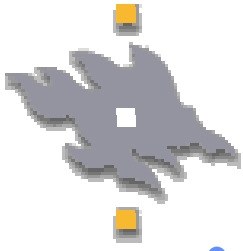


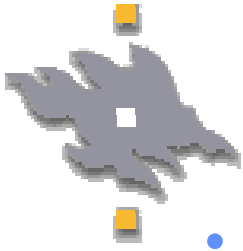
# SOAP

Suresh Chande



# What is SOAP?

- SOAP was formerly known as Simple Object Access Protocol
- SOAP is a simple and flexible messaging framework for transferring information specified in the form of an XML infoset between an initial SOAP sender and an ultimate SOAP receiver. SOAP **does not define** application semantics but **defines** a mechanism to express application messaging semantics.
- ”SOAP defines a simple and lightweight mechanism for exchanging structured and typed information between peers over the web in a decentralized, distributed environment using XML.”



## What is SOAP? [contd..]

- A lightweight protocol for exchange of information in a decentralized, distributed environment
- An XML based protocol that consists of three parts:
  - an envelope that defines a framework for describing what is in a message and how to process it,
  - a set of encoding rules for expressing instances of application-defined datatypes,
  - and a convention for representing remote procedure calls / document objects and corresponding responses.



# SOAP Terminology

- **Node:** Enforcing the rules that govern the exchange of SOAP messages. It accesses the services provided by the underlying protocols through one or more SOAP bindings.
- **Role:** A SOAP node's expected function in processing a message(next, none, ultimate Receiver)
- **Binding:** Formal set of rules for carrying a SOAP message within or on top of another protocol (underlying protocol) for the purpose of exchange
- **Feature:** Extension of the SOAP messaging framework
- **Module:** Specification that contains the combined syntax and semantics of SOAP header blocks

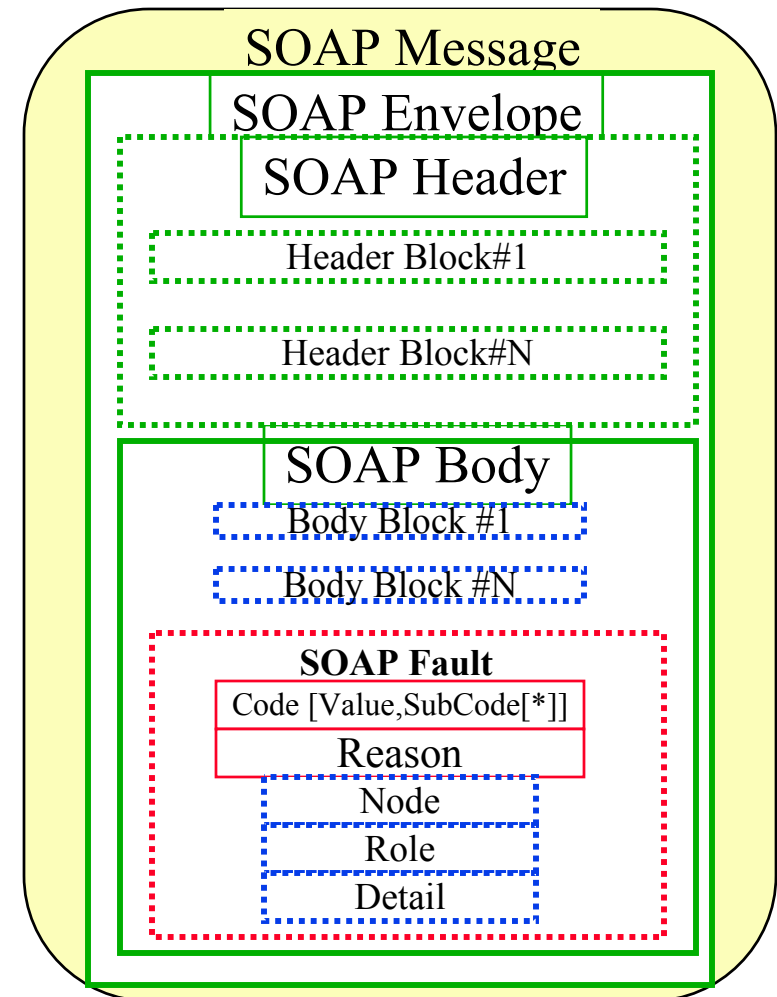


# SOAP Terminology

- **MEP:** Exchange of SOAP messages between SOAP nodes enabled by one or more underlying SOAP protocol bindings
- **SOAP Application:** software entity that produces, consumes or otherwise acts upon SOAP messages in a manner conforming to the SOAP processing model
- **Message Path:** Set of SOAP nodes through which a single SOAP message passes

