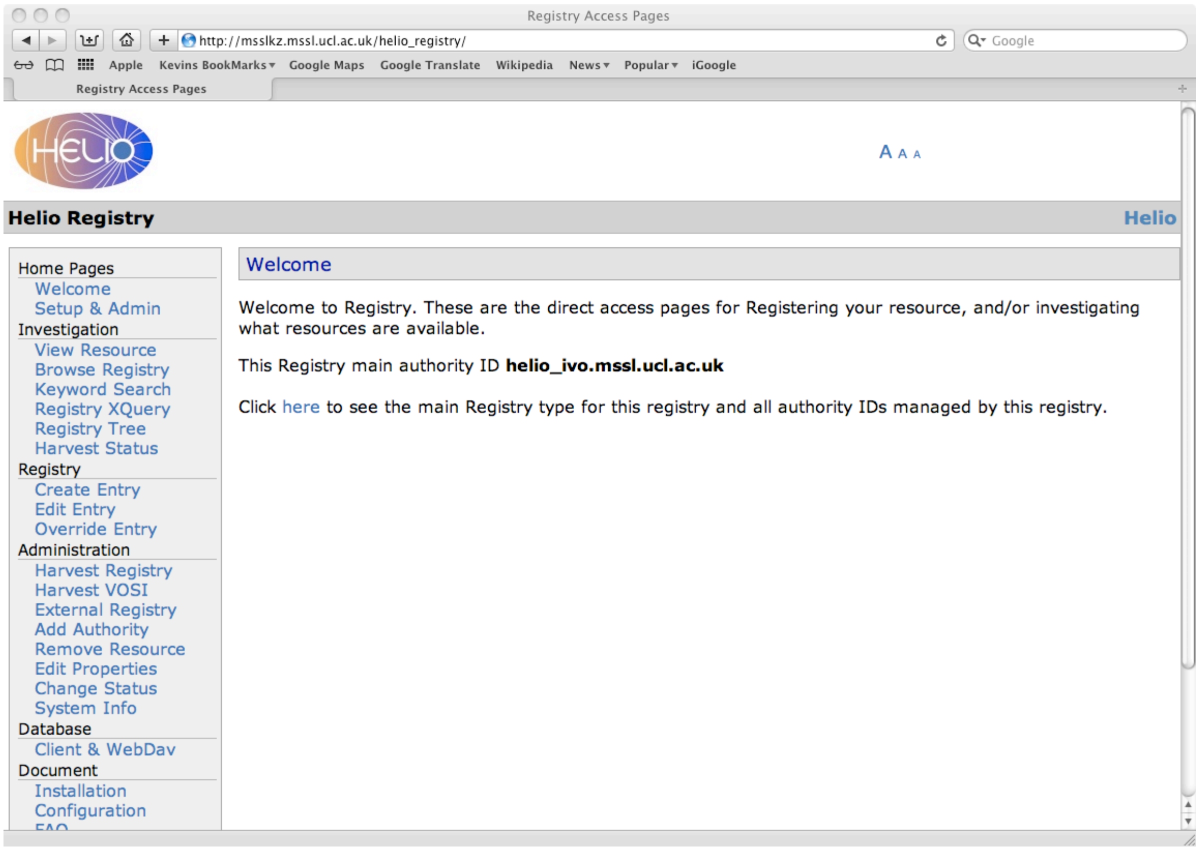
2.1. *Location*

Production registry: http://msslkz.mssl.ucl.ac.uk/helio_registry/

2.2. Welcome Page

The First page is simply a 'Welcome' page to provide access to the capabilities of the registry in the menu items. The Welcome page displays the AuthorityID setup by this registry.

Within the scope of this User Guide only the Querying/Creating/Updating resources are considered (see Creating a resource). The other administration parts of the registry are discussed in the Administrator section of the registry.



