

APL and Web Services

Brian P Becker
Blue Dolphin Solutions
APL2010 Workshop

The Challenge(s)

- ▶ There is quite a bit of APL-based functionality and data that's largely unavailable for use outside of APL.
- ▶ There is quite a bit more nonAPL-based functionality and data that's largely unavailable for use within APL.
- ▶ APLers are generally better at solving problems than designing user interfaces.

Enter SAWS...

- ▶ Stand-Alone Web Service Framework
- ▶ Stand-Alone – doesn't require Microsoft IIS, Apache, IBM WebSphere or other server software.
- ▶ All you need is an open port and network connectivity.
- ▶ Web Service – uses standard protocols and formats, XML, HTTP, SOAP 1.1 and WSDL.
 - But YOU don't have to learn them! 😊
- ▶ Uses Conga for TCP/IP communications over HTTP and HTTPS.

Benefits of SAWS

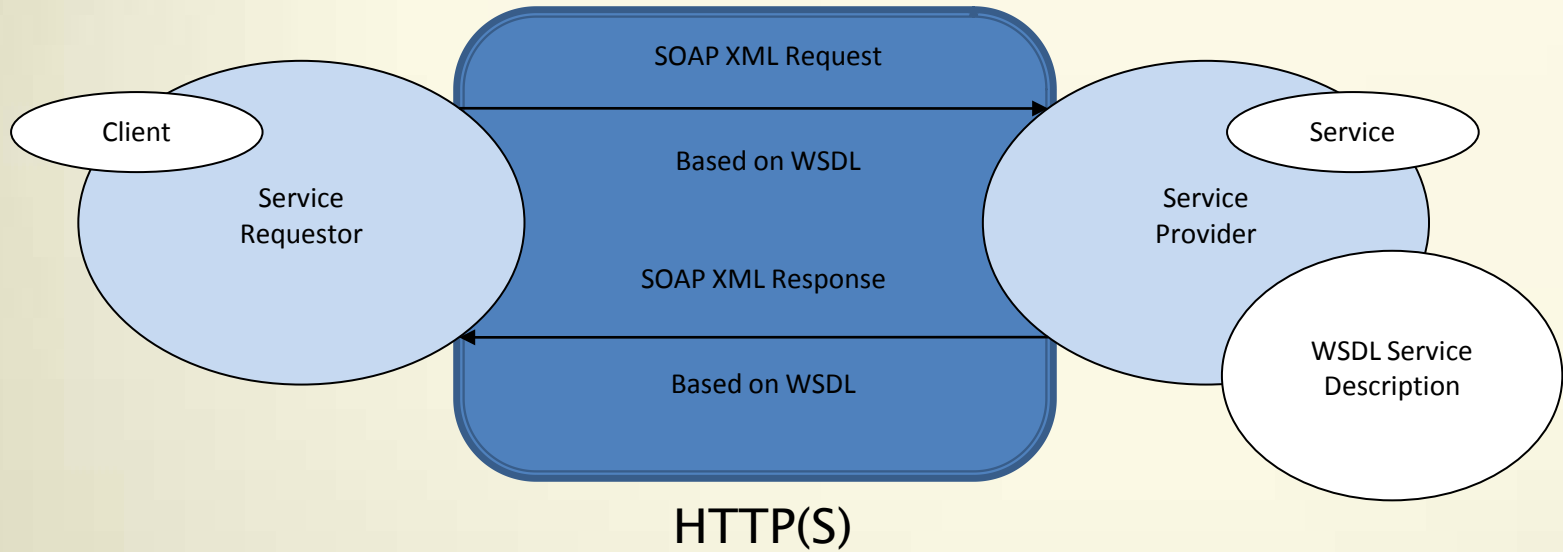
- ▶ Use third party Web Services
- ▶ Gives access to legacy APL applications
- ▶ Build new service oriented applications
- ▶ Platform independence, runs on Windows, Linux, AIX... wherever Dyalog APL v12.1 runs
- ▶ Level the playing field – may reduce some management reluctance to use APL

What are Web Services?

- ▶ **Web Services** are automated information services that are conducted over the Internet, using standardized technologies and formats/protocols that simplify the exchange and integration of large amounts of data over the Internet. They make it easier to conduct work across organizations regardless of the types of operating systems, hardware/software, programming languages, and databases that are being used.