# SOAP Message Structure (1.2)

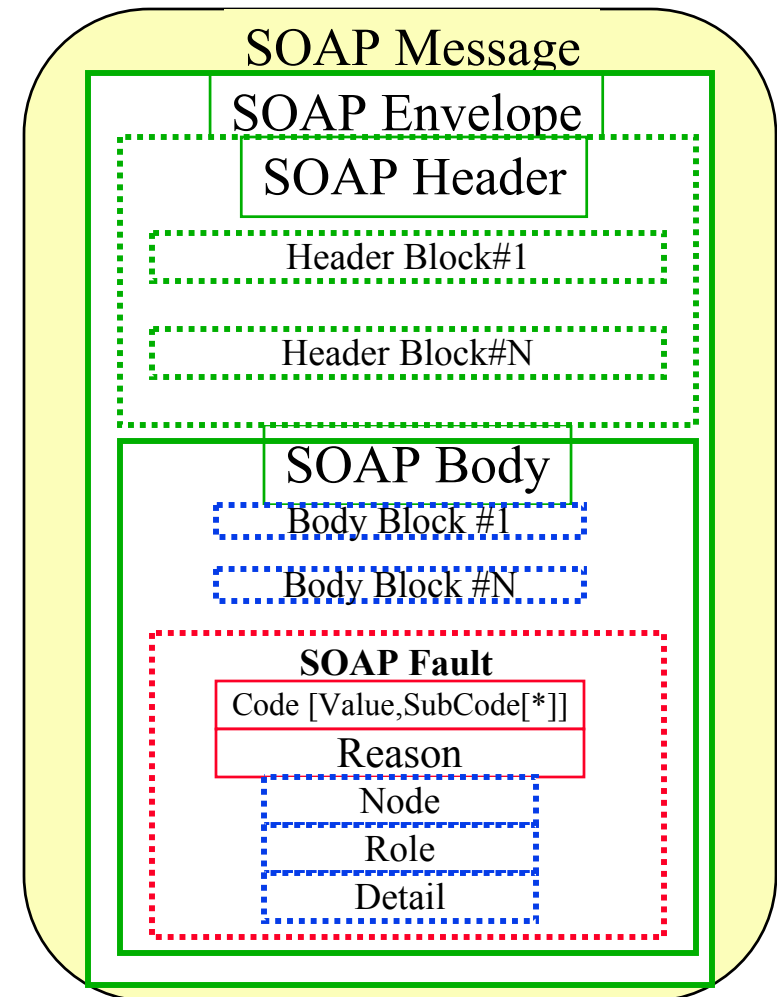
- **Envelope:** This is top level root element of a SOAP Message, which contains the Header and Body element.
- **Header:** A collection of zero or more SOAP header blocks each of which might be targeted at any SOAP receiver within the SOAP message path.
- **Body:** A collection of zero or more *element information items* targeted at an ultimate SOAP receiver in the SOAP message path

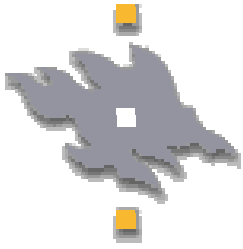


# SOAP Message Structure (1.2)

- **Fault:** A SOAP *element information item* which contains fault information generated by a SOAP node
  - **Code:** contains a highlevel code classifying the nature of the fault
  - **Reason:** Intended for human readable text
  - **Node:** SOAP node on the SOAP message path caused the fault
  - **Role:** The role the node was operating in at the point the fault
  - **Detail:** intended for carrying application specific error information

**Note:** All the above SOAP1.2 Message elements must be defined by the Namespace : <http://www.w3.org/2003/05/soap-envelope>





# SOAP 1.2 Schema document

A look into the SOAP XML Schema document structure



# SOAP's XML Schema

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="http://www.w3.org/2003/05/soap-envelope"
  targetNamespace="http://www.w3.org/2003/05/soap-envelope"
  elementFormDefault="qualified" >

  <xs:import namespace="http://www.w3.org/XML/1998/namespace" />

  <!-- Envelope, header and body -->
  <xs:element name="Envelope" type="tns:Envelope" />
  <xs:complexType name="Envelope" >
    <xs:sequence>
      <xs:element ref="tns:Header" minOccurs="0" />
      <xs:element ref="tns:Body" minOccurs="1" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other" processContents="lax" />
  </xs:complexType>
```

CONTINUED ON NEXT SLIDE....



# SOAP's XML Schema Contd..

```
<xs:element name="Header" type="tns:Header" />
<xs:complexType name="Header" >
  <xs:annotation>
    <xs:documentation>
      Elements replacing the wildcard MUST be namespace qualified,
      but can be in the targetNamespace
    </xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax"
    minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

Extensible Framework



CONTINUED ON NEXT SLIDE...



## SOAP's XML Schema Contd..

```
<xs:element name="Body" type="tns:Body" />
  <xs:complexType name="Body" >
    <xs:sequence>
      <xs:any namespace="##any" processContents="lax"
minOccurs="0" maxOccurs="unbounded" />
    </xs:sequence>
    <xs:anyAttribute namespace="##other"
processContents="lax" />
  </xs:complexType>
```

CONTINUED ON NEXT SLIDE....



# SOAP's XML Schema Contd..

## ATTRIBUTES:

```
<xs:attribute name="mustUnderstand" type="xs:boolean"  
  default="0" />
```

```
<xs:attribute name="relay" type="xs:boolean" default="0"  
 />
```

```
<xs:attribute name="role" type="xs:anyURI" />
```

```
<xs:attribute name="encodingStyle" type="xs:anyURI" />
```

CONTINUED ON NEXT SLIDE....



# SOAP's XML Schema Contd..

## FAULT

```
<xs:element name="Fault" type="tns:Fault" />
  <xs:complexType name="Fault" final="extension" >
    <xs:annotation>
      <xs:documentation>
        Fault reporting structure
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Code" type="tns:faultcode" />
      <xs:element name="Reason" type="tns:faultreason" />
      <xs:element name="Node" type="xs:anyURI" minOccurs="0" />
      <xs:element name="Role" type="xs:anyURI" minOccurs="0" />
      <xs:element name="Detail" type="tns:detail" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
```

CONTINUED ON NEXT SLIDE...



# SOAP's XML Schema Contd..

```
<xs:complexType name="faultreason" >
  <xs:sequence>
    <xs:element name="Text" type="tns:reasontext"
      minOccurs="1" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="reasontext" >
  <xs:simpleContent>
    <xs:extension base="xs:string" >
      <xs:attribute ref="xml:lang" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```



## SOAP's XML Schema Contd..

