Introduction to SOAP

A Walkthrough of Core Technical Concepts

Frank Mantek
Frankman@microsoft.com

Overview

- What is SOAP?
  - Technical walkthrough of SOAP
  - What it is in practical terms
  - What the design is behind it
- What is the purpose of SOAP
  - What does it solve?
  - What doesn't it solve?
- Other issues involved
  - WSDL
  - DISCO
  - UDDI
  - Security

SOAP/1.1 Spec Status

- SOAP/1.1 was submitted to W3C and became a W3C Note on May 8, 2000
- W3C has just started a WG on XML Protocols
  - Paul Cotton and Henrik Nielsen are the MS WG representatives
    - Paul Cotton will present the W3C Process tomorrow
  - SOAP is the starting point for this work
  - Simple rules for standards work:
    - Things change as result of the work
    - The more you put in, the more likely you are to affect the outcome
    - But no guarantee that it always work that way - outcome is based on rough consensus

SOAP/1.1 Authors

- Don Box, DevelopMentor
- David Ehnebuske, IBM
- Gopal Kakivaya, Microsoft
- Andrew Layman, Microsoft
- Noah Mendelsohn, Lotus
- Henrik Frystyk Nielsen, Microsoft
- Satish Thatte, Microsoft
- Dave Winer, UserLand Software

W3C Submitters

- Ariba, Inc.
- Commerce One, Inc.
- Compaq Computer Corporation
- DevelopMentor, Inc.
- Hewlett Packard Company
- International Business Machines Corporation
- IONA Technologies
- Lotus Development Corporation
- Microsoft Corporation
- SAP AG
- UserLand Software Inc.

SOAP Outside Microsoft

- SOAP is being developed very much in Web style:
  - Very distributed community with broad support
- Available Implementations
  - Apache SOAP (started by IBM)
  - Developmentor SOAP toolkits in Perl and Java
  - SOAP::lite (Perl)
  - Libwww based SOAP
  - IONA
  - ...

What is SOAP?

- SOAP is a simple, lightweight XML protocol for exchanging structured and typed information on the Web
- Overall design goal: KISS
  - Can be implemented in a weekend
  - Stick to absolutely minimum of functionality
- Make it Modular and Extensible
  - No application semantics and no transport semantics
  - Think “Web based datagram”

SOAP's Four Parts:

- An extensible envelope expressing (mandatory)
  - **what** features and services are represented in a message;
  - **who** should deal with them,
  - **whether** they are optional or mandatory.
- A set of encoding rules for data (optional)
  - Exchange instances of application-defined data types and directed graphs
  - Uniform model for serializing non-syntactic data models
- A Convention for representation RPC (optional)
  - How to make calls and responses
- A protocol binding to HTTP (optional)

SOAP Example in HTTP

```
POST /Accounts/Henrik HTTP/1.1
Host: www.webservicebank.com
Content-Length: nnnn
Content-Type: text/xml; charset="utf-8"
SOAPAction: "Some-URI"
```

SOAP-HTTP Binding
HTTP Request
SOAP Body
SOAP Header
SOAP Envelope

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP:Header>
      <t:Transaction xmlns:t="some-URI" SOAP:mustUnderstand="1">
            5
      </t:Transaction>
   </SOAP:Header>
   <SOAP:Body>
      <m:Deposit xmlns:m="Some-URI">
         <m:amount>200</m:amount>
      </m:Deposit>
   </SOAP:Body>
</SOAP:Envelope>
```

... or SOAP by Itself...

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP:Header>
      <m:MessageInfo xmlns:m="http://www.wapforum.org/soap/message">
              <m:to href="mailto:you@your.com"/>
              <m:from href="mailto:me@my.com"/>
              <m:contact href="mailto:someone@my.com">
      </m:MessageInfo>
   </SOAP:Header>
   <SOAP:Body>
      <msg:Message xmlns:m="http://www.wapforum.org/soap/message">
              <msg:subject>Your house is on fire!</msg:subject>
              <msg:liveUpdate href="http://your.house.is.on.fire.com/rightnow"/>
      </msg:Message>
   </SOAP:Body>
</SOAP:Envelope>
```

SOAP is a Protocol!

- What does this mean?
  - It is **not** a distributed object system
  - It is **not** an RPC system
  - It is **not even** a Web application
- Your application decides what your application is!
  - You can build a tightly coupled system
  - ...or...
  - You can build a loosely coupled system
- Why does this matter?
  - It means that you have to think about how you design your application

Things to Think About…

- What if a WebService is down?
  - Do I just die?
  - …or do I allow for graceful fail-over?
- Where do I put state associated with a WebService?
  - In global variables?
  - …or do I stick it in the message so that the message represents all the state needed to carry on?
- Where do I put assumptions about the implementation?
  - In the message?
  - …or in the implementation leaving the message implementation independent?

