Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

Web Services in Telescope Interoperability

Markku Verkkoniemi

Nordic Optical Telescope, Email: mverkkon@not.iac.es

Jacob W. Clasen

Nordic Optical Telescope, Email: jclasen@not.iac.es

Peter M. Sørensen

Nordic Optical Telescope, Email: pms@not.iac.es

Abstract. Object access is a tool that is feasible for interoperability of telescope operations. Different subsystems need to be accessed in a standardized way. Simple Object Access Protocol, SOAP, provides the core of this approach. Web Services is the wider context of this peer-to-peer protocol.

A protocol that is used in virtual observatory interconnectivity can also be used for controlling telescope operations. The learning curve of available packages is easy to overcome and people having experience from different operating systems can start coding in SOAP in a matter of hours.

1. Observing Computational Needs at NOT

Nordic Optical Telescope[1] has subsystems controlling different parts of the telescope and the instruments connected to it. These subsystems need a common interface so that the observers can write scripts to control the complete observing flow in a high-level language. The efficiency of the observing run can be improved by writing these scripts in advance.

In addition to enabling the scripting, a common interface to the subsystems facilitates computer control of the telescope. It also helps future subsystems to fit in to the complete system.An abstract interface definition is useful for the staff members and observers alike to understand various aspects of the operations.

2. SOAP and Its Implementation at NOT

World Wide Web Consortium, W3C[2] has defined a set of standards for application to application communication. The core of these programmatic interfaces

[1]`http://www.not.iac.es`

[2]`http://www.w3.org`

made available is SOAP that enables the communication. In our implementation SOAP uses XML on HTTP to enable RPC calls to different systems. It is totally independent of operating systems and programming languages. We are at present using C, C++, Perl and Python with SOAP packages on Linux.

With these packages remote objects can be accessed as if they were local to the application. The intrinsics of the actual remote calls are transparent to the application developer.

At NOT we are using SOAP protocol version 1.1. The core package for the operations is gSOAP[3] , which when using with C has a small memory footprint, only around 150 kB per server process. This allows us to have a multitude of services running on a single Intel PC computer.

The overhead for a typical subsystem call with gSOAP between two 2 GHz Pentium CPU based PCs is around 1 ms. In our subsystem operations this is very well accepted.

In practical terms SOAP works like this:

- The SOAP library on the client side is constructing an XML envelope for the call parameters.
- The standardized XML message is sent on the wire to the server listening at a specific TCP/IP port on a computer.
- The server does its task accordingly, and sends back a message to the client.

3. Thin Client—Thin Server

Each client-server pair consists of SOAP compatible programs. The port numbers for respective services are held in /etc/services file, and a library function is used to pick up the port number. Each subsystem has a server for its set of parallel functionality. For example, if the filter holders of an instrument *can* be operated simultaneously, they *shall* have a server for each single filter holder.

Instead of a monolithic server solution we have several tiny processes doing one thing, as seen in Figure 1.

The logging of the system is dealt with a syslog server that reports the functionality in a common log that is filtered to be displayed for the user. There are four different levels of logging: notice, warning, error, and debug. The display filtering is performed based on these levels.

The operating system is taking care of time-sharing between the servers, and the SOAP library takes care of the message queuing. This way we need not implement multi-threading in the servers. Operations on the subsystems can be made simultaneously, and both client and server program code complexity is reduced to a very simple level.

The TCS does not have any SOAP libraries. Therefore, there is a computer that is implementing SOAP server wrappers to the TCS RPC calls that operate the telescope itself.

[3]`http://http://www.cs.fsu.edu/~engelen/soap.html`

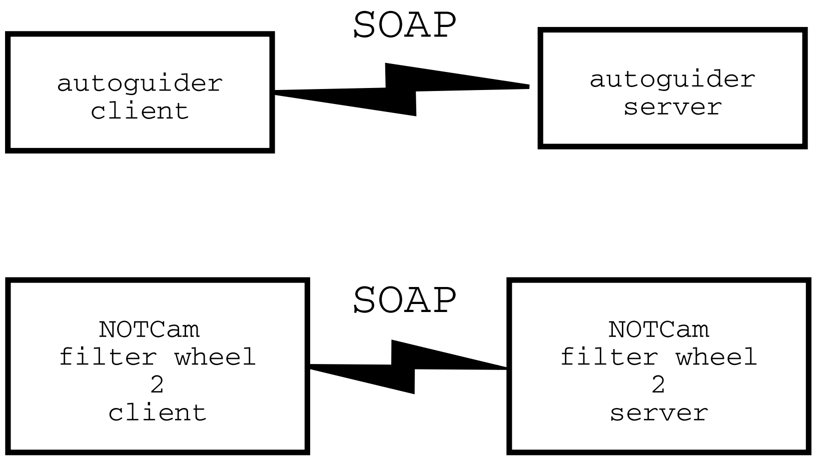

Figure 1. Client/Server pair examples.

4. Status of Subsystems

The status display is updated by the subsystems. When starting subsystems, their status can be queried. It is the responsibility of the client to make sure that the server can perform the request, and it is the responsibility of the server to know its own status. This status is then displayed in a separate subsystem with a SOAP interface visible to the person using the telescope.

A well-defined interface to the status display enables several approaches to the displays that can be used for real-time light path analysis, for example.

5. WSDL

Web Services Description Language is defining the interface to the server. This interface definition is automatically produced by gSOAP, and by that definition a client can be programmed independently to send SOAP requests.

The WSDL file is also created in XML.

With this approach you can start programming the server without clear interface definition. When the server is made ready, the interface is created by the library. This shortens the total development time.

6. What about security?

When mentioning using Web Services for controlling telescope operations a question usually arises about security.

Asynchronous SOAP operations can easily be controlled with packet filtering techniques. As each atomic service has a dedicated server port number, fine-grain control is possible.

There are no secrets sent on the wire, so encrypting the messages is not necessary.

Any RPC mechanism between two or more computers can be interfered. Closed standards are almost as easy to be eavesdropped as open standards, if an unauthorized program can listen to your network traffic. Security by obscurity or complexity is poor security. You have to be able to detect and control what traffic is passing in your network segment.

Both the SOAP client and server are running with user privileges. The network security services should be run in system level.

7. Future Steps

A well-defined interface can be used for many projects. At first, this work is used to enable scripting, but later it can, with possible modifications, be used for projects for remote observing, for instance.

Image transfering is not tested, yet. XML binary transfer is enabled by MIME techniques that need encoding and decoding, but a transfer by reference to the image location is also possible. Then some protocol is used for image transfer instead of Web Services.

References

Wenger, M. et al. 2001, in ASP Conf. Ser., Vol. 238, ADASS X, ed. F. R. Harnden, Jr., F. A. Primini, & H. E. Payne (San Francisco: ASP), 213