```
<xs:complexType name="faultcode">
  <xs:sequence>
    <xs:element name="Value" type="tns:faultcodeEnum" />
    <xs:element name="Subcode" type="tns:subcode"
      minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

```
<xs:simpleType name="faultcodeEnum">
  <xs:restriction base="xs:QName">
    <xs:enumeration value="tns:DataEncodingUnknown" />
    <xs:enumeration value="tns:MustUnderstand" />
    <xs:enumeration value="tns:Receiver" />
    <xs:enumeration value="tns:Sender" />
    <xs:enumeration value="tns:VersionMismatch" />
  </xs:restriction>
</xs:simpleType>
```

**CONTINUED ON NEXT SLIDE...**



## SOAP's XML Schema Contd..

```
<xs:complexType name="subcode">
  <xs:sequence>
    <xs:element name="Value" type="xs:QName"/>
    <xs:element name="Subcode" type="tns:subcode" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="detail">
  <xs:sequence>
    <xs:any namespace="##any" processContents="lax" minOccurs="0"
      maxOccurs="unbounded" />
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax" />
</xs:complexType>
```

CONTINUED ON NEXT SLIDE...



## SOAP's XML Schema Contd..

```
<xs:element name="NotUnderstood"  
  type="tns:NotUnderstoodType" />
```

```
<xs:complexType name="NotUnderstoodType" >  
  <xs:attribute name="qname" type="xs:QName" use="required" />  
</xs:complexType>
```

```
<xs:complexType name="SupportedEnvType" >  
  <xs:attribute name="qname" type="xs:QName" use="required" />  
</xs:complexType>
```

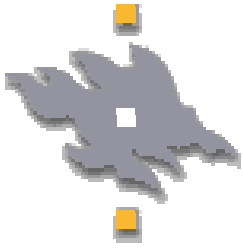
CONTINUED ON NEXT SLIDE...



## SOAP's XML Schema Contd..

```
<xs:complexType name="SupportedEnvType" >
  <xs:attribute name="qname" type="xs:QName"
    use="required" />
</xs:complexType>

<xs:element name="Upgrade" type="tns:UpgradeType" />
<xs:complexType name="UpgradeType" >
  <xs:sequence>
    <xs:element name="SupportedEnvelope"
      type="tns:SupportedEnvType" minOccurs="1"
      maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
</xs:schema>
```



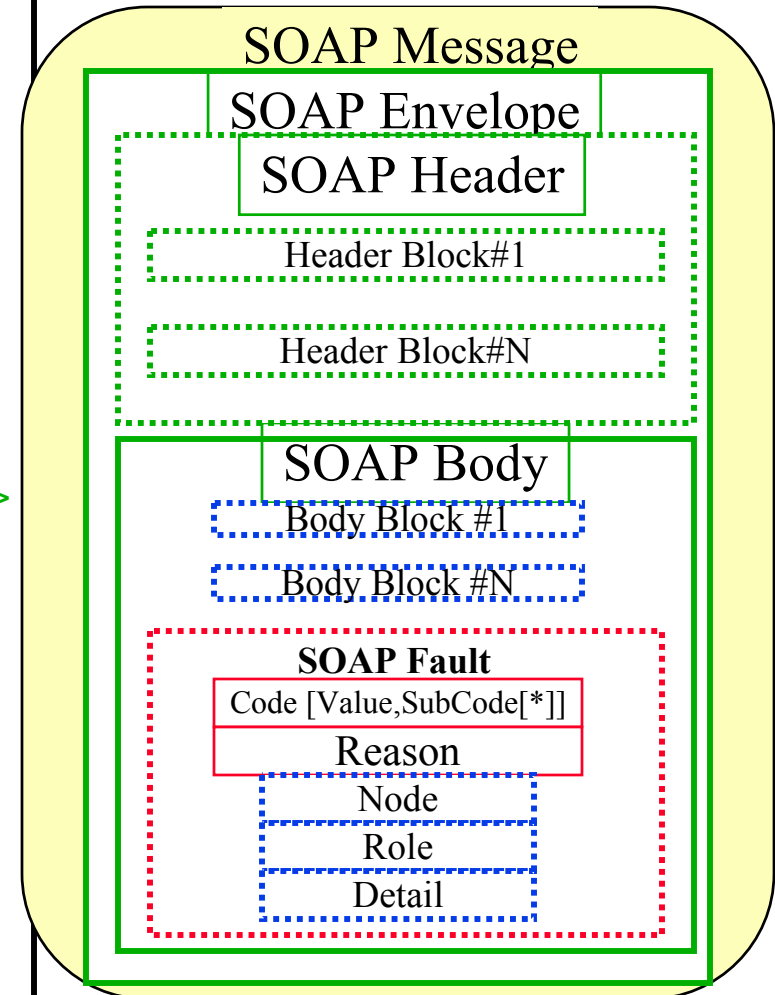
# SOAP Schema's- Instance Documents

- A few examples of SOAP Messages

# Example: SOAP Request Message

```
<?xml version=1.0 encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
<m:authentication
  xmlns:m="http://www.mobileservices.org/identity"
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next "
  env:mustUnderstand="true">
  <m:username>suresh</m:username>
  <m:password>password</m:password>
</m:authentication>
</env:Header>
<env:Body>
  <d:sendMessage xmlns:d="http://www.mobileservices.org/delivery">
    <d:message>
      <d:ID>uuid:093c9kpa2-c981-782b-ws6q-eggg63et9n2f</d:ID>
      <d:Recipients>
        <d:MSISDN>+3585048372980</d:MSISDN>
        <d:MSISDN>+3585048372981</d:MSISDN>
        <d:MSISDN>+3585048372982</d:MSISDN>
        <d:Email>suresh.chande@nokia.com</d:Email>
      </d:Recipients>
    </d:message>
  </d:sendMessage>
</env:Body>
</env:Envelope>
```