Myths about SOAP

- SOAP is for RPC only! No...
  - SOAP doesn't define or imply a programming model
  - It can be used for messaging, RPC, Distributed object systems etc.
- SOAP is for HTTP only! No...
  - SOAP can be used in combination with any protocol that can carry a SOAP envelope
  - SOAP currently defines bindings to HTTP and HTTP Extension Framework - others can be added
- SOAP is request/response! No...
  - SOAP doesn't define a message exchange pattern
  - Can be defined in SOAP or inherited from protocol binding

The Amtrak Message Model

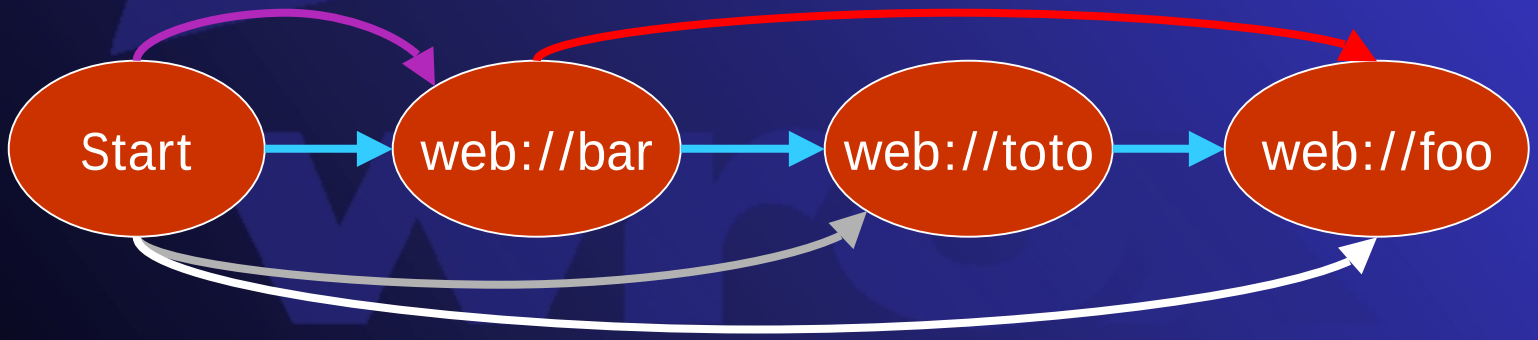
- A train is a SOAP message
  - It starts in the departure city, stops at a set of intermediate cities, and stops at the destination city
- A city is a SOAP processor
  - Ensures that passengers get on and off
  - Passenger tickets are verified
- A passenger is a SOAP feature or service
  - A passenger can get on an off at any stop
  - Passengers can mix in arbitrary ways
- Message model can be put together to form arbitrary message graphs

The SOAP Envelope

- A SOAP envelope defines a SOAP message
  - Basic unit of exchange between SOAP processors
  - Highly flexible
- SOAP messages are one-way transmissions
  - From sender through intermediaries to receiver
  - Often combined to implement patterns such as request/response
- Messages are routed along a "message path"
  - Allows for processing at one or more intermediate nodes in addition to the ultimate destination node.
  - A node is a SOAP processor and is identified by a URI

SOAP Headers

- Allows for modular addition of features and services
  - Open-ended set of headers
    - Authentication, privacy, security etc. etc.
  - Can address any SOAP processor using the "actor" attribute
  - Can be optional/mandatory using the "mustUnderstand" attribute



Semantics of Headers

- Contract between sender and recipient
  - Recipient is described by "actor" attribute
- Allows for different types of negotiation:
  - Take it or leave it
  - Let's talk about it
- And for different settings:
  - Server dictated
  - Peer-to-peer
  - Dictated by third party

actor Attribute

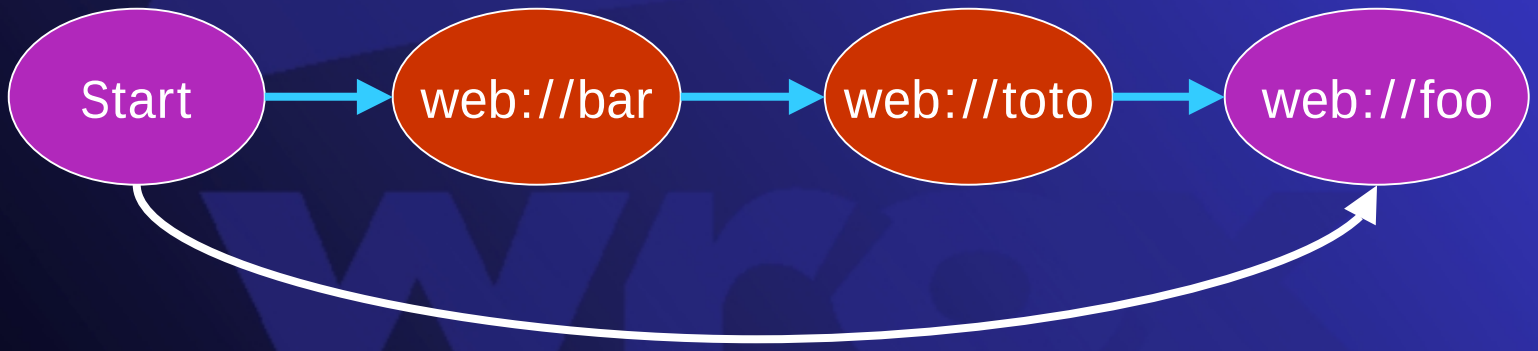
- The "Actor" attribute is a generalization of the HTTP Connection header field
  - Instead of hop-by-hop vs. end-to-end, the actor attribute can address any SOAP processor because it is a URI
  - Special cases:
    - "next hop" has a special URI assigned to it
    - "end" is the default destination for a header

