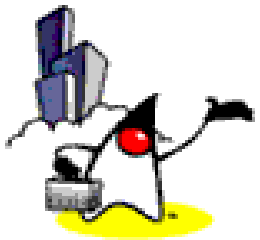
SOAP 1.2

(Simple Object Access Protocol)



Sang Shin

sang.shin@sun.com
Java™ Technology Evangelist
Sun Microsystems, Inc.
www.javapassion.com/webservices

Disclaimer & Acknowledgments

- Even though Sang Shin is a full-time employee of Sun Microsystems, the contents here are created as his own personal endeavor and thus does not reflect any official stance of Sun Microsystems.
- Sun Microsystems is not responsible for any inaccuracies in the contents.
- Acknowledgments
  - Most slides and speaker notes in this presentation are created from W3C SOAP 1.2 Primer and W3C SOAP specifications

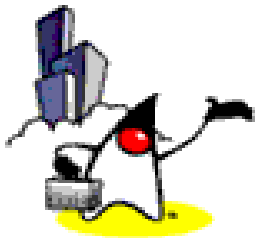
Revision History

- 10/01/2002: version 1, Created (Sang Shin)
- 01/30/2004: version 2, updated (Sang Shin)
- 09/28/2004: version 3, minor modifications (Sang Shin)
- Things to do
  - speaker notes need to be added

Agenda

- What is and is not SOAP?
- SOAP Message Structure
- SOAP Terminology
- SOAP Message Exchange
- SOAP Processing Model
- SOAP Protocol Binding
- SOAP Intermediaries
- Document vs. RPC
- SOAP Encoding
- Quick overview of Java API for SOAP
- Changes from SOAP 1.1 to SOAP 1.2

What is SOAP?
What is not SOAP?
Where is SOAP?



What is SOAP? W3C Definition

- SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment
- SOAP uses XML technologies to define an extensible messaging framework providing a message construct that can be exchanged over a variety of underlying protocols
- The framework has been designed to be independent of any particular programming model and other implementation specific semantics

What is SOAP?

- Simple Object Access Protocol
- Wire protocol similar to
  - IIOP for CORBA
  - JRMP for RMI
- XML is used for data encoding
  - “text” based protocol vs. “binary” protocol
- Supports XML-based RPC (Remote Procedure Call)

Do I Need to know how SOAP works in detail as a Java Developer?

- Yes
  - Understanding it will help you to build better application
  - Ex) Understanding how TCP/IP will help you build better TCP/IP application
- No
  - You will mostly likely use high-level API (JAX-RPC) to build Web applications
  - How SOAP works is hidden from developers

What is SOAP?

- Stateless
- One-way message exchange paradigm
  - Applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information
- Silent on the semantics of any application-specific data it conveys

What SOAP is Not

- Not a component model
  - So it will not replace objects and components, i.e. EJB™, JavaBeans™
- Not a programming language
  - So it will not replace Java
- Not a solution for all
  - So it will not replace other distributed computing technologies such as RMI

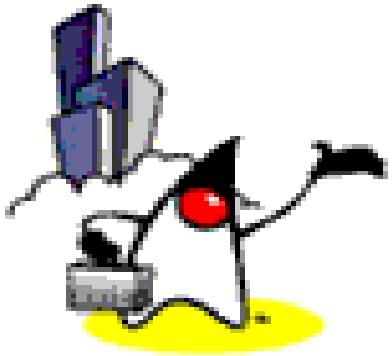
SOAP Design Goals

- Simplicity
- Extensibility
  - New standards define new semantics
- Features not supported (by design)
  - Distributed garbage collection
  - Object by reference
  - Activation
  - Message batching

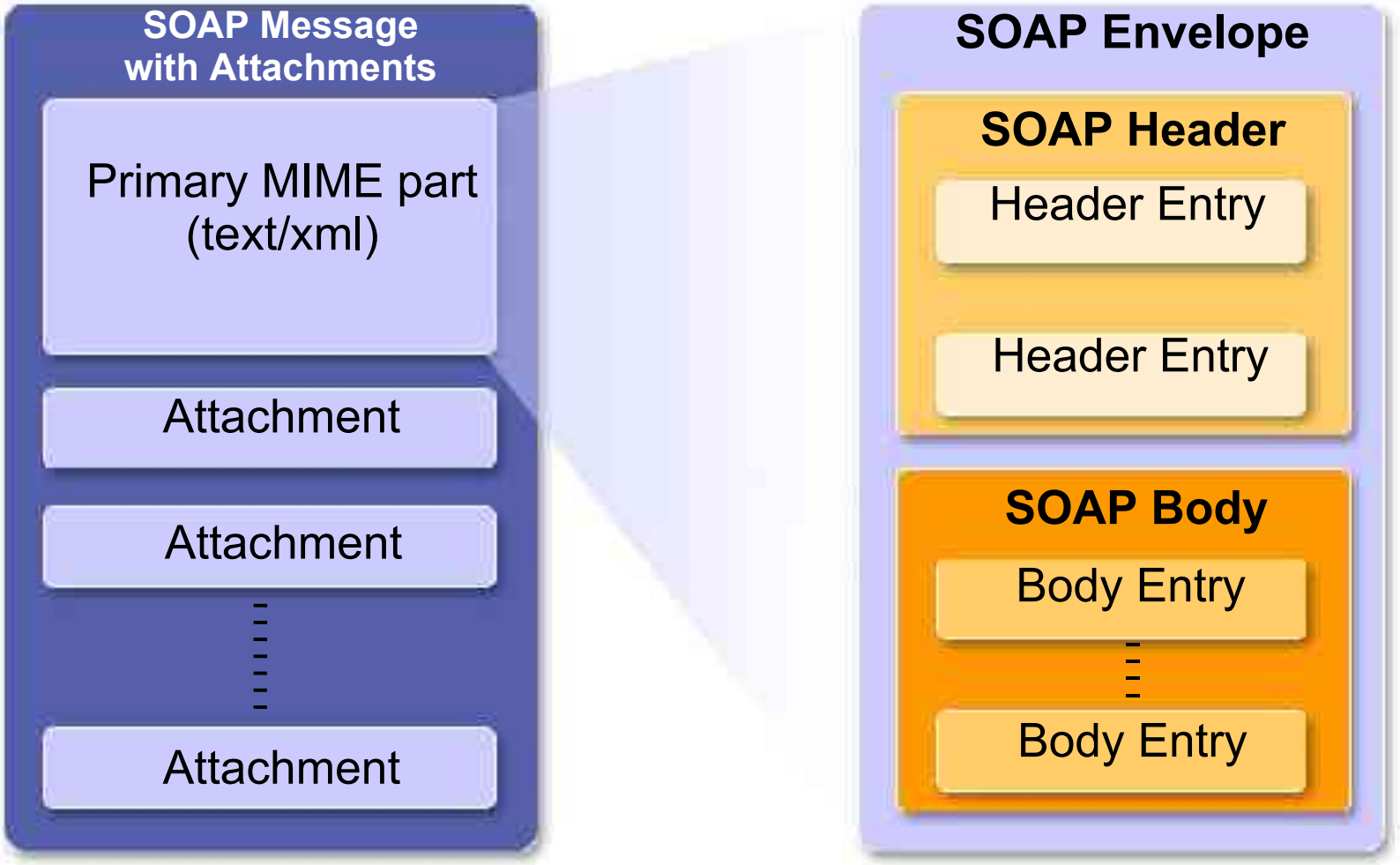
Where is SOAP?