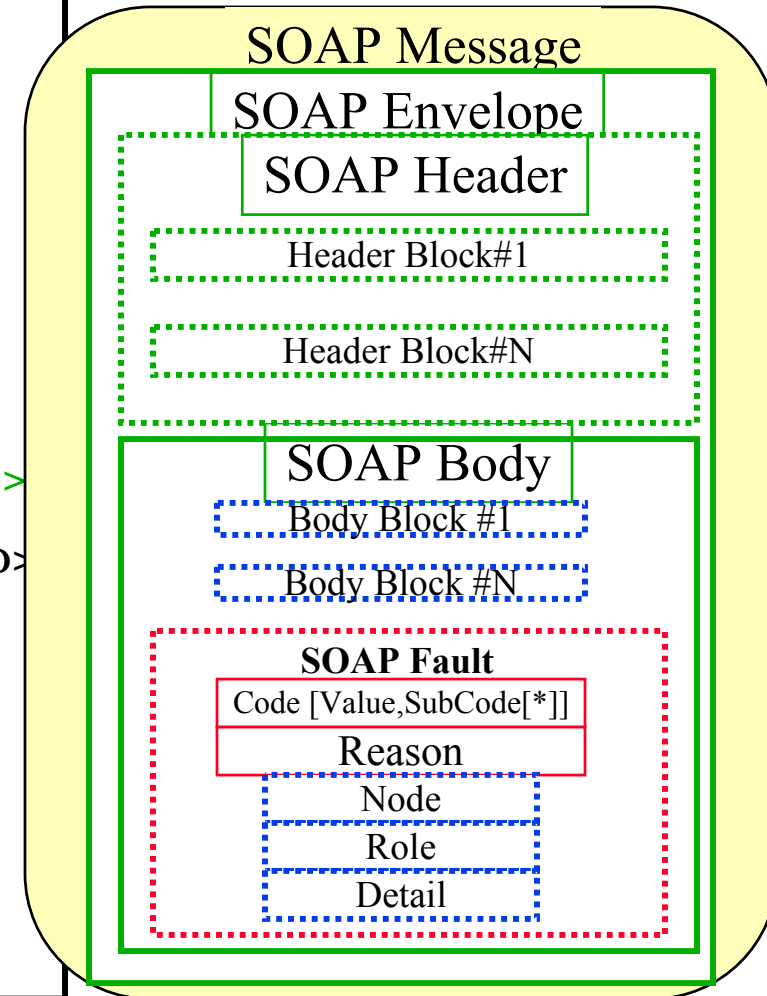
## SOAP Request



# Example: SOAP Response Message

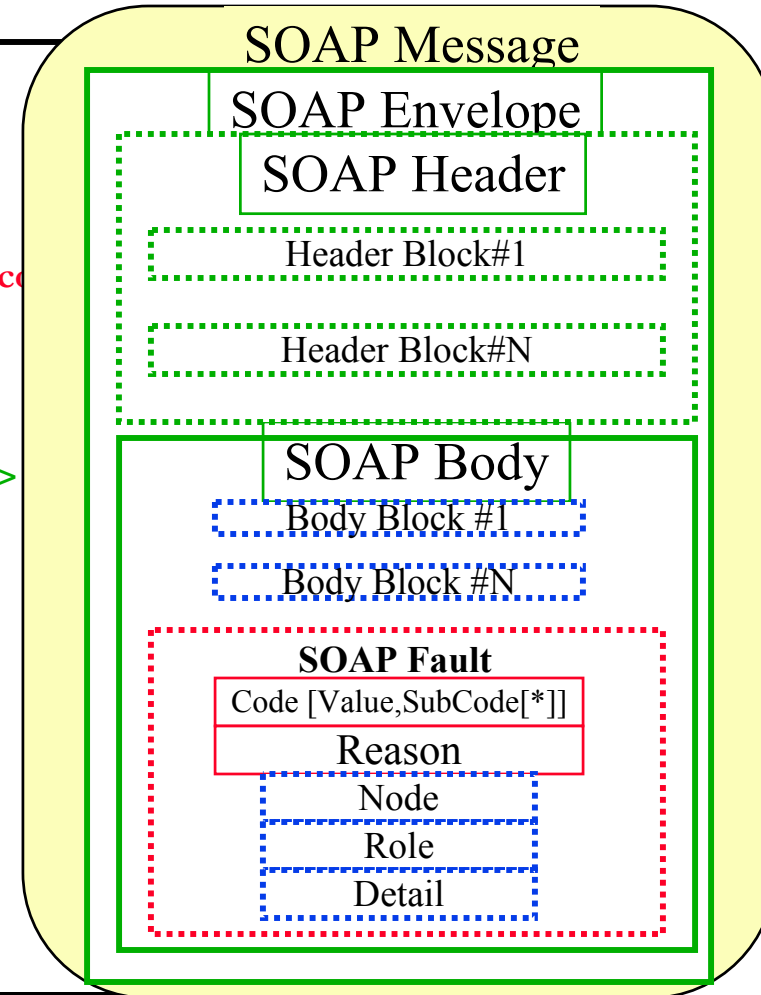
```
<?xml version=1.0 encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
<env:Header>
<m:acknowledgement
  xmlns:m="http://www.mobileservices.org/identity"
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
  env:mustUnderstand="true">
  <m:receipt>1w12le34-1y9w42-13ws-3w2432</ m:receipt>
</m:acknowledgement>
</env:Header>
<env:Body>
  <d:messageDeliveredResponse
    xmlns:d="http://www.mobileservices.org/delivery">
    <d:message>
      <d:RefID>uuid:093c9kpa2-c981-782b-ws6q-eggg63et9n2f</d:RefID>
      <d:Recipients>
        <d:MSISDN>+3585048372980</d:MSISDN>
        <d:MSISDN>+3585048372981</d:MSISDN>
        <d:MSISDN>+3585048372982</d:MSISDN>
        <d:Email>suresh.chande@nokia.com</d:Email>
      </d:Recipients>
    </d:message>
  </d:messageDeliveredResponse>
</env:Body>
</env:Envelope>
```

