Astronomical Data Analysis Software and Systems XII
ASP Conference Series, Vol. 295, 2003
H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds.

ADS Web Services for the Discovery and Linking of Bibliographic Records

Alberto Accomazzi, Günther Eichhorn, Carolyn S. Grant, Michael J. Kurtz, and Stephen S. Murray

Harvard-Smithsonian Center for Astrophysics, 60 Garden Street, Cambridge, MA 02138

Abstract. The NASA Astrophysics Data System (ADS) currently provides free access to over 2.9 million records in four bibliographic databases through a sophisticated search interface. An increasingly larger number of publishers and institutions are using the ADS to verify the existence and availability of references published in the scientific literature. To facilitate the exchange of metadata necessary to establish these links, the ADS is developing prototype Web Services based on emerging industry standards such as SOAP as part of a collaboration with the major NASA Astrophysics Data Centers. In this paper we discuss possible approaches to the implementation of SOAP services and present three different prototypes developed by the ADS group as our contribution to this effort. We conclude with a brief discussion of the issues still confronting data providers and software developers as we embrace these new technologies.

1. Introduction

In the past few years, the Simple Object Access Protocol (SOAP) has emerged as one of the main industry-supported protocols for the implementation of web services. SOAP is an XML-based, platform and language independent protocol that can use HTTP as its transport mechanism. As such, it has become one of the standard ways of exchanging structured data and metadata among web-based services.

There are several SOAP libraries currently available that provide bindings for a variety of programming languages, greatly simplifying the deployment of SOAP clients and servers. These libraries can take care of all aspects related to data serialization so that users can simply write code that passes objects and data structures between clients and servers, without worrying about how such objects are represented in the underlying protocol. As an alternative, most libraries also allow developers to override the default methods used to format and exchange messages between clients and servers. When writing a service that uses SOAP to return data to a client request, the developer can choose three different alternatives:

1. The service returns an object reference to the client. The client then invokes methods on the object to retrieve data from the remote SOAP server. While this may seem the most desirable paradigm (after all the

O in SOAP stands for Object), no standard mechanism exists today for the serialization of objects and methods, nor for the management of state information necessary for this implementation to work. This is still an area where SOAP has not (yet?) delivered the promise of cross-language, object-oriented, distributed computing. Even if the technical issues concerning the serialization of objects between server and client are resolved, the necessity of a round-trip between the two for each method call would present a prohibitive performance barrier for many distributed applications.

2. The service returns an XML document representing a serialization of the query results according to a published schema. The document is embedded in the usual XML-based SOAP envelope used by the client-server protocol. This way of serializing data in a SOAP response (also called "Literal XML Encoding") can be used to override the default data encoding, giving the developer total control of how the data being returned is formatted. However, with this freedom comes the burden of forcing the client to deal with the parsing and validation of the incoming data stream, something that by default is handled transparently by the SOAP protocol.
3. The service returns a pure data structure, that is then transparently serialized by the SOAP library as an XML-based message according to the standard SOAP encoding. The data structure is then unpacked once it reaches the client and can be readily used by the application.

2. ADS SOAP Services Implementation

The promise of interoperability among platforms and languages and the availability of public-domain implementations for this protocol have contributed to the adoption of SOAP by astronomical data providers, among others. The ADS group, in the context of its collaboration with other NASA Data Centers, has started developing prototype SOAP services that can be used to query its search engine and other interfaces used to establish links to its data holdings.

The prototype services we present here provide a SOAP interface to existing functionality that ADS has so far made available via the traditional HTML/HTTP/CGI user interfaces (Accomazzi et al. 1997) as well as through customized client-server interfaces (Eichhorn et al. 1996). We tackled these services first since they are the ones that most of our collaborators use on a daily basis and because they expose the ADS search engine interface and our data holdings in a natural way. To simplify access to these services, we have implemented them following the approach described in point 3 above: the SOAP servers return data structures that can be readily used by the client interfaces. We felt that at this point this implementation is the one which offers the minimum buy-in cost from a user perspective, and is the one that highlights the "S" for "Simple" in SOAP.