- SOAP 1.2 is W3C recommendation
- SOAP 1.2 Part 1 defines
  - SOAP envelope
  - Protocol binding framework
- SOAP 1.2 Part 2 defines
  - Data model for SOAP
  - Binding to HTTP

Message Structure



SOAP Message Structure



SOAP Message Envelope

- Embedded Information
  - Namespaces
  - Encoding information
- Header
  - Optional
  - Can be handled by intermediaries
- Body
  - Mandatory
  - Handled only by ultimate receiver

SOAP Header (<env:Header>)

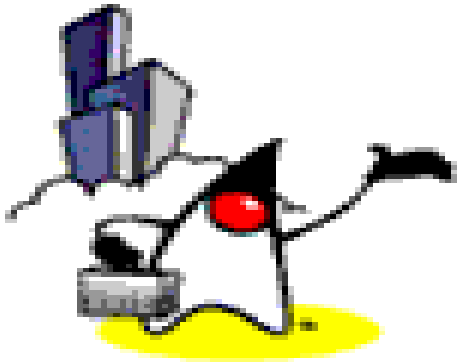
- Used for extension
  - Context
  - Authentication
  - Transaction
  - Management
  - Many other higher level semantics
- Made of Header blocks (Header entries)
- Most Web services standard activities are basically defining standard header entries for a particular domain

SOAP Header Blocks (Entries)

- Child elements of SOAP Header
- Designed in anticipation of various uses for SOAP by SOAP intermediaries
  - Can be individually targeted at SOAP nodes
  - Allows SOAP intermediaries to provide value-added services
- May be inspected, inserted, deleted or forwarded by SOAP nodes encountered along a SOAP message path

SOAP Body (<env:Body>)

- Made of Body blocks (Body entries)
- Consumed by Ultimate SOAP receiver
- Carry end-to-end information
  - Application data (XML document) (document style)
  - RPC method and parameters (rpc style)
  - SOAP fault



Message Structure
Fault Message

SOAP Fault (<env:Fault>)

- Used to carry error and/or status information
- Four sub-elements
  - *faultcode*
  - *faultstring*
  - *faultactor*
  - *detail*

Pre-defined SOAP faultcode values

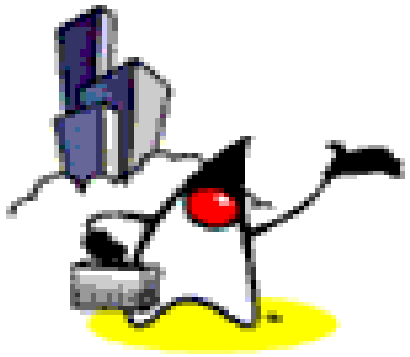
- VersionMismatch
  - Invalid namespace in SOAP envelope
- MustUnderstand
  - Receiver mode cannot handle *mustUnderstand* SOAP header block
- Client
  - Indicates client side error
- Server
  - Indicates server side error

SOAP Fault Example: Cause

```
<env:Envelope
        xmlns:env='http://www.w3.org/2001/06/soap-envelope'>
        <env:Header>
            <abc:Extension1
                    xmlns:abc='http://example.org/2001/06/ext'
                    env:mustUnderstand='1' />
            <def:Extension2
                    xmlns:def='http://example.com/stuff'
                    env:mustUnderstand='1' />
        </env:Header>
        <env:Body>
            . . .
        </env:Body>
</env:Envelope>
```

SOAP Fault Example: Result

```
<env:Envelope xmlns:env='http://www.w3.org/2001/06/soap-envelope'
              xmlns:f='http://www.w3.org/2001/06/soap-faults' >
    <env:Header>
        <f:Misunderstood qname='abc:Extension1'
                         xmlns:abc='http://example.org/2001/06/ext'/>
        <f:Misunderstood qname='def:Extension2'
                         xmlns:def='http://example.com/stuff'/>
    </env:Header>
    <env:Body>
        <env:Fault>
            <faultcode>MustUnderstand</faultcode>
            <faultstring>
                One or more mandatory headers not understood
            </faultstring>
        </env:Fault>
    </env:Body>
</env:Envelope>
```

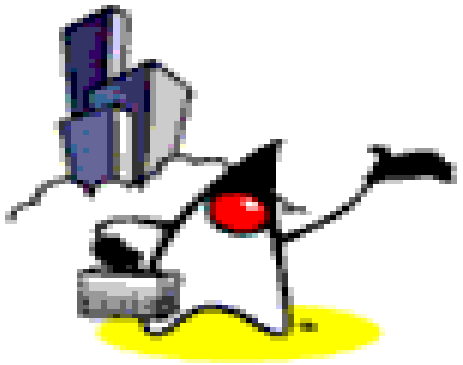


Message Structure

Where do you put your data, in Header block or Body block?

Header Block or Body Block?