## SOAP Response

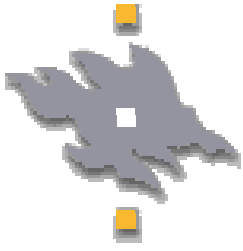


# Example: SOAP Fault Response Message

```
<?xml version=1.0 encoding="UTF-8"?>
<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>m:MessageTimeout</env:Value> </env:Subcode>
</env:Code>
<env:Reason>Sender Timeout</env:Reason>
<env:Detail>
<d:messageDeliveryFailure
xmlns:d="http://www.mobileservices.org/delivery">
<d:message>
<d:RefID>uuid:093c9kpa2-c981-782b-ws6q-eggg63et9n2f</d:RefID>
<d:FailedRecipients>
<d:MSISDN>+3585048372980</d:MSISDN>
<d:MSISDN>+3585048372982</d:MSISDN>
</d:FailedRecipients>
</d:message>
</d:messageDeliveryFailure>
</env:Detail>
</env:Fault>
</env:Body>
</env:Envelope>
```



## SOAP Fault Response



# SOAP Invocation Styles

SOAP specification supports two styles of handling the SOAP messages received at the Nodes, namely RPC and Document style

- **Remote Procedural Call (RPC) Style:**
  - This enables the SOAP engine to directly invoke the underlying software system
- **Document Style:**
  - This enables the SOAP engine to deliver the Message to the corresponding Message Handler.

Note: The closer the system is to the Document Style SOAP encoding the closer is to loose coupling structure the reduced inter-operability issues