<http://epa.gov/exchangenetwork/glossary.html>

Anatomy of a Web Service



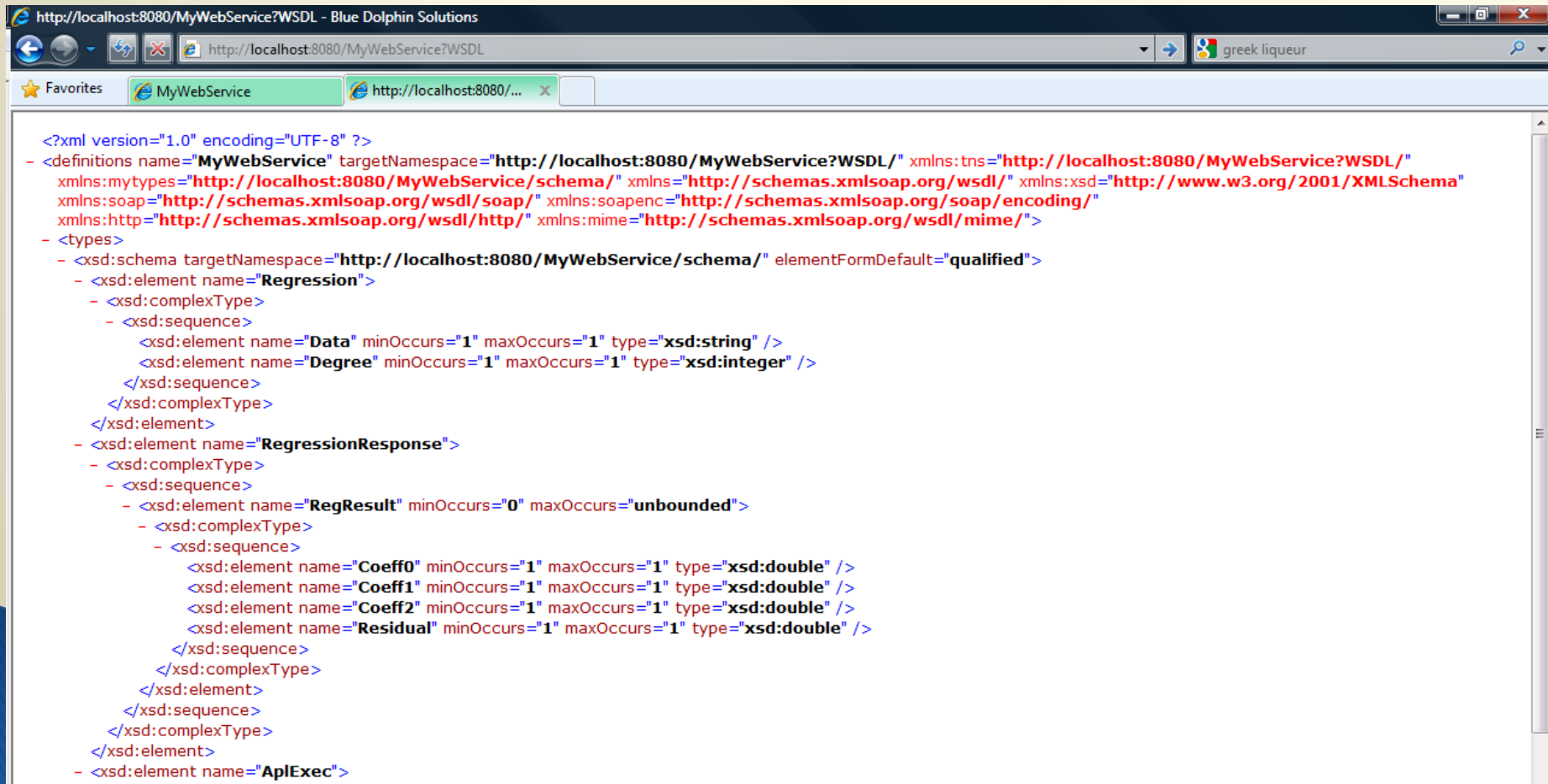
SOAP

- ▶ Originally Simple Object Access Protocol
Now SOAP stands for... SOAP
- ▶ XML Based

```
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="Some-URI">
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

WSDL

- ▶ Web Service Description Language
- ▶ Also XML based



```
<?xml version="1.0" encoding="UTF-8" ?>
- <definitions name="MyWebService" targetNamespace="http://localhost:8080/MyWebService?WSDL/" xmlns:tns="http://localhost:8080/MyWebService?WSDL/"
  xmlns:mytypes="http://localhost:8080/MyWebService/schema/" xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/" xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/">
- <types>
- <xsd:schema targetNamespace="http://localhost:8080/MyWebService/schema/" elementFormDefault="qualified">
- <xsd:element name="Regression">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="Data" minOccurs="1" maxOccurs="1" type="xsd:string" />
  <xsd:element name="Degree" minOccurs="1" maxOccurs="1" type="xsd:integer" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
- <xsd:element name="RegressionResponse">
- <xsd:complexType>
- <xsd:sequence>
- <xsd:element name="RegResult" minOccurs="0" maxOccurs="unbounded">
- <xsd:complexType>
- <xsd:sequence>
  <xsd:element name="Coeff0" minOccurs="1" maxOccurs="1" type="xsd:double" />
  <xsd:element name="Coeff1" minOccurs="1" maxOccurs="1" type="xsd:double" />
  <xsd:element name="Coeff2" minOccurs="1" maxOccurs="1" type="xsd:double" />
  <xsd:element name="Residual" minOccurs="1" maxOccurs="1" type="xsd:double" />
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
- <xsd:element name="AplExec">
```


Key SAWS Feature and Functions

- ▶ `SAWS.Call` - invoke Web Services
- ▶ `SAWS.Run` - provide Web Services
- ▶ `SAWS.RunSecure` - provide secure Web Services using TLS
- ▶ `SAWS.TRACE` - used to display trace information
- ▶ `SAWS.DEBUG` - set various debugging features

Interfacing SAWS with Legacy Code

- ▶ Copy the SAWS namespace into your workspace. SAWS is self-contained.
- ▶ If possible, create an API namespace for your application.
- ▶ Write function(s) to interface to your legacy code
- ▶ Write BuildAPI to describe the interface to your application.

Things to Consider

- ▶ User Authentication and Authorization
- ▶ Web Services are Stateless
- ▶ 3rd Party Service Availability and Reliability
- ▶ Load Balancing

SAWS Evolution

- ▶ Proof of Concept – 2008–9
 - Provide and Consume Web Services
- ▶ Useful Tool – 2010
 - Stable Code
 - Enhanced Functionality
- ▶ Industrial Quality – 2011
 - Error Logging
 - Usage Logging
 - Management Console
 - WS-* specification support
 - And more...