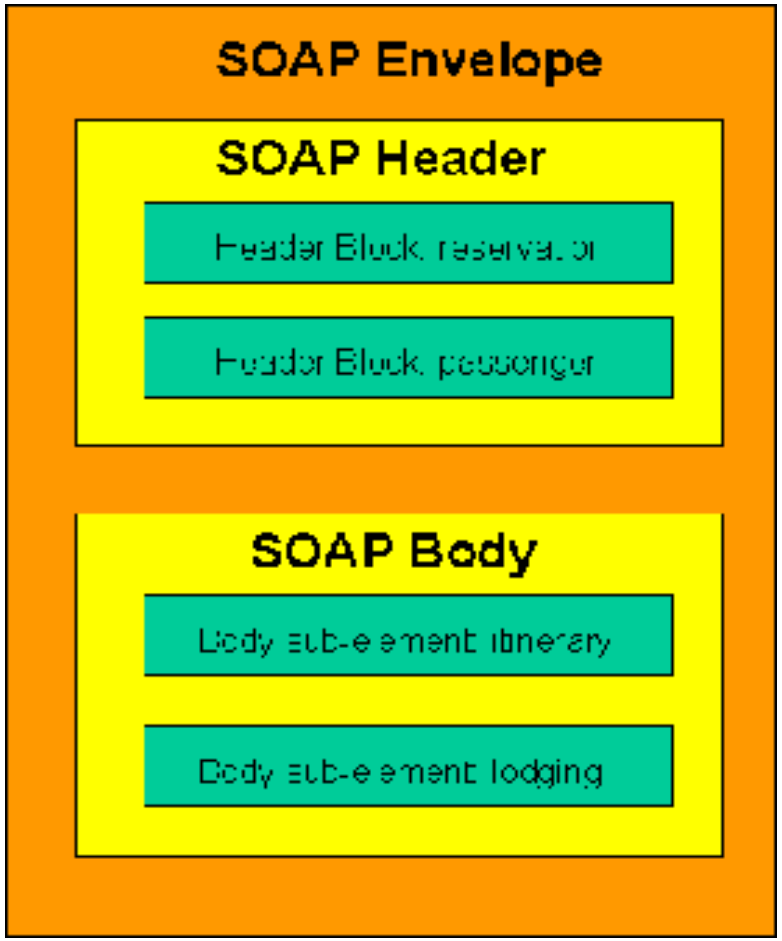
- Decisions made at the time of application design
- Header blocks may be targeted at various nodes that might be encountered along a message's path from a sender to the ultimate recipient
  - Intermediate SOAP nodes may provide value-added services based on data in such headers



Message Structure

Example SOAP Messages

SOAP Message Example: Travel Reservation

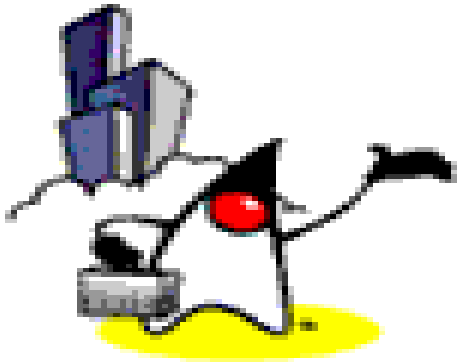


Example1: SOAP Message Travel Reservation (page 1)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">

 <env:Header>

  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
    <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
    <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>

  <n:passenger xmlns:n="http://mycompany.example.com/employees"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
    <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>

 </env:Header>
```

Example1: SOAP Message Travel Reservation (page 2)

```
<env:Body>
  <p:itinerary
    xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>New York</p:departing>
     <p:arriving>Los Angeles</p:arriving>
     <p:departureDate>2001-12-14</p:departureDate>
     <p:departureTime>late afternoon</p:departureTime>
     <p:seatPreference>aisle</p:seatPreference>
   </p:departure>
   <p:return>
     <p:departing>Los Angeles</p:departing>
     <p:arriving>New York</p:arriving>
     <p:departureDate>2001-12-20</p:departureDate>
     <p:departureTime>mid-morning</p:departureTime>
     <p:seatPreference/>
   </p:return>
  </p:itinerary>
  <q:lodging
    xmlns:q="http://travelcompany.example.org/reservation/hotels">
   <q:preference>none</q:preference>
  </q:lodging>
 </env:Body>
</env:Envelope>
```



Message Structure

Quick Namespace Tutorial

XML Namespaces Tutorial

- Used to avoid name collision
- Facilitates grouping of elements
  - I.e: SOAP application knows which elements belong to which namespace
- Can be used as version control scheme
- Syntax
  - Namespace declaration
  - Elements and attributes

XML Namespaces Declaration

- A prefix is associated with URI
- The association is defined as an attribute within an element
  - *xmlns*:prefix
- *xmlns* is Namespaces keyword, prefix is user- defined

```
<classes xmlns:XMLclass=“ http://www.brandeis.edu/rseg-0151-g”>
      <XMLclass:syllabus>
            ...
      </XMLclass:syllabus>
</ classes>
```

SOAP Namespaces Example

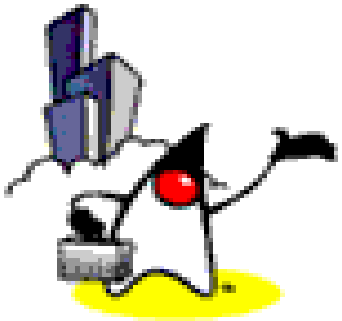
```
<env:Envelope xmlns:env="http://www.w3.org/2001/06/soap-envelope" >
    <env:Body>
        <m:GetLastTradePrice
            env:encodingStyle="http://www.w3.org/2001/06/soap-encoding"
            xmlns:m="http://example.org/2001/06/quotes">
                <symbol>DIS</symbol>
        </m:GetLastTradePrice>
    </env:Body>
</env:Envelope>
```

- env namespace is defined in SOAP
- m namespace is custom namespace

SOAP 1.1 Namespaces URI's

- Envelope
  - http://www.w3.org/2001/06/soap-envelope
  - Used for "version mismatch" check
- Serialization
  - http://www.w3.org/2001/06/soap-encoding
- mustUnderstand fault
  - http://www.w3.org/2001/06/soap-faults
- Upgrade
  - http://www.w3.org/2001/06/soap-upgrade

SOAP Terminology



Protocol Concepts

- SOAP node
- SOAP role
- SOAP binding
- SOAP feature
  - An extension of the SOAP messaging framework: reliability, security, correlation
- SOAP module
  - Realization of SOAP features
- SOAP message exchange pattern
- SOAP application

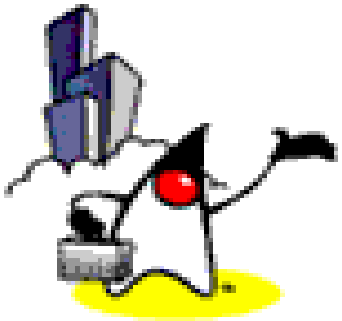
Data Encapsulation Concepts

- SOAP message
- SOAP envelope
- SOAP header
- SOAP header block
- SOAP body
- SOAP fault

Message Sender & Receiver Concepts

- SOAP sender
- SOAP receiver
- SOAP message path
- Initial SOAP sender
- SOAP intermediary
- Ultimate SOAP receiver

SOAP Message Exchange

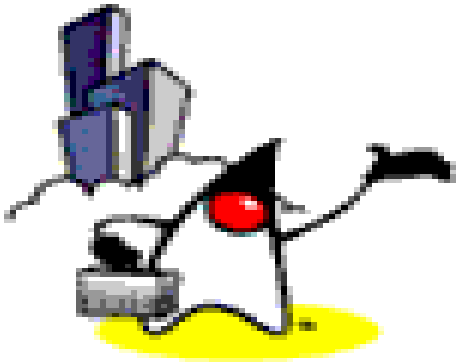


SOAP Exchange Model

- SOAP is a simple messaging framework for transferring information specified in the form of an XML infoset between an initial SOAP sender and an ultimate SOAP receiver
- The more interesting scenarios typically involve multiple message exchanges between these two nodes
  - request and response pattern

Request-Response Pattern

- Conversational message exchange
  - used to exchange XML documents
  - can be multiple message exchange pattern
- RPC (Remote Procedure Call)
  - used when there is a need to model a certain programmatic behavior



SOAP Message Exchange

Conversational Message Exchange

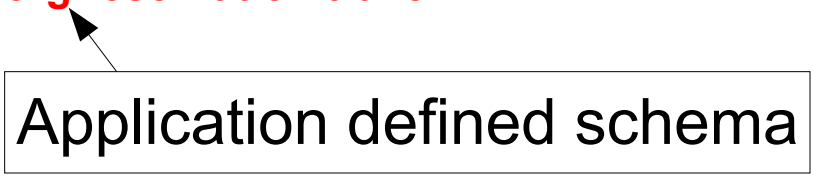
Example2: SOAP Message Response: Travel Reservation (page 1)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:35:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
```

