Astronomical Data Analysis Software and Systems XVI **O6a.2**
ASP Conference Series, Vol. XXX, 2006
R. Shaw, F. Hill and D. Bell, eds.

Astro Runtime - An API to the Virtual Observatory

Noel Winstanley[1], John D. Taylor[2], Mark B. Taylor[3], Keith Noddle[4]
VOTech Project, http://eurovotech.org

Eduardo Gonzalez Solares
Institute of Astronomy, University of Cambridge, UK.

Johan Lindroos
CSC - Scientific Computing Ltd, Finland.

Abstract. The infrastructure of the virtual observatory (VO) is becoming complex. The technology is challenging to master and time-consuming to stay abreast of. This may delay the adoption of VO insfrastructure and services by scientists or application developers.

We present a method to programatically access the VO while remaining isolated from much of the detail through using *Astro Runtime*(AR). AR provides a facade API to the VO that can be accessed from applications or scripts written in almost any language, on any platform.

This paper describes the AR, shows example code that calls it, and lists how it is currently being used by various astronomy applications.

1. Accessing the Virtual Observatory

The infrastructure of the Virtual Observatory (VO) is becoming complex: partly through necessity, partly due to growing pains. Some of the causes of this are: the growing number of standards; interaction between standards; variation between standard versions; non-compliance of implementations; use of existing computing technologies (e.g. XML Schema, SOAP, WS-*); non-standard implementations of these technologies; and that the whole system is still developing.

This technology is challenging to master, and then time-consuming to stay abreast of. In our opinion, this may delay the adoption of VO infrastructure by scientists and application developers.

---

[1]Jodrell Bank Observatory, University of Manchester, UK.

[2]Institute for Astronomy, University of Edinburgh, UK.

[3]Department of Physics, University of Bristol, UK.

[4]Department of Physics & Astronomy, University of Leicester, UK.

Figure 1. Architecture of a system using Astro Runtime

We present a practical solution on this complexity, which has already been utilized by a number of applications. Astro Runtime (AR)[5] is a platform-independent executable that contains efficient, error-tolerant clients for VO standards and other popular services. This *middleware* hides the complexity of the emerging VO behind a simple, stable, consistent and self-contained high-level API - a *facade*.

This facade API provides functionality for implementing data analysis applications that use remote VO resources. It is suitable for use by both production applications and informal science scripts that desire easy access to the VO while still being isolated from its details and evolution. The API is procedural, accessible via a range of different access methods, making it usable from practically any programming language.

2. How Astro Runtime works

Figure 1 shows how the components of a system using AR interact. In a typical configuration, AR runs in the background on the users's desktop. Applications that wish to communicate with remote VO services invoke operations of the facade API of the AR. In the figure, Applications, Scripts & Browser enumerate the three kinds of client that might use Astro Runtime.

Clients invoke operations on AR using one of (currently) four access methods. The method selected depends on how the client is implemented: those written in Java typically use the Java RMI or Direct function call methods. Clients

[5] `http://www.astrogrid.org/desktop/astro-runtime`

written in other languages connect using XML-RPC if a library is available[6], otherwise the HTTP-based REST method. Regardless of how clients communicate with AR, identical functionality is available (modulo limitations of the access method) and the same implementation is used.

We intend AR to support and track all IVOA standards, plus other popular services. Amongst its current capabilities are: locate, query and interpret responses from standard image, catalogue and spectra DAL services; query registries to return object models of resources; access to CDS services; and load and store data from MySpace[7]. It also provides utility functions for processing common data formats, and a collection of reusable GUI dialogues (e.g registry browser, file chooser). All functionality is accessible from the API.

As a shared desktop service, AR is also a useful place to cache service responses, store system configuration and manage credentials for single sign-on. It can also be used as a hub for PLASTIC messaging (Taylor J. 2006).

3. Using the API

When a client invokes an operation, it may trigger multiple calls to one or more VO services. E.g., to query a secure catalog service requires calls to registry (to resolve service endpoints), community, authentication services, and the catalog service. If the query result is placed in MySpace, then further service calls are required to retrieve it. This complexity is hidden from the calling application.

