

The Astro Runtime for data access

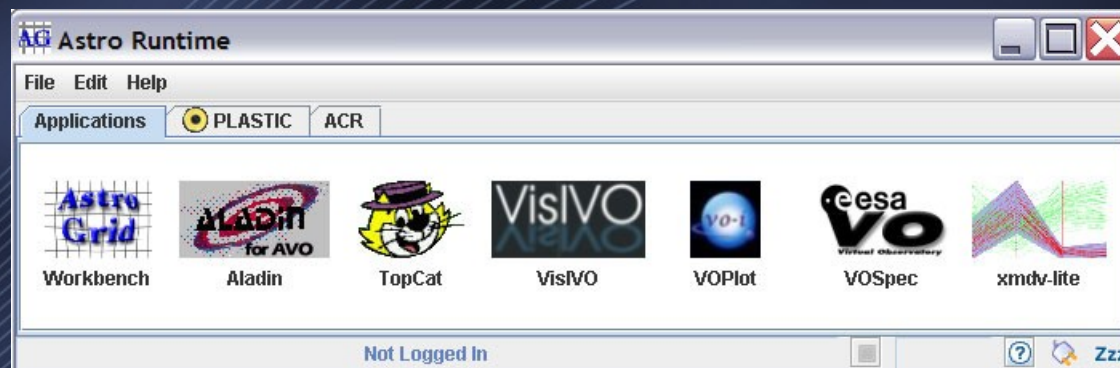
Noel Winstanley
Jodrell Bank, AstroGrid

with the part of Noel played by
John Taylor,
IfA Edinburgh/AstroGrid



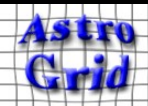
The Astro Runtime

uniform access to all VO services, from all programming languages



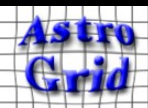
How do we access VO Services?

- Use published WSDL to generate own SOAP client, call services directly
 - need to understand how AG services interact
 - security – needs advanced SOAP handling
 - SOAP difficult or impossible from some platforms
 - Not just SOAP - many protocols to learn
- Auto-generate language-specific clients
- Call methods on the Astro Runtime
 - Clean Facade Interface to VO Services
 - Provides extra benefits
- Info <http://software.astrogrid.org/developerdocs/>



AstroRuntime Terminology

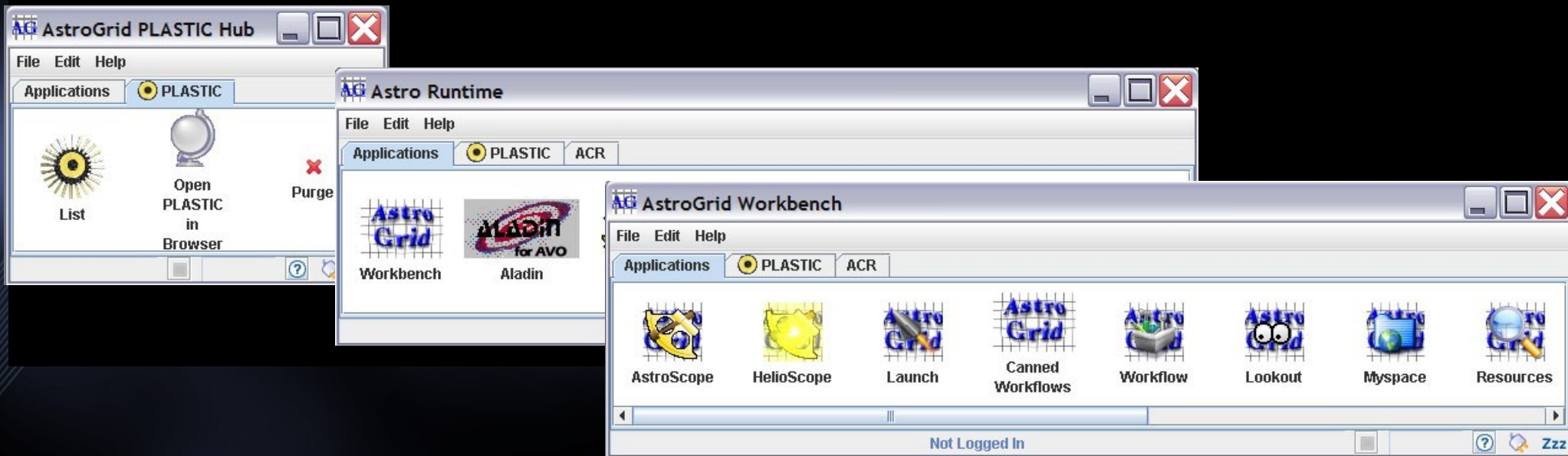
- ACR (Astro Client Runtime) is a desktop service that makes it simple for other programs to access VO services.
- ASR (Astro Service Runtime) is the server-side equivalent – same API, but no GUI components and multi-user support.
- Workbench is a suite of GUI applications built upon the ACR
- <http://software.astrogrid.org/beta/ar/>
 - Single-click launch using Java WebStart
 - choose 'Workbench Launch'
 - try it now :)