Reference to request message

Example2:SOAP Message Response Travel Reservation (page 2) - Choices of Airport

```
<env:Body>
 <p:itineraryClarification
  xmlns:p="http://travelcompany.example.org/reservation/travel">
  <p:departure>
   <p:departing>
    <p:airportChoices>
     JFK LGA EWR
    </p:airportChoices>
   </p:departing>
  </p:departure>
  <p:return>
   <p:arriving>
    <p:airportChoices>
     JFK LGA EWR
    </p:airportChoices>
   </p:arriving>
  </p:return>
 </p:itineraryClarification>
</env:Body>
</env:Envelope>
```

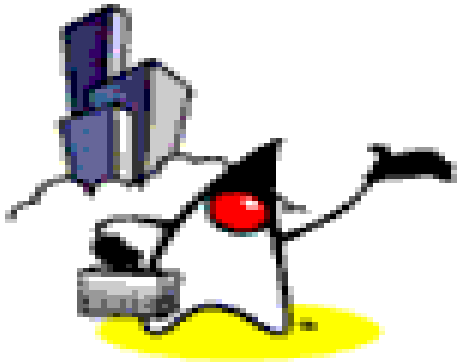


Example3: SOAP Message Travel Reservation (page 1- to be continued)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation
    xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-
    pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-29T13:36:50.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
 </env:Header>
```

Example3: SOAP Message Travel Reservation (page 2) - Selection of Airport

```
<env:Body>
  <p:itinerary
   xmlns:p="http://travelcompany.example.org/reservation/travel">
   <p:departure>
     <p:departing>LGA</p:departing>
   </p:departure>
   <p:return>
     <p:arriving>EWR</p:arriving>
   </p:return>
  </p:itinerary>
 </env:Body>
</env:Envelope>
```



SOAP Message Exchange
Remote Procedure Call

Things Needed for RPC (page 1)

- Address of the target SOAP node
- Procedure or method name
- Identities and values of any arguments and any output parameters and return value
- A clear separation of the arguments used to identify the Web resource which is the actual target for the RPC, as contrasted with those that convey data or control information used for processing the call by the target resource
  - Web resource is specified in the form of URI

Things Needed for RPC (page 2)

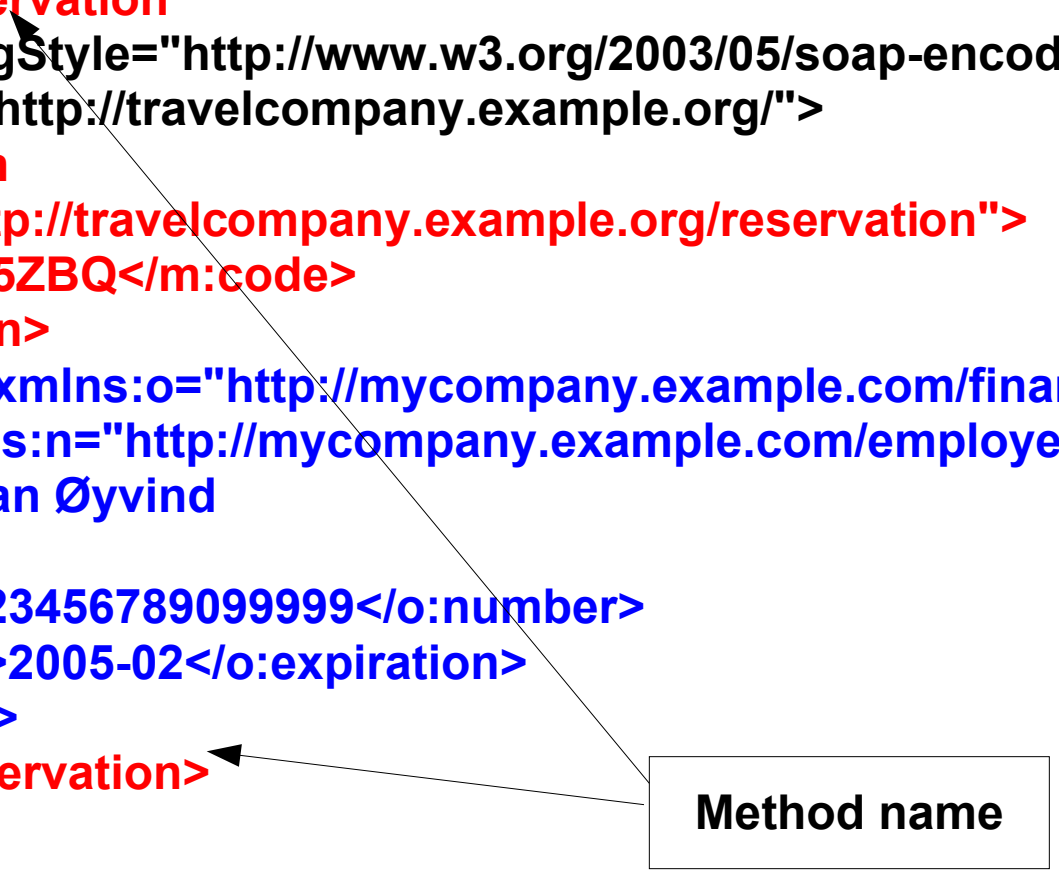
- The message exchange pattern which will be employed to convey the RPC, together with an identification of the so-called "Web Method" to be used
- Optionally, data which may be carried as a part of SOAP header blocks

Example4: SOAP RPC Request (page 1)

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
  <t:transaction
        xmlns:t="http://thirdparty.example.org/transaction"
        env:encodingStyle="http://example.com/encoding"
        env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
```