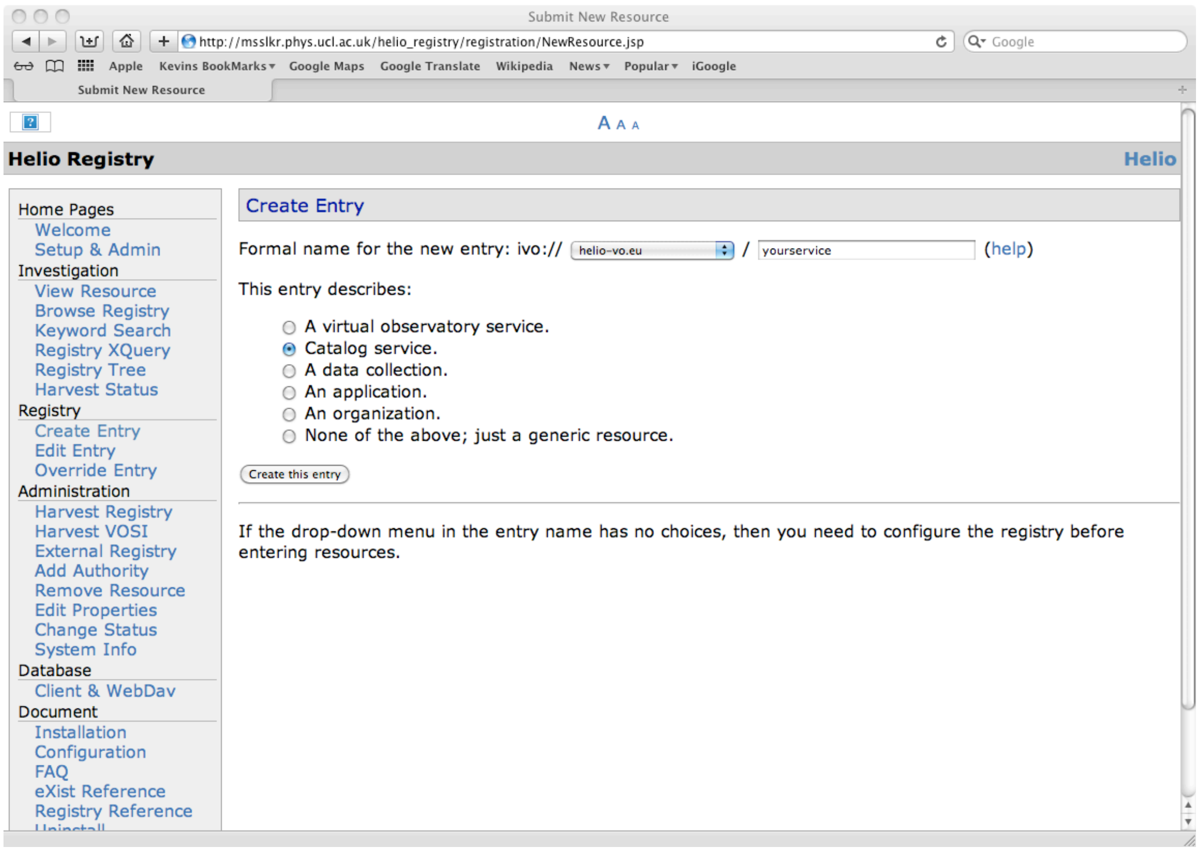
Welcome to Registry. These are the direct access pages for Registering your resource, and/or investigating what resources are available.

This Registry main authority ID **helio_ivo.mssl.ucl.ac.uk**

Click here to see the main Registry type for this registry and all authority IDs managed by this registry.

2.3. Creating a Resource

Choosing 'Create Entry' will begin creating a new resource, you must choose a unique identifier for this registry and the 'type' of resource. (See Creating a resource)



Create Entry

Formal name for the new entry: ivo:// helio-vo.eu / yourservice (help)

This entry describes:

- A virtual observatory service.
- [✓ Catalog service.]
- A data collection.
- An application.
- An organization.
- None of the above; just a generic resource.

If the drop-down menu in the entry name has no choices, then you need to configure the registry before entering resources.