Astro Runtime variants

All variants are webstartable (except ASR) and available as executable jars and embeddable libraries

Variant	Size	Plastic Hub	Access to VO services AG,CDS,NVO,IVOA	Dialogs myspace browser...	Apps AstroScope...
Plastic Hub	4M				
ASR	13M				
ACR	20M				
Workbench	24M				

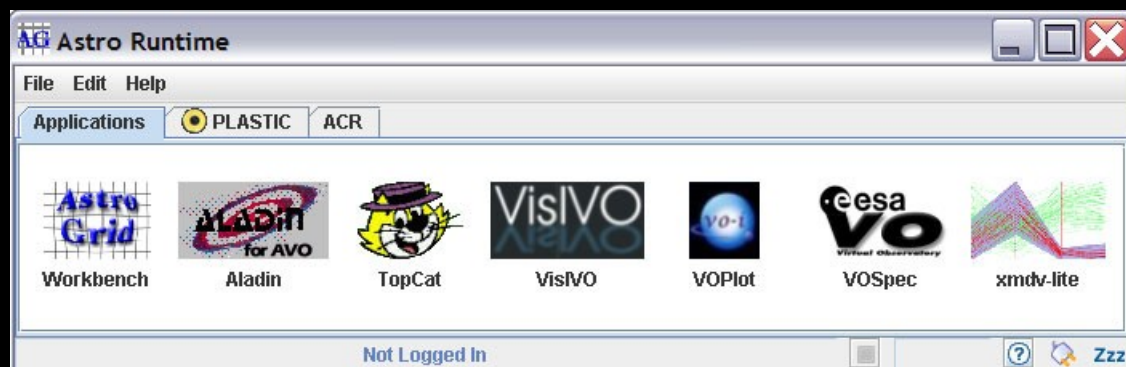


Astro Runtime variants

Only discuss these in this talk:

All variants are webstartable (except ASR) and available as executable jars and embeddable libraries

Variant	Size	Plastic Hub	Access to VO services AG,CDS,NVO,IVOA	Dialogs myspace browser...	Apps AstroScope...
Plastic Hub	4M				
ASR	13M				
ACR	20M				
Workbench	24M				



ACR – Purpose

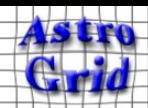
- A uniform way to access VO components..
 - remote: web services – SOAP, REST, etc
 - client side: GUI components; dialogues; helper libraries
- ... from any programming, scripting or shell language
- ... on any platform
- In this talk will concentrate on access to data:
 - Cone search
 - S*AP
 - SkyNode
 - CEA
 - Registry

ACR Design

- ACR designed to be accessible from all programming languages
- Procedural design, rather than OO (astronomer friendly)
- A service that runs on the user's desktop
 - accepts requests from other desktop applications
 - processes requests by calling webservices using the AstroGrid Java client libraries.
- Components
 - ACR provides a large set of components / services that can be called by any of the access methods
 - related components organized into modules.

What's in it for you?