Example4: SOAP RPC Request (page 2)

```
<env:Body>
 <m:chargeReservation
    env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
      xmlns:m="http://travelcompany.example.org/">
  <m:reservation
   xmlns:m="http://travelcompany.example.org/reservation">
   <m:code>FT35ZBQ</m:code>
  </m:reservation>
  <o:creditCard xmlns:o="http://mycompany.example.com/financial">
   <n:name xmlns:n="http://mycompany.example.com/employees">
        Åke Jógvan Øyvind
   </n:name>
   <o:number>123456789099999</o:number>
   <o:expiration>2005-02</o:expiration>
  </o:creditCard>
 </m:chargeReservation>
</env:Body>
</env:Envelope>
```



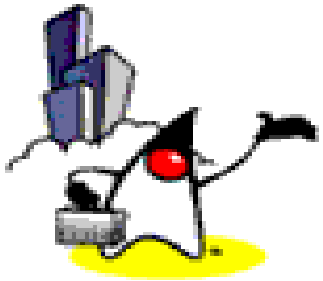
Example5a: RPC response with two output parameters for the call in Example 4

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
    <t:transaction
      xmlns:t="http://thirdparty.example.org/transaction"
       env:encodingStyle="http://example.com/encoding"
        env:mustUnderstand="true">5</t:transaction>
 </env:Header>
 <env:Body>
    <m:chargeReservationResponse
       env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
          xmlns:m="http://travelcompany.example.org/">
      <m:code>FT35ZBQ</m:code>
      <m:viewAt>
       http://travelcompany.example.org/reservations?code=FT35ZBQ
      </m:viewAt>
    </m:chargeReservationResponse>
 </env:Body>
</env:Envelope>
```

Example5b: RPC response with a "return" value and two "out" parameters for the call in Example 4

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
   <t:transaction
     xmlns:t="http://thirdparty.example.org/transaction"
       env:encodingStyle="http://example.com/encoding"
        env:mustUnderstand="true">5</t:transaction>
 </env:Header>
 <env:Body>
   <m:chargeReservationResponse
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
        xmlns:rpc="http://www.w3.org/2003/05/soap-rpc"
          xmlns:m="http://travelcompany.example.org/">
     <rpc:result>m:status</rpc:result>
     <m:status>confirmed</m:status>
     <m:code>FT35ZBQ</m:code>
     <m:viewAt>
      http://travelcompany.example.org/reservations?code=FT35ZBQ
     </m:viewAt>
   </m:chargeReservationResponse>
 </env:Body>
</env:Envelope>
```

SOAP Processing Model



SOAP Processing Model

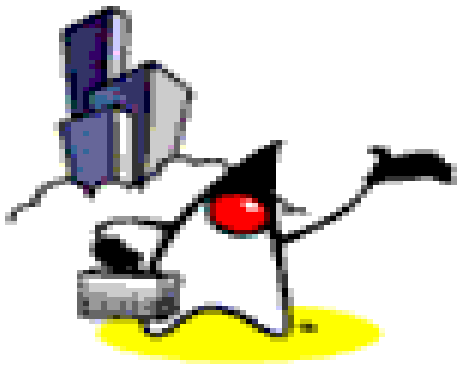
- Describes the (logical) actions taken by a SOAP node on receiving a SOAP message
- Common processing
  - Analyze SOAP-specific elements (elements with SOAP "env" namespace)
  - Check that the SOAP message is syntactically correct
- Attributes of header block
  - Dictates further actions

SOAP Processing Model

- Intermediate nodes can be added to the message path without forcing changes to the messages
  - Message path can be changed
- Messages can be changed without forcing nodes to change their processing model

SOAP Header Block Attributes

- “role” attribute
- “mustUnderstand” attribute
- “relay” attribute



SOAP Processing Model

“role” Attribute in SOAP 1.2 (same as “actor” Attribute in SOAP 1.1)

“role” Attribute (env:role=...)

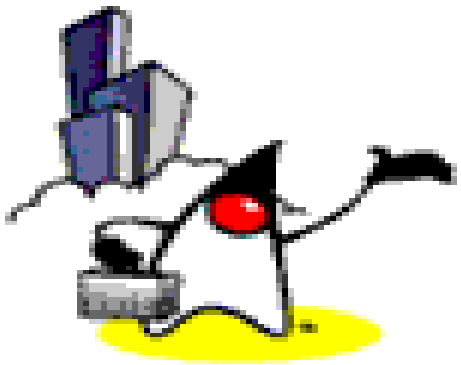
- Identifies the role played by the intended target of that header block
- Custom roles and standard roles
- If absent, the SOAP message is targeted for ultimateReceiver
- Please note that <env:Body> element does not have a “env:role” attribute
  - <env:Body> is always for ultimateReceiver

Standardized Roles

- none
  - "http://www.w3.org/2003/05/soap-envelope/role/none"
- next
  - "http://www.w3.org/2003/05/soap-envelope/role/next"
- ultimateReceiver
  - "http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"

Example7a: SOAP message showing a variety of header blocks

```
<?xml version="1.0" ?>
 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
   <p:oneBlock xmlns:p="http://example.com"
        env:role="http://example.com/Log">
   ...
   </p:oneBlock>
   <q:anotherBlock xmlns:q="http://example.com"
        env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
   ...
   </q:anotherBlock>
   <r:aThirdBlock xmlns:r="http://example.com">
   ...
   </r:aThirdBlock>
  </env:Header>
  <env:Body >
   ...
  </env:Body>
 </env:Envelope>
```



SOAP Processing Model

“mustUnderstand” Attribute

“mustUnderstand” Attribute

- If set to "true", the targeted SOAP node must process the header block (with the semantics described in the header's specification), or generate a fault
- If set to “false” or not present, the targeted SOAP node may process the header block

When to Use "mustUnderstand" Attribute?

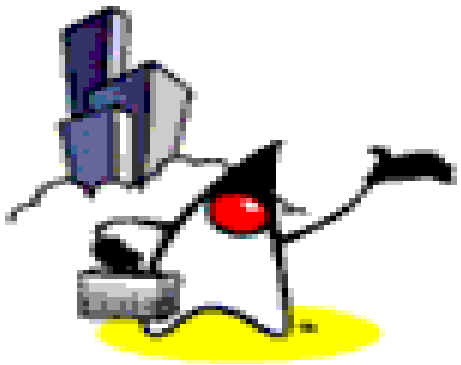
- To ensure SOAP nodes do not ignore mandatory header blocks which are important to overall purpose of the application
  - Example: Making sure all SOAP receivers who play the role of "logger" must perform logging

Processing Header Blocks Might include

- Removing the header (default behavior)
- Reinserting the header with the same or altered value
- Inserting a new header

Example7b: SOAP message with "mustUnderstand" attribute

```
<?xml version="1.0" ?>
 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <p:oneBlock xmlns:p="http://example.com"
          env:role="http://example.com/Log"
          env:mustUnderstand="true">
    ...
    </p:oneBlock>
    <q:anotherBlock xmlns:q="http://example.com"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next">
    ...
    </q:anotherBlock>
    <r:aThirdBlock xmlns:r="http://example.com">
    ...
    </r:aThirdBlock>
  </env:Header>
  <env:Body >
    ...
  </env:Body>
 </env:Envelope>
```



SOAP Processing Model

“relay” Attribute (new from SOAP 1.2)

“relay” Attribute (env:relay=...)