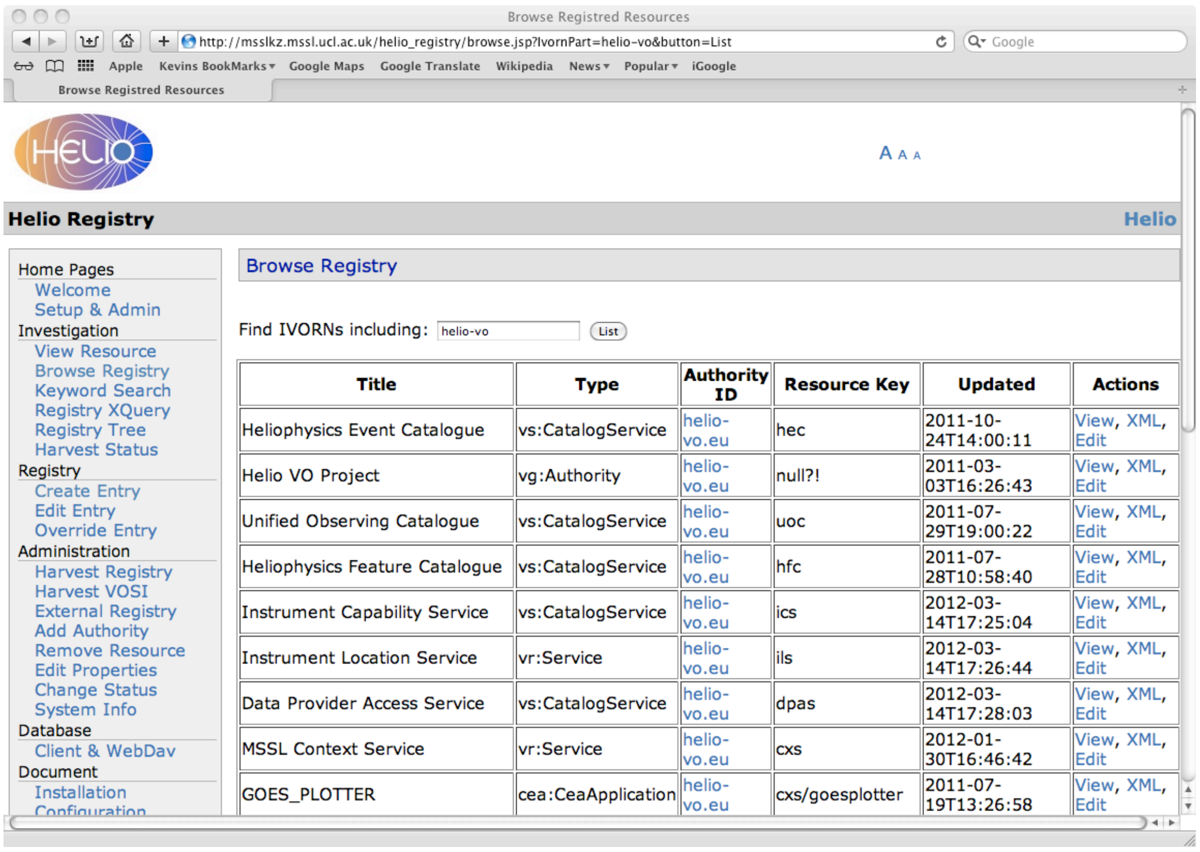
2.4. *Core Data*

The Core Data as shown in Figure 3 must be filled out prior to creating a resource. It is the Core part of the resource XML, which is entered into the registry. A 'help' link is provided next to each field to help enter the data. It is desirable to open the help link as a new tab or window.

<<INSERT IMAGE>>

2.5. *Browsing the Registry*

Several menu options exist to query or investigate the registry, however the most commonly used option is 'Browse'. When initially clicked, all the resources in the registry are displayed. It is possible to filter by identifier if needed. Only selected information is shown about the resource on the Browse screen including title, type of resource, and identifier (see Browse). It may be necessary to select to perform a particular action or task on the resource such as updating, viewing, or see the raw XML.