# SOAP - Document Style

## SOAP - Document Style

POST /Delivery/v1.0/deliver  
HOST: www.nokia.com  
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:authentication xmlns:m="http://www.mobileservices.org/identity"  
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next"  
  env:mustUnderstand="true">  
  <m:username>suresh</m:username>  
  <m>Password>mypassword</m:password>  
</m:authentication>  
</env:Header>  
<env:Body>  
  <d:message xmlns:d="http://www.mobileservices.org/delivery">  
    <d:ID>uuid:093c9kpa2-c981-782b-ws6q-eggg63et9n2f</d:ID>  
    <d:Recipients>  
      <d:MSISDN>+3585048372980</d:MSISDN>  
      <d:MSISDN>+3585048372981</d:MSISDN>  
      <d:MSISDN>+3585048372980</d:MSISDN>  
      <d:Email>firstname.last@nokia.com</d:Email>  
    </d:Recipients>  
  </d:message>  
</env:Body>  
</env:Envelope>
```

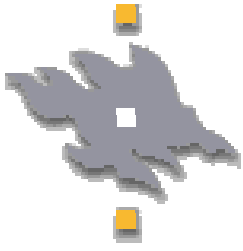
# SOAP - RPC Style

## SOAP - RPC Style

POST /Delivery/v1.0/deliver  
HOST: www.nokia.com  
Content-Type; application/soap+xml; charset="utf-8"  
Content-Length:nnnn

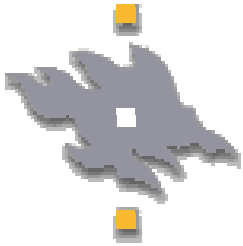
```
<?xml version=1.0>  
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">  
<env:Header>  
<m:authentication xmlns:m="http://www.example.org/identity"  
  env:role="http://www.w3.org/2003/05/soap-envelope/role/next"  
  env:mustUnderstand="true">  
  <m:username>suresh</m:username>  
  <m>Password>mypassword</m:password>  
</m:authentication>  
</env:Header>  
<env:Body>  
  <d:sendMessage xmlns:d="http://www.mobileservices.org/delivery">  
    <d:message>  
      <d:ID>uuid:093c9kpa2-c981-782b-ws6q-eggg63et9n2f</d:ID>  
      <d:Recipients>  
        <d:MSISDN>+3585048372980</d:MSISDN>  
        <d:MSISDN>+3585048372981</d:MSISDN>  
        <d:MSISDN>+3585048372980</d:MSISDN>  
        <d:Email>firstname.last@nokia.com</d:Email>  
      </d:Recipients>  
    </d:message>  
  </d:sendMessage>  
</env:Body>  
</env:Envelope>
```

Will invoke a sendMessage()  
RPC on the ultimate Recipient

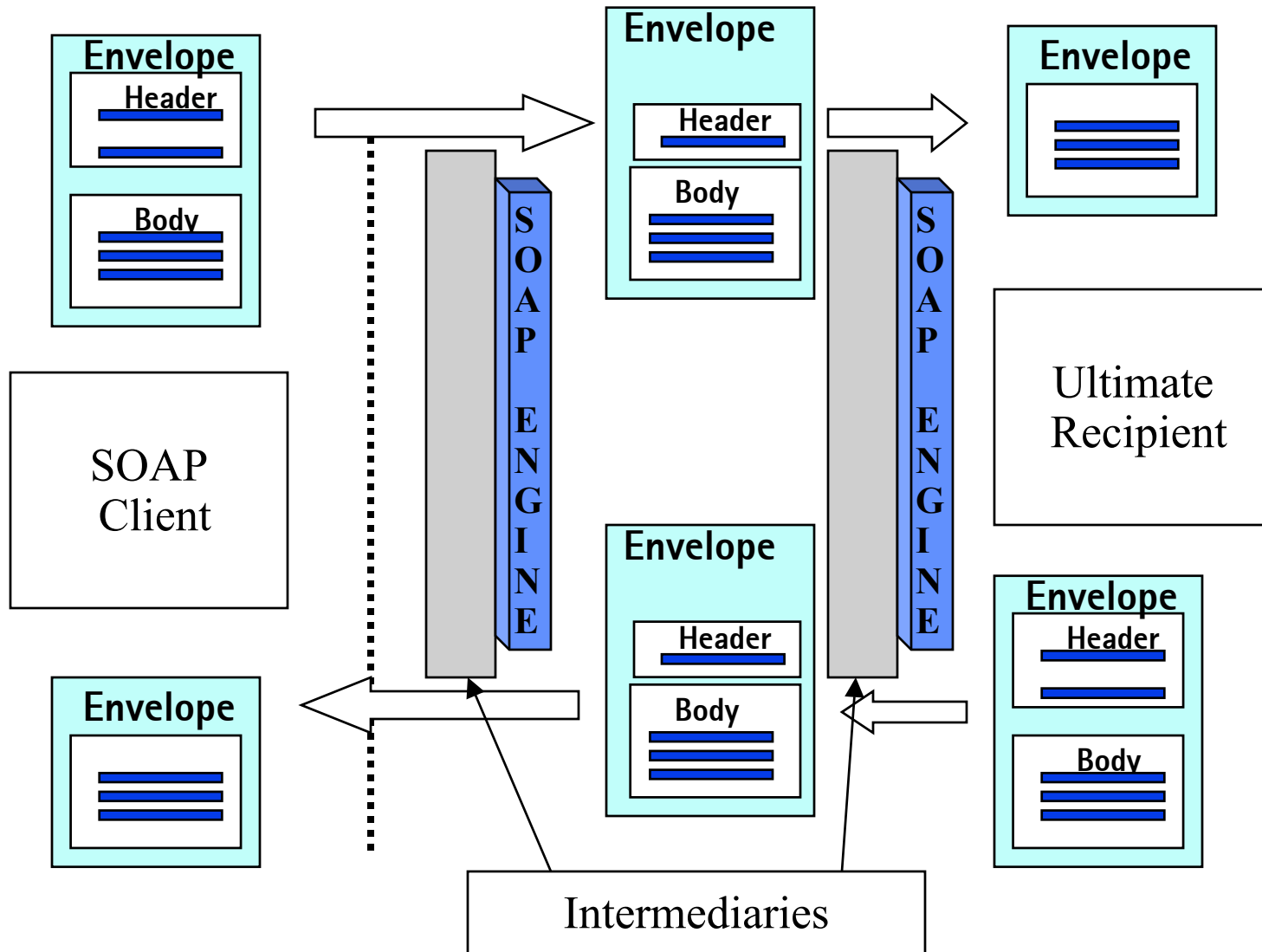


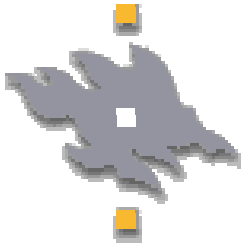
# SOAP Processing Model

- **Determine the set of roles in which the node is to act.** The contents of the SOAP envelope, SOAP header blocks and the SOAP body are inspected in making such determination.
- **Identify all header blocks targeted at the node that are mandatory.**
- **If one or more of the SOAP header blocks identified are not understood by the node then generate a single SOAP fault with the Value of Code set to "env:NotUnderstood".** If such a fault is generated, any further processing **MUST NOT** carry on.
- **Process all mandatory SOAP header blocks** targeted at the node and, in the case of an ultimate SOAP receiver, the SOAP body. A SOAP node **MAY** also choose to process non-mandatory SOAP header blocks targeted at it.



# SOAP - Intermediaries





# SOAP1.1

Don Box, DevelopMentor

David Ehnebuske, IBM

Gopal Kakivaya, Microsoft

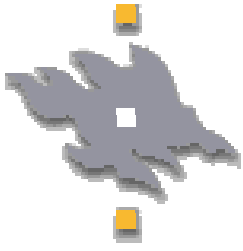
Andrew Layman, Microsoft

Noah Mendelsohn, Lotus Development Corp.

Henrik Frystyk Nielsen, Microsoft

Satish Thatte, Microsoft

Dave Winer, UserLand Software, Inc.



# Namespaces

- **SOAP1.1 Namespaces:**
  - SOAP-ENV => "<http://schemas.xmlsoap.org/soap/envelope/>"
  - SOAP-ENC=> "<http://schemas.xmlsoap.org/soap/encoding/>"
  - xsi => "<http://www.w3.org/1999/XMLSchema-instance>" (1999->2001)
  - xsd => "<http://www.w3.org/1999/XMLSchema>" (1999->2001)
- SOAP 1.2 Namespaces:
  - SOAP-ENV => "http://www.w3.org/2003/05/soap-envelope"
  - SOAP-ENC=> "http://www.w3.org/2003/05/soap-encoding"



# SAMPLE SOAP1.2 Message with HTTP binding

**POST /Stockquote HTTP/1.1**

**HOST: www.stockquote.com**

**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:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <env:Body>
    <m:getInfoRequest xmlns:m="http://www.myexample.com/1_0/InformationService/">
      <m:inputParameter>20</m:inputParameter>
    </m:getInfoRequest>
  </env:Body>
</env:Envelope>
```