- Newly introduced in SOAP 1.2
- Indicates whether a header block targeted at a SOAP intermediary must be relayed or not if it is not processed
  - Default behavior for a processed or unprocessed header block targeted at a role played by a SOAP intermediary is that it must be removed before the message is relayed
  - You don't want to unknowningly propagate header blocks

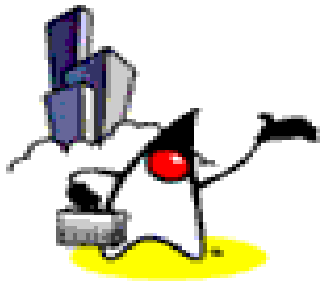
When to Use relay=”true” Attribute?

- Application designer would like to introduce a new feature, manifested through a SOAP header block, targeted at any capable intermediary which might be encountered in the SOAP message path
- Make it vailable to those intermediaries that "understood" it, but ignored and relayed onwards by those that did not
  - Use it with env:mustUnderstand=”false” or env:mustUnderstant not present
  - Use it with env:role=”../next”

Example7c: SOAP message with “relay” attribute

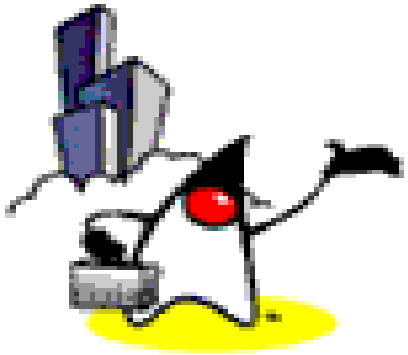
```
<?xml version="1.0" ?>
 <env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Header>
   <p:oneBlock xmlns:p="http://example.com"
        env:role="http://example.com/Log"
           env:mustUnderstand="true">
   ...
   </p:oneBlock>
   <q:anotherBlock xmlns:q="http://example.com"
    env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
    env:relay="true">
   ...
   </q:anotherBlock>
   <r:aThirdBlock xmlns:r="http://example.com">
   ...
   </r:aThirdBlock>
  </env:Header>
  <env:Body >
  ...
  </env:Body>
 </env:Envelope>
```

Protocol Bindings



What is Protocol Binding (SOAP Binding)?

- Specification of how SOAP messages may be passed from one SOAP node to another using an underlying protocol



Protocol Bindings

XML Infoset & Protocol Binding

SOAP Message as XML Infoset

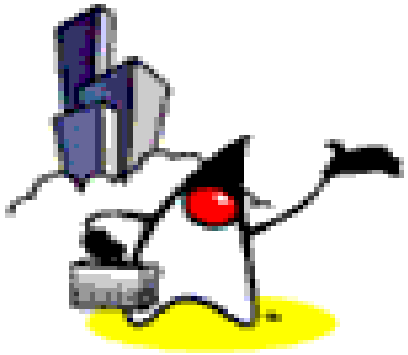
- SOAP message is defined in the form of an XML Infoset
  - in terms of element and attribute information items of an abstract "document" called the env:Envelope
- XML Infoset provides abstract description of its contents
  - Can have different on-the-wire representations
  - (Examples) XML document, DOM object, Custom format

XML Infoset and Protocol Binding

- Any SOAP env:Envelope infoset representation will be made concrete through a protocol binding
  - provides a serialized representation of the infoset that can be conveyed to the next SOAP node in the message path in a manner such that the infoset can be reconstructed without loss of information

Serialized Representation of XML Infoset

- Typically the serialized representation is in the form of XML document
- However, there may be other serialized representation (via different protocol bindings)
  - Example1: between two SOAP nodes over a limited bandwidth interface where an alternative, compressed serialization of the same infoset may be chosen
  - Example2: encrypted structure



Protocol Bindings

XML Features & Protocol Binding

Protocol Binding and SOAP Features

- Protocol binding provides the mechanisms to support features that are needed by a SOAP application
- A feature is a specification of a certain functionality provided by a binding
- A feature description is identified by a URI, so that all applications referencing it are assured of the same semantics
  - reliability, correlation, encryption, etc

SOAP Binding Specification

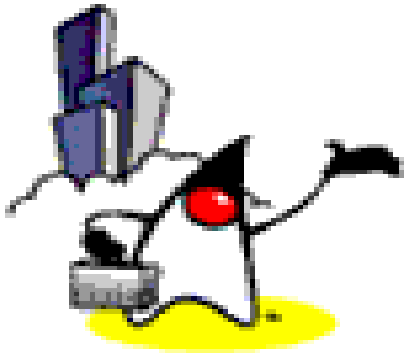
- Describes, among other things, which (if any) features it provides
  - HTTP binding provides request/response co-relation feature
- Some features may be provided natively by the underlying protocol
- If the feature is not available through the binding, it may be implemented within the SOAP envelope, using SOAP header blocks
  - SOAP Module

Example: How do you provide SOAP Message Request/Response Correlation Feature?

- Over UDP
  - either directly by the application or more likely as a part of the SOAP infosets being exchanged
- Over HTTP
  - no further support need be provided at the application or the SOAP level because HTTP binding with SOAP provides request/response correlation feature

SOAP Binding Specification

- Defines message exchange pattern(s) that it supports
- Two message exchange patterns
  - SOAP Request-Response message exchange pattern
    - one SOAP message is exchanged in each direction between two adjacent SOAP nodes
  - SOAP Response message exchange pattern
    - a non-SOAP message (i.e. HTTP request) acting as a request followed by a SOAP message included as a part of the response



Protocol Bindings

SOAP HTTP Binding

SOAP HTTP Binding

- Leverages HTTP request and response model
  - HTTP implicitly correlates its request message with its response message
  - A SOAP application can choose to infer a correlation between a SOAP message sent in the body of a HTTP request message and a SOAP message returned in the HTTP response
- HTTP identifies the server endpoint via a URI
  - URI serves as the identification of a SOAP node at the server

Two Exchange Patterns

- SOAP request-response message exchange pattern
  - Use of the HTTP POST method for conveying SOAP messages in the bodies of HTTP request and response message
- SOAP response message exchange pattern
  - Use of the HTTP GET method in a HTTP request to return a SOAP message in the body of a HTTP response

Which One to Use?