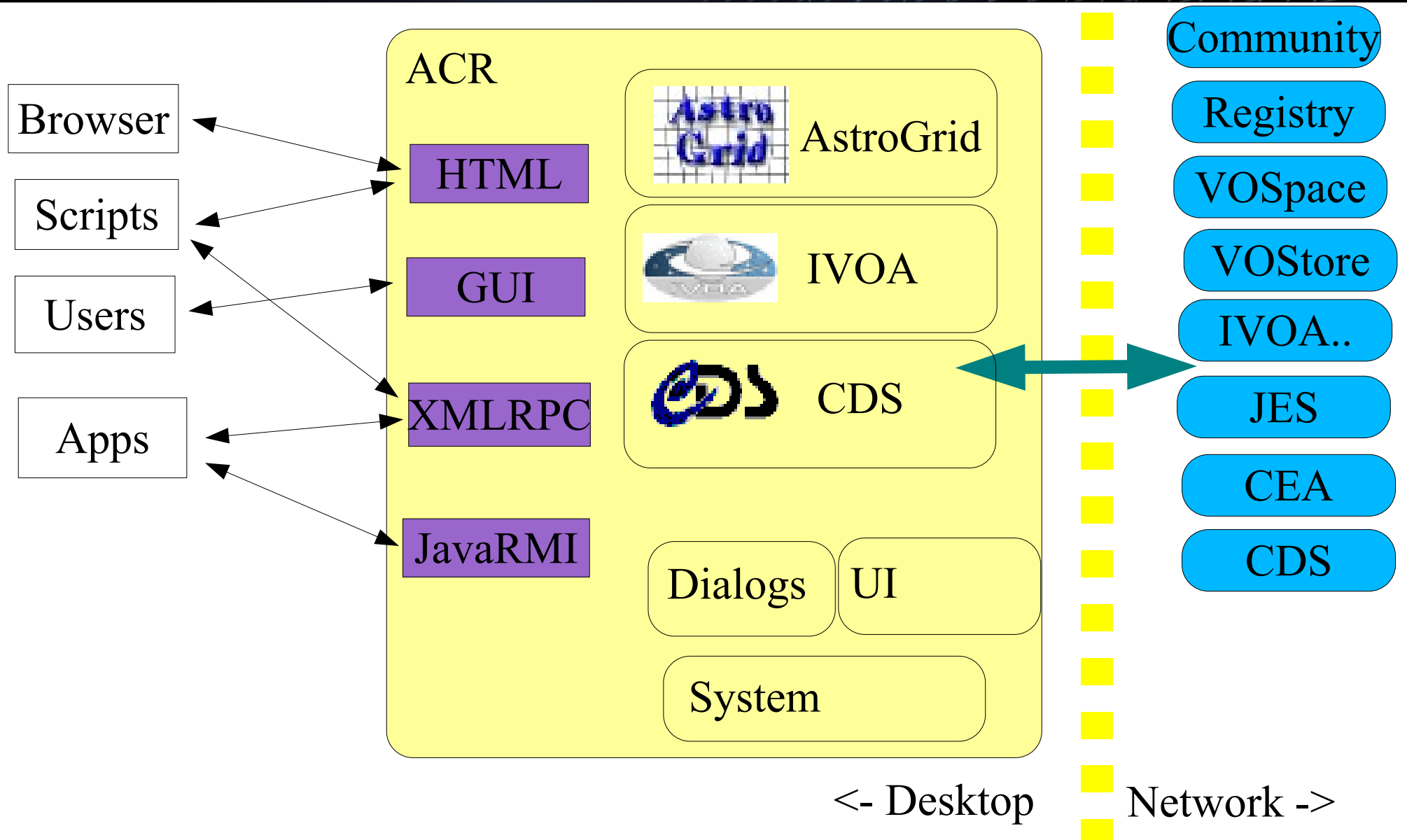
- A library of virtual-observatory functions
- A common facade for the VO
 - aim to integrate all VO standards, popular ad-hoc services, and suitable helper functions.
- uniform abstraction level and types
 - cleaner API, fewer special cases, shallow learning curve
- single configuration
 - taken care of – client programmer doesn't need to care.
- simple deployment
 - trivial to install using Java WebStart and easily embeddable
- Shared component – single signon, cached registry entries, myspace trees, insulated from change



Access Methods

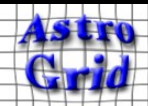
- JavaRMI (Java, Groovy, Jython)
 - JVM-only inter-process communication
 - strongly typed
 - requires a minimal set of libraries
 - allows remote event listeners to be registered
- XMLRPC (Python, Perl, C++, C# , Java)
 - Forerunner of SOAP: <http://www.xmlrpc.com/>
 - simpler types than SOAP
 - implementations for a wide range of languages
- HTTP-Get (Shell, R, IDL, Matlab)
 - rough-n-ready procedure call
 - fallback for other languages

ACR Schematic



What services can I get at?