We have also attempted to minimize the complexity of the data structures returned by the SOAP servers by adopting a simple representation for the bibliographic elements in our records. While this may fall short of the level of detail desired by some of our users, it greatly simplifies the amount of post-processing

```
#!/bin/env perl
# ADS Reference resolution client via SOAP services
# Usage: refresolver-client.pl refstring [...]

use SOAP::Lite;
use XML::Simple;

my $response = SOAP::Lite
    -> uri('http://ads.harvard.edu/RefResolver')
    -> proxy('http://ads.harvard.edu/ws/bibserver')
    -> resolve(@ARGV);

print XMLout($response->result, noattr => 1);

$ refresolver-client.pl 'Accomazzi, A., et al. 2000, A&AS, 143, 85'
<opt>
  <anon>
    <input>Accomazzi, A., et al. 2000, A&amp;AS, 143, 85</input>
    <bibcode>2000A&amp;AS..143...85A</bibcode>
    <status>Ok</status>
    <url>http://adsabs.harvard.edu/cgi-bin/bib_query?2000A%26AS..143...85A</url>
  </anon>
</opt>
```

Figure 1. Sample PERL SOAP client script for reference resolution. The script uses the public domain PERL library *SOAP::Lite* to send a list of freetext references given on the command line to the SOAP server for identification with ADS records. The server sends back a data structure that is accessible via the variable returned by `$response->result`. In the *SOAP::Lite* implementation, this corresponds to a reference to an array of hash references, which we serialize as an XML document via a call to `XMLout`, available from the *XML::Simple* PERL module.

that client applications need to perform in order to create links to the ADS services. The following SOAP services have been implemented so far:

- **Bibcode Verification:** a service that verifies the existence of bibliographic records and returns their canonical bibcodes and URLs suitable for linking.
- **Reference Resolution:** a service that requests the identification of freetext reference strings with their corresponding bibliographic records in ADS.
- **Abstract Query:** a service that returns a list of records in the ADS abstract databases matching the input search parameters.

A sample PERL SOAP client for reference resolution and an example of its usage is shown in Figure 1.

3. Discussion

The services described in this paper offer new access methods to the existing ADS query interfaces. In this sense, they do not provide new functionalities but rather make the existing functionality available through technology that is becoming an industry standard. While this may encourage the deployment of distributed agents that take advantage of these services, much work still remains to be done by the ADS and the other astronomical data centers before we can

all take advantage of these technologies in a seamless way. Some of the issues we have identified as needing resolution are listed below.

- Publication of interface specifications for web services using standards such as WSDL. These specifications can help implementors in creating clients capable of properly accessing the many web services being deployed.
- Agreement on a common set of XML Schemas that can be adopted by the astronomical data centers. Being able to attach semantic meaning to a dataset representation by making use of a common set of schemas would tremendously increase the level of interoperability between data providers.
- Standardization of protocols and client/server responses. Even assuming most of the data centers settle on a single protocol such as SOAP, there are still issues that need to be resolved. Should there be services returning objects rather than datasets? Should stateful information be included in the messages exchanged by clients and servers?

4. Conclusions

We have implemented the SOAP prototypes described in this paper in response to the suggestions and requests from other NASA Astrophysics Data Centers. Their deployment and the deployment of similar services by other data providers allows interested parties to test how usefully they can be integrated in distributed applications. Currently, access to these SOAP services is available upon request. For more information on how to register to gain access to them, please visit the following url: `http://ads.harvard.edu/pubs/ws`.

We hope that the availability of these interfaces will provide us with feedback about their usefulness and suggestions about how they can be improved. To this end, we solicit any and all comments from any potential users.

Acknowledgments. The ADS is funded by NASA Grant NCC5-189.

References

Accomazzi, A., Eichhorn, G., Kurtz, M. J., Grant, C. S., & Murray, S. S. 1997, in ASP Conf. Ser., Vol. 125, Astronomical Data Analysis Software and Systems VI, ed. G. Hunt & H. E. Payne (San Francisco: ASP), 357

Eichhorn, G., Accomazzi, A., Grant, C. S., Kurtz, M. J., & Murray, S. S. 1996, in ASP Conf. Ser., Vol. 101, Astronomical Data Analysis Software and Systems V, ed. G. H. Jacoby & J. Barnes (San Francisco: ASP), 569