- SOAP request-response message exchange pattern
  - Use it when information resource is manipulated
- SOAP response message exchange pattern
  - Use it when an application is assured that the message exchange is for the purposes of information retrieval
  - Information resource is "untouched" as a result of the interaction
  - Safe and idempotent

SOAP HTTP GET Usage

- Response to a HTTP GET request is a SOAP message in the HTTP response
  - instead of html or xhtml for browser consumption
  - data centric
  - leverages SOAP framework for expressing some application-specific feature through the use of SOAP headers.
- HTTP Accept header is used to indicate the preferred representation of the resource being requested
  - “application/soap+xml”

Example8a: HTTP Get Request

```
GET /travelcompany.example.org/reservations?code=FT35ZBQ  HTTP/1.1
Host: travelcompany.example.org
Accept: text/html;q=0.5, application/soap+xml
```

- The HTTP Accept header is used to indicate the preferred representation of the resource being requested

Example8b: SOAP message returned as a response to the HTTP GET in Example 8a

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
      env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
         env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</m:reference>
   <m:dateAndTime>2001-11-30T16:25:00.000-05:00</m:dateAndTime>
  </m:reservation>
 </env:Header>
 <env:Body>
    ...
 </env:Body>
</env:Envelope>
```

SOAP HTTP POST Usage

- Used for both exchange of general XML data (Document-driven) or RPC
- HTTP Content-type header must be "application/soap+xml"
- The combination of HTTP Post and Host headers represents URI of the resource
  - http://travelcompany.example.org/Reservations
- If error occurs, HTTP 500 "Internal Server Error" is returned along with an embedded SOAP message containing a SOAP fault

Example9: RPC in Example 4 carried in an HTTP POST Request

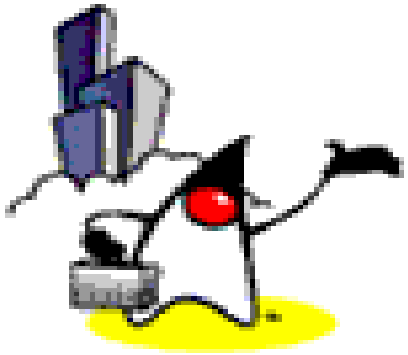
```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
  <t:transaction
        xmlns:t="http://thirdparty.example.org/transaction"
        env:encodingStyle="http://example.com/encoding"
        env:mustUnderstand="true" >5</t:transaction>
 </env:Header>
 <env:Body>
  <m:chargeReservation ...
  ...
 </env:Body>
</env:Envelope>
```

Example10: HTTP Response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
 <env:Header>
      ...
      ...
 </env:Header>
 <env:Body>
      ...
      ...
 </env:Body>
</env:Envelope>
```

Example11: HTTP Response with Failure

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <env:Fault>
     <env:Code>
       <env:Value>env:Sender</env:Value>
       <env:Subcode><env:Value>rpc:BadArguments</env:Value></env:Subcode>
     </env:Code>
     <env:Reason>
      <env:Text xml:lang="en-US">Processing error</env:Text>
      <env:Text xml:lang="cs">Chyba zpracování</env:Text>
     </env:Reason>
     <env:Detail>
      <e:myFaultDetails xmlns:e="http://travelcompany.example.org/faults" >
        <e:message>Name does not match card number</e:message>
        <e:errorcode>999</e:errorcode>
      </e:myFaultDetails>
     </env:Detail>
    </env:Fault>
  </env:Body>
</env:Envelope>
```



Protocol Bindings
Web Architecture
SOAP Usage

URI as Resource Identifier

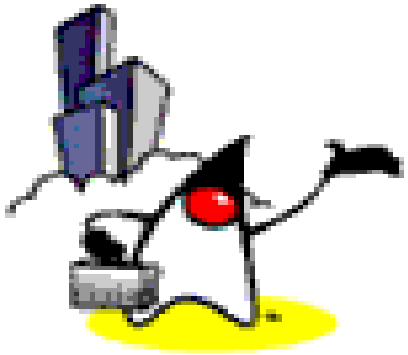
- SOAP services that use the HTTP binding and wish to interoperate with other Web software should use URI's to address all important resources in their service
- For information retrieval, use of a SOAP body to carry the request for the state with an element of the body representing the object in question is strongly discouraged because the resource is not identified by the Request-URI of the HTTP GET

Example12a: Discouraged Practice (for Information Retrieval)

```
POST /Reservations HTTP/1.1
Host: travelcompany.example.org
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Body>
    <m:retrieveItinerary
        env:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
            xmlns:m="http://travelcompany.example.org/">
      <m:reservationCode>FT35ZBQ</m:reservationCode>
    </m:retrieveItinerary>
  </env:Body>
</env:Envelope>
```

Example12: Encouraged Practice (for Information Retrieval)

```
GET /Reservations/itinerary?reservationCode=FT35ZBQ  HTTP/1.1
Host: travelcompany.example.org
Accept: application/soap+xml
```



Protocol Bindings

SOAP Email Binding

SOAP Email Binding

- SOAP messages can be moved as either email text or attachments
- One-way message exchange
- Applications can take advantage of Delivery Status Notification (DSN) of SMTP
  - Separate from message processing at the SOAP layer
  - SOAP Message correlation has to be handled at SOAP level

Example14: SOAP Request message carried in a SMTP message

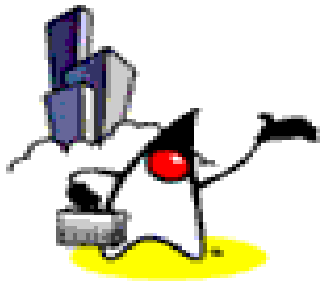
```
From: a.oyvind@mycompany.example.com
To: reservations@travelcompany.example.org
Subject: Travel to LA
Date: Thu, 29 Nov 2001 13:20:00 EST
Message-Id:
    <EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
 ...
 </env:Header>
 <env:Body>
 ...
 </env:Body>
</env:Envelope>
```

Example15: SOAP Response message carried in a SMTP message

From: reservations@travelcompany.example.org
To: a.oyvind@mycompany.example.com
Subject: Which NY airport?
Date: Thu, 29 Nov 2001 13:35:11 EST
Message-Id: <200109251753.NAA10655@travelcompany.example.org>
In-reply-
    to:<EE492E16A090090276D208424960C0C@mycompany.example.com>
Content-Type: application/soap+xml

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
 ...
 </env:Header>
 <env:Body>
 ...
 </env:Body>
</env:Envelope>
```

Using SOAP Intermediaries



Scope of SOAP Specification

- SOAP spec. does not specify how a message path is determined and followed
  - Other specifications will deal with it
- SOAP spec. does describe, though, how a SOAP node should behave if it receives a SOAP message for which it is not the ultimate receiver
- Two types of intermediaries
  - Forwarding intermediaries
  - Active intermediaries

Forwarding Intermediary

- Forwards the SOAP message to another SOAP node
- Things that forwarding intermediary do
  - Remove all processed SOAP header blocks
  - Remove all non-relay'able SOAP header blocks that were targeted at the forwarding node but ignored during processing
  - Retain all relay'able SOAP header blocks that were targeted at the forwarding node but ignored during processing.