**HTTP/1.1 200 OK**

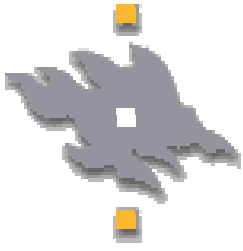
**Content-Type: application/soap+xml; charset="utf-8"**

**Content-Length: nnnn**

<<? xml version="1.0">

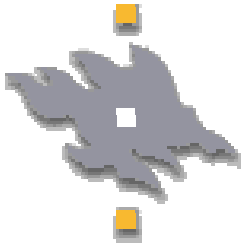
```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  <env:Body>
    <m:getInfoResponse xmlns:m="http://www.myexample.com/1_0/InformationService/">
      <m:Value>10000</m:value>
    </m:getInfoResponse>
  </env:Body>
</env:Envelope>
```

Department of Computer Science



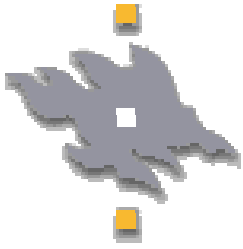
# SOAP Binding with HTTP

- SOAP HTTP binding with or without using the HTTP Extension Framework
- Carrying SOAP in HTTP does not mean that SOAP overrides existing semantics of HTTP but rather that the semantics of SOAP over HTTP maps naturally to HTTP semantics.
- SOAP naturally follows the HTTP request/response message model providing SOAP request parameters in a HTTP request and SOAP response parameters in a HTTP response
- SOAP intermediaries are NOT the same as HTTP intermediaries. HTTP intermediary addressed with the HTTP Connection header field cannot be expected to inspect or process the SOAP entity body carried in the HTTP request
- HTTP applications MUST use the media type “application/soap+xml” for SOAP based messages



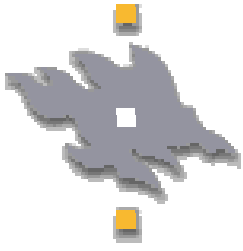
# SOAP Exchange Model

- SOAP messages are fundamentally one-way transmissions
- Messages are often combined to implement patterns such as request/response.
- Messages are often exchanged and expressed via a Message Exchange Patterns :
  - » One-Way
  - » Request-Response
  - » Solicit-Response
  - » Notifications
  - » Peer-Peer



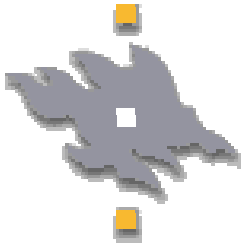
# SOAP Processing Model

- Messages are routed along a so-called "message path", which allows for processing at one or more intermediate nodes in addition to the ultimate destination.
- Identify all parts of the SOAP message relevant to the receiver
- Verify and process the relevant parts to the receiver accordingly. Must process all parts set mandatory.
- If SOAP application is not the ultimate destination of the message then remove all parts identified and processed in the previous test before forwarding the message.



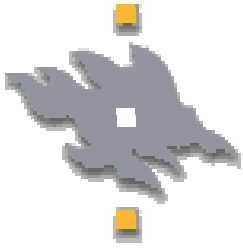
# SOAP Must / Nots

- **MUST's:**
  - A SOAP application **MUST** be able to process SOAP namespaces in messages that it receives
  - **MUST** discard messages that have incorrect namespaces
  - **MAY** process SOAP messages without SOAP namespaces as though they had the correct SOAP namespaces
- **Nots:**
  - **MUST NOT** contain a Document Type Declaration
  - **MUST NOT** contain Processing Instructions.



# SOAP1.1 Encoding

- SOAP encodingStyle global attribute can be used to indicate the serialization rules used in a SOAP message
- This attribute *MAY* appear on any element, and is scoped to that element's contents and all child elements not themselves containing such an attribute,
- The attribute value is an ordered list of one or more URIs identifying the serialization rule or rules that can be used to deserialize the SOAP message indicated in the order of most specific to least specific.



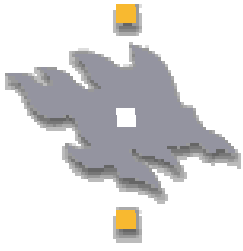
# SOAP Header Processing

- A SOAP message travels from the originator to the ultimate destination, potentially by passing through a set of SOAP intermediaries along the message path.
- SOAP intermediary is an application that is capable of both receiving and forwarding SOAP messages.
- Intermediaries as well as the ultimate destination are identified by a URI
- A recipient receiving a header element **MUST NOT** forward that header element to the next application in the SOAP message path.
- The recipient **MAY** insert a similar header element



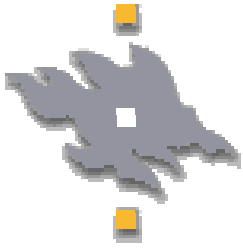
# SOAP Header Processing

- SOAP actor global attribute can be used to indicate the recipient of a header element
- The value of the SOAP actor attribute is a URI. The special URI "<http://www.w3.org/2003/05/soap-envelope/role/next>" indicates that the header element is intended for the very first SOAP application that processes the message.
- Omitting the SOAP actor attribute indicates that the recipient is the ultimate destination of the SOAP message.
- SOAP mustUnderstand global attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process.