mustUnderstand Attribute

- The "mustUnderstand" is the same as the "mandatory" in the HTTP Extension Framework
  - Requires that the receiving SOAP processor must accept, understand and obey semantics of header or fail
  - This allows old applications to gracefully fail on services that they do not understand

SOAP Body

- Special case of header
  - Default contract between sender and ultimate recipient
  - Defined as a header with attributes set to:
    - Implicit mustUnderstand attribute is always "yes"
    - Implicit actor attribute is always "the end"



SOAP Fault

- The SOAP Fault mechanism is designed to support the composability model
  - Is not a scarce resource as in HTTP where there can be only one (the Highlander principle)
- A SOAP message can carry one SOAP Fault element
  - Must be placed in the Body of the message
- The Fault Detail element carries information for faults resulting from the body
- Detail information for faults resulting from headers are carried in the header
- The SOAP fault code extension mechanism is for SOAP only
  - Application faults should use existing SOAP fault codes
  - Client is for request faults, Server is for processing faults

SOAP Fault Example

- A SOAP message containing an authentication service:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP:Header>
      <m:Authentication xmlns:m="http://www.auth.org/simple">
               <m:credentials>Henrik</m:credentials>
      </m:Authentication>
   </SOAP:Header>
   <SOAP:Body>
      … body goes here …
   </SOAP:Body>
</SOAP:Envelope>
```

SOAP Fault Example… 2

- …results in a fault because the credentials were bad:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope"
  SOAP:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP:Header>
      <m:Authentication xmlns:m="http://www.auth.org/simple">
               <m:realm>Magic Kindom</m:realm>
      </m:Authentication>
   </SOAP:Header>
   <SOAP:Body>
      <SOAP:Fault>
           <SOAP:faultcode>SOAP:Client</faultcode>
           <SOAP:faultstring>Client Error</faultstring>
       </SOAP:Fault>
   </SOAP:Body>
</SOAP:Envelope>
```

WSDL

- An XML-based grammar for describing the capabilities of Web Services
- Extensible
- Jointly developed by Microsoft and IBM
  - Convergence of SDL/SCL and NASSL
- Similar in concept to IDL, but it's not IDL
  - IDL is platform dependent
  - WSDL is platform independent

WSDL Example (1)

```
<?xml version="1.0"?>
<message name="GetLastTradePriceRequest">
   <part name="tickerSymbol" element="xsd:string"/>
   <part name="time" element="xsd:timeInstant"/>
</message>
<message name="GetLastTradePriceResponse">
  <part name="result" type="xsd:float"/>
</message>
 <portType name="StockQuotePortType">
   <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceRequest"/>
      <output message="tns:GetLastTradePriceResponse"/>
   </operation>
</portType>
```

WSDL Example (2)

```
<binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="GetLastTradePrice">
  <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
        <input>
        <soap:body use="encoded"
        namespace="http://example.com/stockquote"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
        </input>
  <output>
        <soap:body use="encoded" namespace="http://example.com/stockquote"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
  </output>
  </operation>
</binding>
```