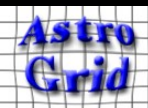
- IVOA – **SIAP**, SSAP, skyNode, adql converter, **registry**
- “IVOA” - cone search, VOSpace
- AstroGrid – CEA applications and workflows
- CDS – GLU, **sesame**, UCD, VizieR



Code demo

Use the registry to locate a SIAP service
Query SIMBAD to locate an object
Construct and execute a SIA query

....all in Python, without the use of SOAP, http or even seeing a URL



Applications using the Astro Runtime

Searches the registry
Queries SIAP services
Saves to MySpace



Launches CEA apps
on HPC resources



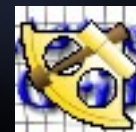
Browses
MySpace



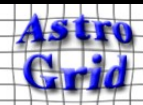
Topcat



Searches the registry
Queries SIAP services
Performs cone searches
Queries SSAP services



AstroScope



Contacts and references

Noel Winstanley Noel.Winstanley@manchester.ac.uk

John Taylor jdt@roe.ac.uk

The Astro Runtime

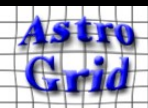
<http://software.astrogrid.org/beta/ar/>

<http://wiki.astrogrid.org/bin/view/Astrogrid/AstroClientRuntime>

API Docs

<http://software.astrogrid.org/beta/ar/xmlrpc.html>

<http://software.astrogrid.org/beta/ar/apidocs/index.html>



More references:

AstroRuntime code recipes (Java, Python, Perl, C, bash, R, Matlab...)

<http://wiki.astrogrid.org/bin/view/Astrogrid/AcrRecipes>

AstroRuntime tutorial

<http://wiki.astrogrid.org/bin/view/Astrogrid/MakingAppsVOAwareWorksheet>

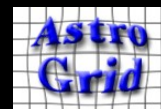
Report on use of ACR in Aladin (Boch)

<http://eurovotech.org/twiki/bin/view/VOTech/UsageOfAcrApiInAladin>

Other presentations on the AR

<http://www.ivoa.net/internal/IVOA/InterOpOct2005Applications/acr-voclient-ivoa-oct-2005.sxi>

<http://wiki.astrogrid.org/bin/view/Astrogrid/AgTechWorkshopJan06>



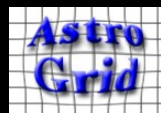
Workbench UI

The screenshot displays the AstroGrid Workbench interface with several active windows:

- Application Launcher - 6dF:** Shows a search for 'roe' with parameters like XML, Info, and Chooser. Below it, an SQL query is visible:

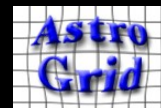

```
select b.OBJID , b.CATNAME
from DENISI as b
where
```
- Registry Browser:** Displays search results for 'subaru', including titles like 'Radio and Infrared observations of EROs (Smail+, 2002) - ERO Catalog' and identifiers like 'ivo://CDS/VizieR/JJ/ApJ/581/844/table1'. A description of the radio image of A851 is also present.
- AstroScope:** Shows configuration for 'Radial Hyperbolic' with position/object 'm32' and region '0.1'. It includes visualization controls and a 'Go to Top' button.
- VO Lookout:** A message log window showing a list of messages with columns for Subject, Date, and From. The latest message is from 'jes.galahad.star.le.ac.uk' dated '23/11/05 05:36'.

Subject	Date	From
Status Change	23/11/05 05:36	jes.galahad.star.le.ac.uk
Information	11/11/05 21:55	JES
Information	11/11/05 21:55	JES
Information	11/11/05 21:55	JES
Information	11/11/05 21:55	JES
Information	11/11/05 21:55	JES
Information	11/11/05 21:56	JES
Results	23/11/05 05:36	jes.galahad.star.le.ac.uk
- Workflow Editor:** Shows a 'Tree view' of a workflow named 'AstroGrid Redshift Maker'. It includes steps like 'Calculate redshifts from imaging data', 'Script WFS DQC query', and 'Script scriptx'.
- File Explorer:** Shows a directory structure with folders like 'intwfs' and 'boo'.
- Properties Window:** Displays details for the file 'Run-169604-CCD-2.fits', including creation and modification dates, node information, size (16394 Kb), and store location.
- Message Log:** Shows a list of files and folders, including 'Run-180480-CCD-1.fits' through 'Run-168385-CCD-4.fits' and 'wibble'.



Stop

Further information including code examples follows this slide.