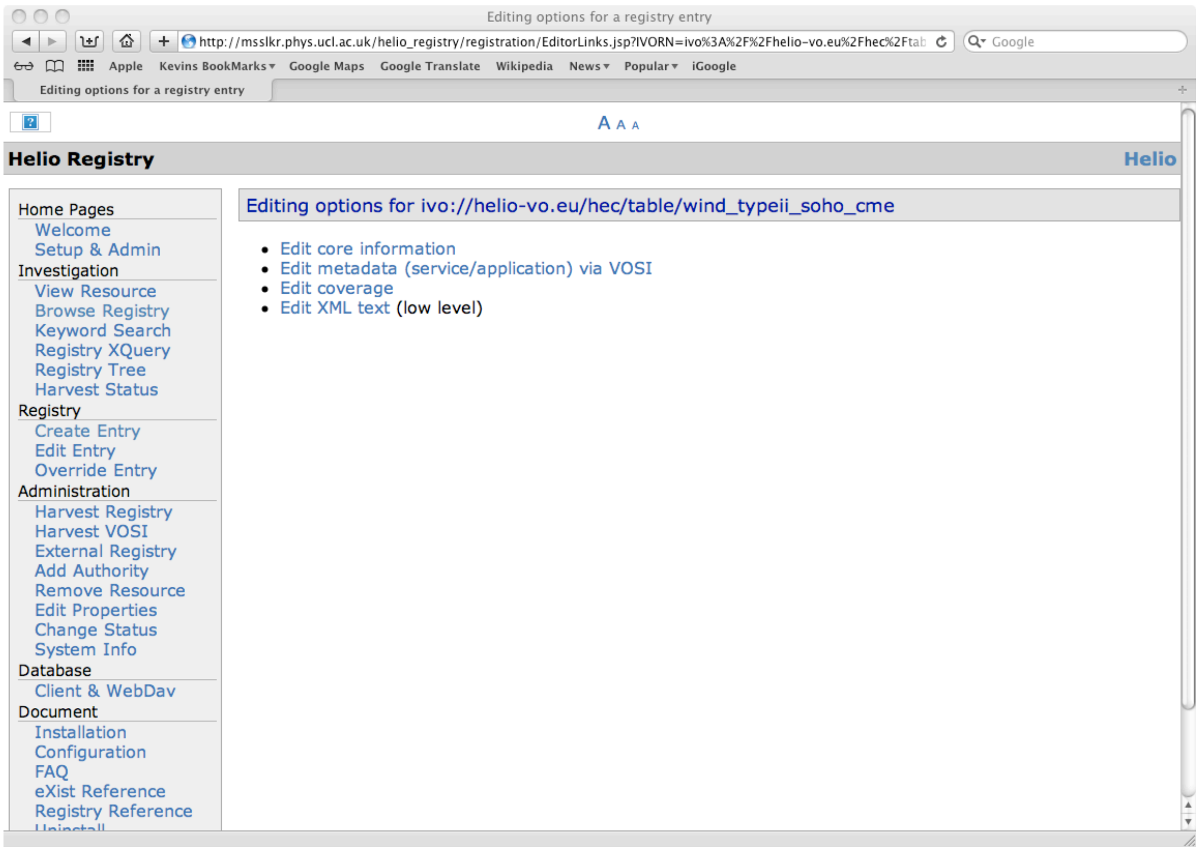


Find IVORNs including: helio-vo

Title	Type	Authority ID	Resource Key	Updated	Actions
Heliophysics Event Catalogue	vs:CatalogService	helio-vo.eu	hec	2011-10-24T14:00:11	View, XML, Edit
Helio VO Project	vg:Authority	helio-vo.eu	null?!	2011-03-03T16:26:43	View, XML, Edit
Unified Observing Catalogue	vs:CatalogService	helio-vo.eu	uoc	2011-07-29T19:00:22	View, XML, Edit
Heliophysics Feature Catalogue	vs:CatalogService	helio-vo.eu	hfc	2011-07-28T10:58:40	View, XML, Edit
Instrument Capability Service	vs:CatalogService	helio-vo.eu	ics	2012-03-14T17:25:04	View, XML, Edit
Instrument Location Service	vr:Service	helio-vo.eu	ils	2012-03-14T17:26:44	View, XML, Edit
Data Provider Access Service	vs:CatalogService	helio-vo.eu	dpas	2012-03-14T17:28:03	View, XML, Edit
MSSL Context Service	vr:Service	helio-vo.eu	cxs	2012-01-30T16:46:42	View, XML, Edit
GOES_PLOTTER	cea:CeaApplication	helio-vo.eu	cxs/goesplotter	2011-07-19T13:26:58	View, XML, Edit

2.6. *Update Resource*

On the Browse registry page 'Edit' link in the Query pages (shown in Browse) is available to fulfil the options of resource detail update. A choice of Updating the Core Information, Updating other information via the VOSI interface, Updating Coverage information, and finally an Update Screen that gives access to the Raw XML are available.



Editing options for ivo://helio-vo.eu/hec/table/wind_typeii_soho_cme

- Edit core information
- Edit metadata (service/application) via VOSI
- Edit coverage
- Edit XML text (low level)

2.7. *Edit Core*

Similar page as 'Creating a resource' with the html form fields populated with what is contained in the registry is shown in Updating Core metadata.