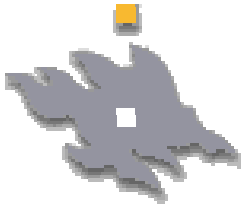
# SOAP Body processing

- A simple mechanism for exchanging mandatory information intended for the ultimate recipient of the message.
- The SOAP Fault element **MUST** appear as a body entry
- Contains :
  - faultcode
  - faultstring
  - faultactor
- Error information related to header entries should be passed as Header entries



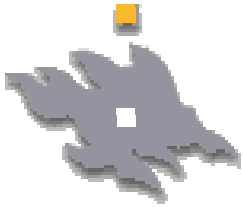
# SOAP RPC

- The URI of the target object
  - A method name
  - An optional method signature
  - The parameters to the method
  - Optional header data
- 
- SOAP relies on the protocol binding to provide a mechanism for carrying the URI. For example, for HTTP the request URI indicates the resource that the invocation is being made against.



# SOAP1.2

- SOAP 1.2 does not permit any element after the body.
- SOAP 1.2 does not allow the `env:encodingStyle` attribute to appear on the SOAP `env:Envelope`
- SOAP 1.2 defines the new `env:NotUnderstood` header element for conveying information on a mandatory header block which could not be processed,
- the `env:mustUnderstand` attribute in header elements takes the (logical) value "true" or "false"
- the `env:mustUnderstand` attribute in header elements takes the (logical) value "true" or "false"
- SOAP 1.2 replaces the attribute `env:actor` with `env:role`
- 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.



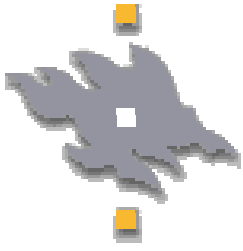
# SOAP1.2

- SOAP 1.2 replaces "client" and "server" fault codes with "Sender" and "Receiver".
- SOAP 1.2 uses the element names `env:Code` and `env:faultReason`, respectively, for what used to be called `faultcode` and `faultstring` in SOAP 1.1. SOAP 1.2 also allows multiple `env:Text` child elements of `env:faultReason` qualified by `xml:lang` to allow multiple language versions of the fault reason.
- SOAP 1.2 provides a hierarchical structure for the mandatory SOAP `env:Code` sub-element in the `env:Fault` element, and introduces two new optional subelements, `env:Node` and `env:Role`.
- In the SOAP 1.2 HTTP binding, the SOAPAction HTTP header defined in SOAP 1.1 has been removed, and a new HTTP status code 427 has been sought from IANA for indicating (at the discretion of the HTTP origin server) that its presence is required by the server application. The contents of the former SOAPAction HTTP header are now expressed as a **value of an (optional) "action" parameter of the "application/soap+xml" media type that is signaled in the HTTP binding.**



# SOAP1.2

- Content-type header should be "application/soap+xml" instead of "text/xml"
- 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.



# SOAP1.2 - Encodings

- An abstract data model based on a directed edge labeled graph has been formulated for SOAP 1.2
- Support of the SOAP 1.2 encodings and SOAP 1.2 RPC conventions are optional.
- `href` attribute in SOAP 1.1 (of type `xs:anyURI`) is called `enc:ref` in SOAP 1.2 and is of type `IDREF`.
- SOAP 1.2 provides several fault sub-codes for indicating encoding errors.



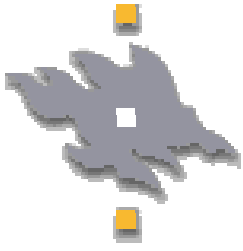
# Changes in SOAP 1.2 from 1.1

- SOAP1.2 is a W3C recommendation
- SOAP 1.2 does not permit any element after the body.
- SOAP 1.2 does not allow the `env:encodingStyle` attribute to appear on the SOAP `env:Envelope`
- SOAP 1.2 defines the new `env:NotUnderstood` header element for conveying information on a mandatory header block which could not be processed,
- The `env:mustUnderstand` attribute in header elements takes the (logical) value "true" or "false"
- SOAP 1.2 replaces the attribute `env:actor` with `env:role`
- SOAP 1.2 defines a new attribute, `env:relay`, for header blocks to indicate if unprocessed header blocks should be forwarded.



## Changes in SOAP 1.2 from 1.1

- SOAP 1.2 provides a hierarchical structure for the mandatory SOAP `env:Code` sub-element in the `env:Fault` element, and introduces two new optional subelements, `env:Node` and `env:Role`.
- In SOAP1.2 the HTTP header "soapAction" does not exist any more, but the contents of the former SOAP Action HTTP header are now expressed as a value of an (optional) "action" parameter of the "application/soap+xml" media type that is signaled in the HTTP binding.



# SOAP1.2 - Message Exchange Patterns

- Must provide a URI to name the MEP
- Describe the life cycle of a message exchange conforming to the pattern
- Describe the relationships of multiple messages exchanged in conformance with a particular pattern (for e.g. responses follow requests and are sent to the originator of the request.)
- Describe the normal and abnormal termination of a message exchange conforming to the pattern



# SOAP1.2 Fault Headers Blocks

- When a SOAP node generates a fault under the below two conditions it should provide the corresponding header blocks along with the general faults in the response :
- VersionMismatch (upgrade) and MustUnderstand (notunderstood).

For example: In the case of the VersionMismatch :

Body should contain the following fault :

```
<env:Fault>
<env:Code><env:Value>env:VersionMismatch</env:Value></env:Code>
<env:Reason>Version Mismatch</env:Reason>
</env:Fault>
```

in addition to the following header entry:

```
<env:Upgrade>
  <env:SupportedEnvelope qname="ns1:Envelope" xmlns:ns1="http://www.w3.org/2003/05/soap-
  envelope"/>
  <env:SupportedEnvelope qname="ns2:Envelope"
  xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope"/>
</env:Upgrade>
```