The following 12-line Python[8] example uses AR to query a SIAP service, display the result using PLASTIC and save the images to MySpace. We hope this demonstrates the conciseness and utility of the AR API, and illustrates how the API can be used for science scripting.

```
# Invoke a function named 'cds.sesame.resolve' that resolves an object name to a position.
p = ar.cds.sesame.resolve('m32') # returns a structure

# Build a Simple Image Access Protocol (SIAP) query.
s = 'ivo://adil.ncsa/targeted/SIA' # resource ID of the SIAP service
q = ar.ivoa.siap.constructQuery(s,p['ra'],p['dec'],1.0) # returns a query URL

# Execute the SIAP query.
r = ar.ivoa.siap.execute(q) # returns an array of row structures
print 'Rows returned: ', len(r) , 'Column Names: ', r[0].keys()

# Display the query response in a Votable viewer, by sending a PLASTIC message.
me = ar.plastic.hub.registerNoCallBack('script') #register with the hub
m = 'ivo://votech.org/votable/loadFromURL' # message to emit
ar.plastic.hub.requestAsynch(me,m,[q]) # send the message

# Save images selected by the query to MySpace.
home = ar.astrogrid.myspace.getHome() # path to user's myspace home (causes login)
dir = ar.astrogrid.myspace.createChildFolder(home,p['posStr']) # create a new directory
ar.ivoa.siap.saveDatasets(q,dir) # instruct MySpace to download and save the images

# Save an ascii version of the SIAP query response to the same directory.
ar.util.tables.convertFiles(q, 'votable',dir + '/resp.txt','ascii')
```

[6]there are good XML-RPC libraries for (at least) Python, Perl, Java, C, C+, Tcl

[7]the precursor to VoSpace (Graham 2006)

[8]which can be similarly expressed in Perl or IDL

4. Status

Table 1. Some third-party applications that currently use Astro Runtime

Application	Features Used	AR Access	Language
Aladin (Boch 2005)	myspace, remote tasks	RMI	Java
AstroWeka (Walshe 2006)	registry, DAL, dialogs	RMI	Java
SED Builder (Pierfederici & Dolensky 2006)	myspace, registry	XML-RPC	Python
Topcat (Taylor M. 2005)	myspace	XML-RPC	Java
Visivo (Becciani 2005)	remote tasks to HPC	XML-RPC	C++
Voda (Lindroos 2006)	registry, DAL	RMI	Java
Workbench	all	Direct	Java

Astro Runtime has been available for download since September 2005, and is now approaching maturity and a stable release. It is highly configurable, requires minimal setup and is available in a range of flavours: as a stand-alone executable; webstartable; an embeddable library; and as part of the AstroGrid Workbench[9] – a suite of GUI tools for the VO implemented using AR.

Table 1 lists a range of third-party applications that use AR: existing applications (Topcat, Aladin); adaptations of existing applications (AstroWeka); purpose-written for the VO (VODA); and non-GUI services (SED Builder). Furthermore, AR can be used to access VO services from existing packages (e.g. PyRaf or Parseltongue), or for general science scripting.

Our immediate plans are to extend and refine the API; track the developing standards; implement a C library binding; and document AR in an IVOA note.

References

Taylor, J. D. et al. 2006, this volume, [O7.4]

Lindroos J. et al. 2006, this volume, [P3.36]

Graham, M. et al. 2006, this volume, [O7.3]

Taylor, M. B. 2005, in ASP Conf. Ser., Vol. 347, ADASS XIV, ed. P. Shopbell, M. Britton, & R. Ebert (San Francisco: ASP)

Boch T. 2005, `http://eurovotech.org/twiki/bin/view/VOTech/UsageOfAcrApiInAladin`

Becciani U. et al. 2005 `http://wiki.eurovotech.org/bin/view/VOTech/VisIVO`

Walshe B. 2006 `http://astroweka.sourceforge.net/`

Pierfederici F. & Dolensky M. 2006 `http://eurovotech.org/twiki/bin/view/VOTech/SEDBuilder`

[9] `http://www.astrogrid.org/desktop`