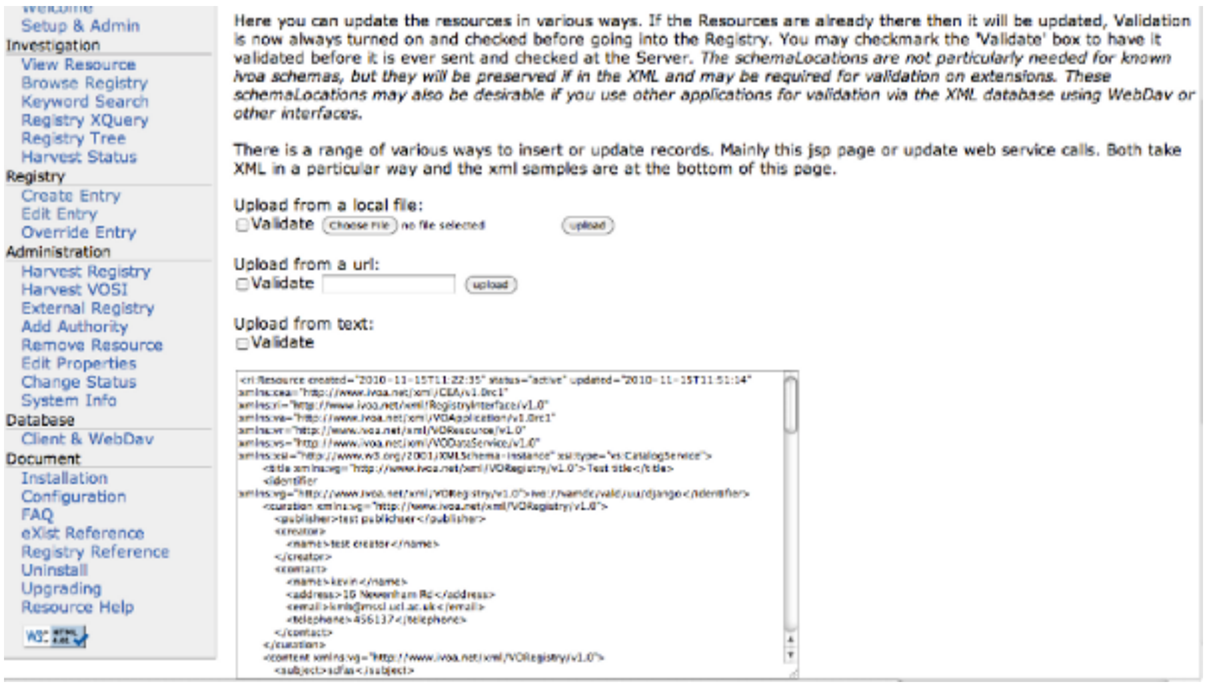
<<INSERT Image>>

2.8. *Updating VOSI*

After Creating or during Updating a service an option is be given to update the service with a VOSI capabilities location. This reads the XML metadata and populates the resource accordingly (shown in Populating resource based on VOSI/Capabilities). This Form expects a VOSI URL to point to 'Capabilities' URL that describes locations and standards of this service and VOSI services (see Populating resource based on VOSI/Capabilities).

2.9. *Editing an Existing Resource via Raw XML*

Populating resource based on VOSI/Capabilities demonstrates the ability to upload Raw XML, local file, URL location or using an html Text box. Options shown in Using raw XML to update or create resources may be useful if an XML resource is already locally saved, one may then edit manually and directly upload a new update. This option is also useful for making quick changes. When submitted it is validated and placed into the registry.



Here you can update the resources in various ways. If the Resources are already there then it will be updated, Validation is now always turned on and checked before going into the Registry. You may checkmark the 'Validate' box to have it validated before it is ever sent and checked at the Server. *The schemaLocations are not particularly needed for known ivoa schemas, but they will be preserved if in the XML and may be required for validation on extensions. These schemaLocations may also be desirable if you use other applications for validation via the XML database using WebDav or other interfaces.*

There is a range of various ways to insert or update records. Mainly this jsp page or update web service calls. Both take XML in a particular way and the xml samples are at the bottom of this page.

Upload from a local file:
☐ Validate

Upload from a url:
☐ Validate

Upload from text:
☐ Validate

```
<ri:Resource created="2010-11-15T11:22:35" status="active" updated="2010-11-15T11:51:14"
xmlns:cea="http://www.ivoa.net/xml/CEA/v1.0rc1"
xmlns:ri="http://www.ivoa.net/xml/RegistryInterface/v1.0"
xmlns:va="http://www.ivoa.net/xml/VOApplication/v1.0rc1"
xmlns:vr="http://www.ivoa.net/xml/VOResource/v1.0"
xmlns:vs="http://www.ivoa.net/xml/VODataService/v1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="vs:CatalogService">
    <title xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0">Test title</title>
    <identifier
xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0">ivo://mssl/vald/uu/django</identifier>
    <curation xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0">
      <publisher>test publichser</publisher>
      <creator>
        <name>test creator</name>
      </creator>
      <contact>
        <name>kevin</name>
        <address>10 Newenham Rd</address>
        <email>kmb@mssl.ucl.ac.uk</email>
        <telephone>456137</telephone>
      </contact>
    </curation>
    <content xmlns:vg="http://www.ivoa.net/xml/VORegistry/v1.0">
      <subject>sdfas</subject>
```