Java RMI

Import ACR classes

```
import org.astrogrid.acr.Finder;
import org.astrogrid.acr.astrogrid.Registry;
import org.astrogrid.acr.builtin.ACR;
import org.astrogrid.acr.system.Configuration;

import java.net.URI;
import java.util.Iterator;
import java.util.Map;
```

Instantiate finder

Find running ACR, or execute new

Get reference to service

Alternative way to get service

Call service function

Tell program to exit

```
public class Connect {

    public static void main(String[] args) {
        try {
            Finder f = new Finder();
            ACR acr = f.find();
            // retrieve a service - by specifying the interface class
            Configuration conf =
                (Configuration)acr.getService(Configuration.class);
            // call a method on this service.
            Map l = conf.list();
            for (Iterator i = l.entrySet().iterator(); i.hasNext(); ) {
                System.out.println(i.next());
            }
            // retrieve another service from the acr - this time by name
            Registry registry = (Registry)acr.getService("astrogrid.registry");

            // use this service..
            URI u = new URI("ivo://org.astrogrid/Pegase");
            System.out.println(registry.getResourceInformation(u));
            // returns a struct of data.
            // registry.getRecord(u) returns a org.w3c.dom.Document..

            u = new URI("ivo://uk.ac.le.star/filemanager");
            System.out.println(registry.resolveIdentifier(u));
            // returns a java.net.URL

        } catch (Exception e) {
            e.printStackTrace();
        }
        //shut the app down - necessary, as won't close by itself.
        System.exit(0);
    }
}
```



Python XML-RPC

Import xmlrpc library

Read ACR configuration file

Construct xmlrpc endpoint

Create client

Get reference to service

Call service function

```
#!/usr/bin/env python
# Noel Winstanley, Astrogrid, 2005
# minimal example of connecting to acr and calling a service.
import xmlrpclib
import sys
import os

#parse the configuration file.
prefix = file(os.path.expanduser("~/astrogrid-desktop")).next().rstrip()
endpoint = prefix + "xmlrpc"
print "Endpoint to connect to is", endpoint

#connect to the acr
acr = xmlrpclib.Server(endpoint)

#get a reference to the registry service from the acr.
registry = acr.astrogrid.registry

#call a method
print registry.getResourceInformation('ivo://org.astrogrid/Pegase')
    # returns a struct of data

print registry.getRecord('ivo://org.astrogrid/Pegase')
    # return the xml of a registry entry (string)

print registry.resolveIdentifier('ivo://uk.ac.le.star/filemanager')
```

Perl XML-RPC – same pattern

Import xmlrpc library
- alternatives?

Read ACR configuration file

Construct xmlrpc endpoint

Create client

Call service function

```
#!/usr/bin/perl
#Noel Winstanley, Astrogrid, 2005
#basic perl example - incomplete.
#connects to acr using xmlrpc interface.

#xmlrpc client for perl, downloadable from cpan
use Frontier::Client;

# create the server
# don't know how to find current user's home dir,
#or how to read in files nicely - hope someone can show me this
open(CONFIG_FILE, "/home/noel/.astrogrid-desktop")
  || die("Could not open acr config - check ACR is running");
$prefix=<CONFIG_FILE>;
close(CONFIG_FILE);
chomp $prefix;
$url = $prefix . "xmlrpc";
#create xmlrpc client
$sacr = Frontier::Client->new(url => $url);

# call some methods on the acr
$record = $sacr->call('astrogrid.registry.getRecord'
                    , 'ivo://org.astrogrid/Pegase');
print $record, "\n";

$endpoint = $sacr->call('astrogrid.registry.resolveIdentifier'
                      , 'ivo://uk.ac.le.star/filemanager');
print $endpoint, "\n";
```

Shell – raw HTTP

Determine server endpoint

function name

```
# read config file to get endpoint
SERVER=`cat ~/.astrogrid-desktop`

#uses curl to do the work - consult manual for possibilities.
echo retrieve a registry record
echo `curl -d "ivorn=ivo://org.astrogrid/Pegase" -s ${SERVER}astrogrid/registry/getRecord/plain`

echo resolve an identifier
echo `curl -d "ivorn=ivo://uk.ac.le.star/filemanager" -s ${SERVER}astrogrid/registry/resolveIdentifier/plain`

echo plaintext keyword search
echo `curl -d "keywords=ROSA" -d "orValues=false" -s ${SERVER}astrogrid/registry/keywordSearchRI/plain`
```

parameters

result format

- develop this using HTML interface