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

  - Single-click launch using Java WebStart
  - choose 'Workbench Launch'
  - try it now :)

IVOA Interoperability Meeting, Victoria 2006
## Astro Runtime variants

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

<table>
<thead>
<tr>
<th>Variant</th>
<th>Size</th>
<th>Plastic Hub</th>
<th>Access to VO services</th>
<th>Dialogs</th>
<th>Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plastic Hub</td>
<td>4M</td>
<td></td>
<td>AG, CDS, NVO, IVOA</td>
<td>myspace browser...</td>
<td>AstroScope...</td>
</tr>
<tr>
<td>ASR</td>
<td>13M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACR</td>
<td>20M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workbench</td>
<td>24M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Diagrams

- [Astro Grid PLASTIC Hub](#)
- [Astro Runtime](#)
- [Astro Grid Workbench](#)
**Astro Runtime variants**

Only discuss these in this talk:

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

<table>
<thead>
<tr>
<th>Variant</th>
<th>Size</th>
<th>Plastic Hub</th>
<th>Access to VO services AG,CDS,NVO,IVOA</th>
<th>Dialogs myspace browser...</th>
<th>Apps AstroScope...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plastic Hub</td>
<td>4M</td>
<td>Green</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASR</td>
<td>13M</td>
<td>Green</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACR</td>
<td>20M</td>
<td>Green</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Workbench</td>
<td>24M</td>
<td>Green</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

![Astro Grid interface](image)
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

Browser

Scripts

Users

Apps

ACR

HTML

GUI

XMLRPC

JavaRMI

AstroGrid

IVOA

CDS

Dialogs

UI

System

<- Desktop

Network ->

Community

Registry

VOSpace

VOStore

IVOA..

JES

CEA

CDS
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

- **Topcat**
  - Browses MySpace
  - Searches the registry
  - Queries SIAP services
  - Saves to MySpace

- **AstroScope**
  - Launches CEA apps on HPC resources
  - Searches the registry
  - Queries SIAP services
  - Performs cone searches
  - Queries SSAP services

- **Astro Grid**

- **VisIVO**
Contacts and references
Noel Winstanley  Noël.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://eurovotecoh.org/twiki/bin/view/VOTech/UsageOfAcrApiInAladin

Other presentations on the AR
http://wiki.astrogrid.org/bin/view/Astrogrid/AgTechWorkshopJan06
Workbench UI
Stop

Further information including code examples follows this slide.
Java RMI

```java
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);
        }
    }
}
```
#!/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

#acr = xmlrpclib.Server(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

#!/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
$acr = Frontier::Client->new(url => $url);

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

$endpoint = $acr->call('astrogrid.registry.resolveIdentifier'
     , 'ivo://uk.ac.le.star/filemanager');
print $endpoint, "\n";
Shell – raw HTTP

Determine server endpoint

- develop this using HTML interface