3. Querying

Querying requires software to conform to the IVOA registry interface specification: http://www.ivoa.net/Documents/RegistryInterface/ . There are four main query methods:

- Xquery - not supported in all IVOA registries, but Helio does support Xquery. It is the most advanced way of querying on the registry. But requires knowledge of all the schema structure to construct certain XPATH nodes. Software such as Astrogrid VODesktop gives you a simple query interface and performs the more complex Xquery behind the scenes.
- ADQL - also an advanced way of querying the registry. It is an SQL form over XML. It does not have all the functionality as Xquery such as ADQL can only return the full resource from the registry, but is standard and all IVOA registries conform to ADQL. Software that wants to be certain to work for all IVOA registries tend to use this method such as TopCat.
- KeywordSearch - generic keyword search mechanism.
- GetResource - gets one particular resource entry from the registry.
  - See Querying the Registry for Helio Resources for the available libraries and help querying of the registry.

3.1. XQuery

The recommended way to look for things in the registry is to send in queries in the XQuery language. The registry responds with XML documents carrying the information matching the query.

For a given XQuery and for a given programming language, the details of the query can be encapsulated in a client library; the library phrases the query based on simple parameters to a method call. This has been done for typical Helio queries from Java, and the library is described below. Often this is all you need, but sometimes it is easier or more efficient to make the query directly from your application code.

If you do not understand the basics of XQuery you will not understand the details of this section. Either skip ahead to the descriptions of the client library or have a look at an XQuery tutorial.

This is an example of a registry XQuery. It finds the formal names of all the Helio HQI services.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
for $x in //ri:Resource
where $x/capability[@standardID='ivo://helio-vo.eu/std/helioquery']
and $x/@status='active'
return $x/identifier
```

The query could be translated as “Find all the registration documents containing a capability with the Helio HQI identifier, taking only those for active services; give me the IVORNs and throw the rest away”. The XPath construct //ri:Resource means “all the registration documents”. Because this searches for a type of element, and because types have namespaces, we have to map the namespace to a prefix (the first line) and use that prefix in specifying the type (the ri: in ri:Resource).

The registry’s response will be be a document containing identifier elements as immediate children of the document element.

Most queries will be in this general form. It is important to restrict the search to active resources because the registry contains some that are “inactive” (resting, pending refurbishment) or “deleted” (gone for good, but not actually removed from the registry database).

This is a possible rearrangement of the query above.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
return //ri:Resource[capability[@standardID='ivo://helio-vo.eu/std/helioqueryservice' and
@status='active']/identifier
```

The constraints have been moved inside the square brackets in the return clause and the where clause disappears. Both queries should raise the same results; you can use whichever form is easiest for you.

Here is a different query, searching for TAP services.

```
declare namespace ri='http://www.ivoa.net/xml/RegistryInterface/v1.0';
for $x in //ri:Resource
where $x/capability[@standardID='ivo://ivoa.net /TAP']
and $x/@status='active'
return $x
```

Here, the identifier for the capability is different - IVOA TAP (Table Access Protocol) instead of Helio Query Interface - and, more importantly, the query returns all the parts of the registration documents, not just the identifiers.

4. Querying Registry with other Software

There are, broadly, four ways to put a query to the registry from Java. In increasing order of abstraction and preference they are:

- Call the registry (SOAP) web-service directly.
- Use the AstroGrid client library.
- Use the AstroGrid Astro-Runtime API.
- Use the Helio API
-

The AstroGrid client library is worth considering. If you have a simple query (e.g. if you already know the identifier for the service of choice and just want to extract the access URL) then the library is quite good. If you have a more-general query, particularly one that will return results from more than one registration, then the library has to be forced into a non-standard configuration to work properly.

The Astro Runtime is a better abstraction for the registry and is actually intended for applications programmers (the AstroGrid client-library above is aimed at system engineers). It can return results as Java objects rather than as XML, which is sometimes easier to deal with. However, you have to write your own query text, typically in XQuery. There is a Helio API client-library (see below), which tries to abstract common queries so you don't need to write any XQuery text. This library knows about (some of) the service types important in Helio. Support for forming queries is good. Support for parsing the results is limited; you either get a DOM or simple values in strings, depending on the kind of query.