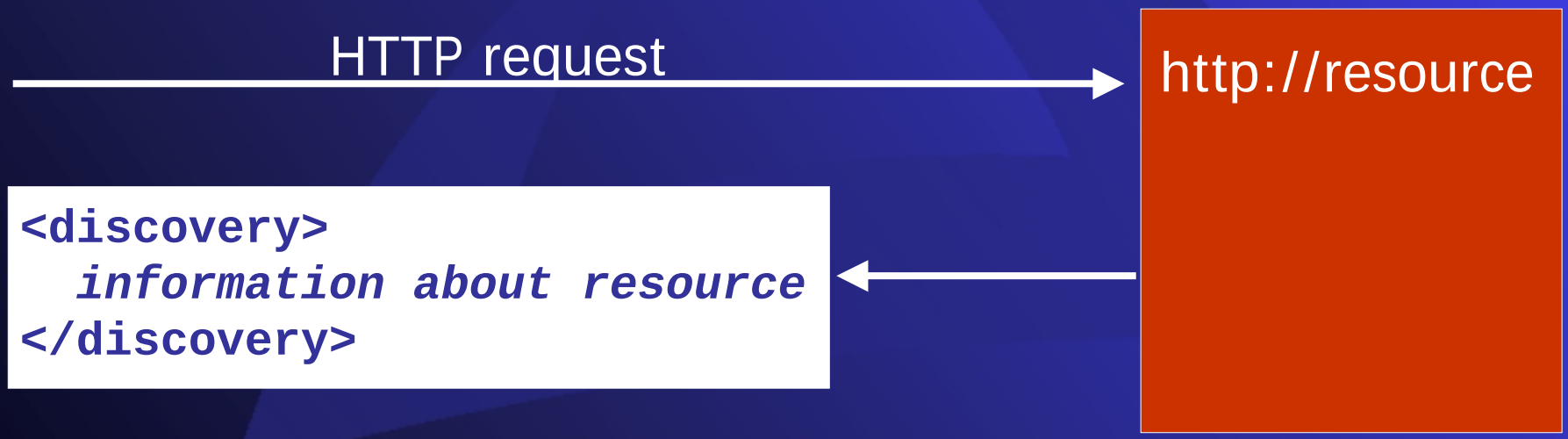
WSDL Example (3)

```
<service name="StockQuoteService">
   <documentation>My first service</documentation>
   <port name="StockQuotePort" binding="tns:StockQuoteBinding">
        <soap:address location="http://example.com/stockquote"/>
   </port>
</service>
</definitions>
```

DISCO – Discovery Protocol

- Retrieve Service Descriptions
- Locate Type of SOAP Service at URL

DISCO – How it works



Retrieving WSDL

- Using to retrieve WSDL's

```
<discovery>
  <contractRef href='URL'/>
</discovery>
```

- SOAP services should support DISCO
  - Return DISCO document on HTTP GET
  - Handle SOAP on HTTP POST

UDDI

- Universal Description, Discovery and Integration
- A project to speed interoperability and adoption for web services
  - Standards-based specifications for service description and discovery
  - Shared operation of a business registry on the web
- Partnership among industry and business leaders
- Open process with clear roadmap to a standards body

SOAP and Security

- IPSEC: Point-to-point entity authentication and channel encryption
  - Authentication is usually via a pre-shared secret
  - Can be done router-to-router
  - “one size fits all” encryption
- SSL: Point-to-point server authentication and channel encryption
  - De facto standard on the web
  - Builds in PKIX (X.509) trust model
  - Crypto heavy lifting done by the server
  - “one size fits all” encryption

Soap and Security (cont)

- S/MIME: Mutual authentication and message-based signature and encryption
  - Supports “wrapped” only. Opening breaks signature and encryption
  - Email is either point-to-point or through a DL. Does not support “pass-through” signing and encryption.
  - Builds in X.509 PKIX trust model
-

Why something new?

- SOAP security has different requirements. It must:
  - Be optional
  - Be stateless
  - Be granular: element signing and encryption
  - Be lightweight
  - Accommodate multi-hop and broadcast messaging
  - Support persistent encryption and signing
  - Require only XML tools
  - Remain trust management agnostic

What we need

- Entity Authentication: none, sender, receiver, mutual
- Message Security:
  - Sealing : make it tamper-proof
  - Privacy : encrypt it
  - Authentication : digitally sign it
  - Make all of this work together
- Protocol Design Issues:
  - Minimize round trips
  - (Optionally) create/maintain “session” state
  - Provide private key proof-of-possession
  - Prevent replay attacks

Building Blocks

- XML Digital Signature (XMLDSig)
  - W3C/IETF proposed standard (@w3.org/dsig)
  - Demo at the PDC
  - Committed for .NET Runtime Beta 1
- XML Encryption (XMLEncrypt)
  - W3C workshop scheduled Nov. 2
  - Briefing package (requirements + proposal) available
  - Broad-based interest

Microsoft tools for SOAP

- SOAP Sample toolkit
- .NET framework

Wo gibt's weitere Info's?

- msdn xml workshop
  - http://msdn.microsoft.com/xml/default.asp
- msdn quickie
  - http://www.microsoft.com/germany/msdn/quickie
- msdn TechTalk-Newsgroup
  - news://msnews.microsoft.com/microsoft.public.de.german.techtalk
- Developer Mentor
  - http://www.develop.com/soap/default.htm
  - Very active listserver
- .Net information
  - http://msdn.microsoft.com/net/

Fragen!?

Where do you want to go today?

Microsoft