Active Intermediary

- Undertake additional processing that can modify the outbound SOAP message in ways not described in the inbound SOAP header blocks
- Example processing
  - security services, annotation services, content manipulation services

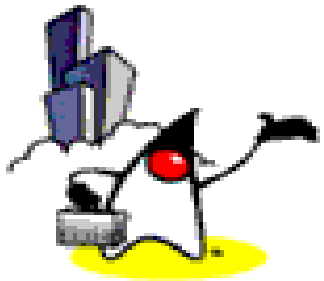
Active Intermediary

- Processing done by active SOAP intermediaries should be detectable by SOAP nodes in the downstream
  - inserting a header block

Example16: SOAP message after an active intermediary has inserted a mandatory header

```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:reservation xmlns:m="http://travelcompany.example.org/reservation"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
        env:mustUnderstand="true">
   <m:reference>uuid:093a2da1-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:reservation>
  <n:passenger xmlns:n="http://mycompany.example.com/employees"
     env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
       env:mustUnderstand="true">
   <n:name>Åke Jógvan Øyvind</n:name>
  </n:passenger>
  <z:travelPolicy
    xmlns:z="http://mycompany.example.com/policies"
      env:mustUnderstand="true">
   <z:class>economy</z:class>
   <z:fareBasis>non-refundable<z:fareBasis>
   <z:exceptions>none</z:exceptions>
  </z:travelPolicy>
</env:Header>
```

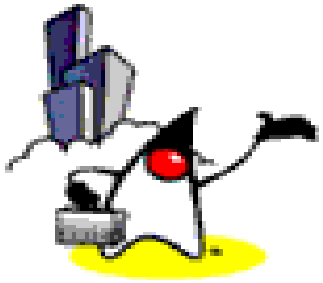
SOAP with Attachments



SOAP 1.1 With Attachments

- Submitted to W3C for basis of XMLP
  - http://www.w3.org/TR/SOAP-attachments
- Uses MIME "multipart/related" as a container for:
  - SOAP envelope
  - Arbitrary "attachments"
- SOAP envelope and payload can reference "attachments" via relative URLs (href) in the SOAP envelope

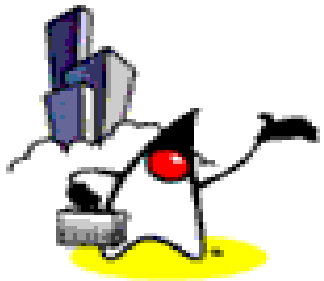
SOAP Security



SOAP Security

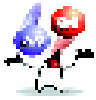
- Transport level security
  - SOAP over HTTP/S
- SOAP message level security
  - WS-Security
  - ebXML Message Service

Quick Overview on Java APIs for SOAP

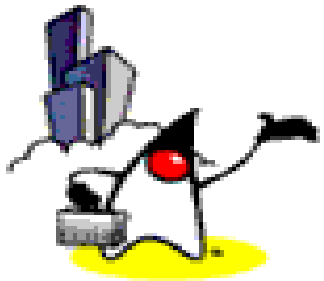


Java Programming API for SOAP

- JAX-RPC (JSR-101)
  - Java™ API for XML-based RPC
  - Core Java API for SOAP programming
  - Uses SAAJ underneath
- SAAJ (JSR-67)
  - Java™ API for SOAP Message creation and manipulation
- JAXM (old JSR-67)



Changes from SOAP 1.1 to SOAP 1.2



Changes: Document Structure

- Document structure
  - SOAP 1.2 has been rewritten in terms of XML infosets, and not as serializations of the form <?xml....?> required by SOAP 1.1

Changes: Additional or Changed Syntax (page 1)

- In the SOAP 1.2 infoset-based description, the env:mustUnderstand attribute in header elements takes the (logical) value "true" or "false" (instead of 1 or 0)
- SOAP 1.2 provides a new fault code DataEncodingUnknown.
- The various namespaces defined by the two protocols are of course different.
- SOAP 1.2 replaces the attribute env:actor with env:role but with essentially the same semantics.

Changes: Additional or Changed Syntax (page 2)

- SOAP 1.2 defines a new attribute, env:relay, for header blocks to indicate if unprocessed header blocks should be forwarded
- SOAP 1.2 defines two new roles, "none" and "ultimateReceiver", together with a more detailed processing model on how these behave
- SOAP 1.2 replaces "client" and "server" fault codes with "Sender" and "Receiver"

Changes: SOAP HTTP Binding

- In the SOAP 1.2 HTTP binding, the SOAPAction HTTP header defined in SOAP 1.1 has been removed
- In the SOAP 1.2 HTTP binding, the Content-type header should be "application/soap+xml" instead of "text/xml" as in SOAP 1.1
- Support of the HTTP extensions framework has been removed from SOAP 1.2
- SOAP 1.2 provides an additional message exchange pattern which may be used as a part of the HTTP binding that allows the use of HTTP GET for safe and idempotent information retrievals

Changes: RPC

- SOAP 1.2 provides a rpc:result element assessor for RPCs
- SOAP 1.2 provides several additional fault codes in the RPC namespace
- SOAP 1.2 offers guidance on a Web-friendly approach to defining RPCs where the procedure's purpose is purely "safe" informational retrieval

Changes: SOAP Encoding

- An abstract data model based on a directed edge labeled graph has been formulated for SOAP 1.2
  - SOAP 1.2 encodings are dependent on this data model
- The SOAP RPC conventions are dependent on this data model, but have no dependencies on the SOAP encoding
- Support of the SOAP 1.2 encodings and SOAP 1.2 RPC conventions are optional

Version Handling

- SOAP 1.1 node receiving 1.2 message
  - generates SOAP version mismatch SOAP fault
- SOAP 1.2 node receiving 1.1 message
  - may process the message as a SOAP/1.1 message (if supported)
  - generate a version mismatch SOAP fault based on a SOAP/1.1 message construct
  - SOAP fault SHOULD include an Upgrade SOAP header block

Example: SOAP Version 1.2 node generating a SOAP/1.1 version mismatch fault message

```
<?xml version="1.0" ?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <env:Upgrade>
   <env:SupportedEnvelope qname="ns1:Envelope"
          xmlns:ns1="http://www.w3.org/2003/05/soap-envelope"/>
  </env:Upgrade>
 </env:Header>
 <env:Body>
  <env:Fault>
   <faultcode>env:VersionMismatch</faultcode>
   <faultstring>Version Mismatch</faultstring>
  </env:Fault>
 </env:Body>
</env:Envelope>
```

Passion!