

# Web Services

Seminar on Semantic Web & Web Services  
- W3C Finland -

6th May 2003

**Suresh Chande,**  
Software Technology Laboratory  
Nokia Research Center,

[suresh.chande@nokia.com](mailto:suresh.chande@nokia.com)

**Acknowledgements:** Markku Laitkorpi / Esa Eerola / Eero Kukko

## Presentation Outline

- Overview
- Core Web Service Technologies
- Web Services for Mobile Domain
- Standardization
- Conclusion

# Overview

- Web Evolution
- Web Services Overview
- Web Services in a Nut shell
- Initial Hype

# Web Evolution

- Web: The **universe of network-accessible information** available through your network enabled devices [W3C].
- Direct human consumable information accessible using User Agents such as browsers (HTML/WML)
  - Static information resources (HTML)
    - Articles and Information resources
    - Corporate\Portal information distribution media
  - Semi-dynamic Information resources
    - CGI, Perl Script, JavaScript, Applets
- Dynamic content for human consumption: Ubiquitous access to Web Content and wide spread utilization of web technologies led to robust tools and technologies to make dynamic content over the web:
  - Perl
  - ASP
  - JSP, Servlets

# Web Evolution

- Success of Web attracted the distributed computing systems, to reap the ubiquitous benefits of the Web and utilize the web for services oriented applications
  - B2B
  - B2C
  - EAI / EDI
- Commercialization of the Web content required distributed systems to be integrated and exposed to the internet, which resulted in technologies such as:
  - J2EE /.NET
  - DCOM
  - CORBA
- Problem with the inter-operability between various deployment platforms lead to seek solutions in a platform neutral manner by using XML as the core technology.
  - Web Services
    - XML, XMLSchema, XML Namespace, SOAP, WSDL

# Web Services Overview - Definition

Web Services are the means of exposing an organizations services to its consumers or partners over the Web utilizing open Web-based standards, which are agnostic to development platform used, vendor tools and the underlying implementation specifics.

*"A Web service is a software system identified by a URI, whose public interfaces and bindings are described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML-based messages conveyed by Internet protocols." [W3C]*

# Web Services Overview - Benefits

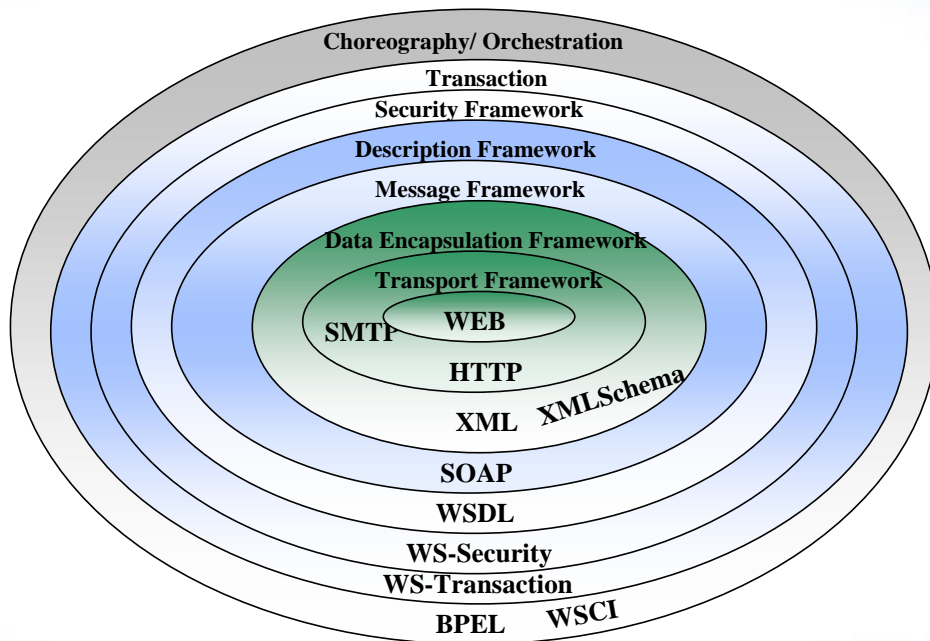
- Allows **loose coupling** between systems willing to communicate and collaborate.
- **Removes platform dependencies** between the communicating systems
- Introduces **rapid application development**
- **Reduces** Integration costs and **speedens** the integration process
- **Low entry barrier**
- Enable **wider utilization** of the services as it does not dictate a platform specific requirements on the consuming clients.
- Continue to **utilize legacy** systems

# Web Services in a Nut Shell

- The next wave of web development - **Programmatic access to the Web Resources.**
- Programmatic access enabled via a set of XML related horizontal Frameworks :
  - Data Encapsulation (XML, XML Schemas).
  - **Message Framework (SOAP).**
  - **Description Framework (WSDL).**
  - **Security Framework (WS-Security).**
  - Discovery Framework (UDDI/ Local WSIL).
  - Transaction (WS-Transaction).
  - Assertions & Authorization (SAML).
  - **Choreography & Workflows (BPEL4WS, WSCI)**
  - Reliability (WS-Reliability)

**SOAP is the Central building block of these set of frameworks !!**

# Web Services - SOAP based Framework



## Web Services - an initial Hype (2001)

- Automatic Business Integration: A Piece of software will automatically discover, access, integrate and invoke new services from unknown companies dynamically without the need for human intervention.
- Easily discover the required services and integrate into existing solutions.
- "Silver bullet", solves all the integration problems that we ever faced in distributed and heterogeneous systems.
- Is ready and available for immediate system development
- Inter-operable and a light weight protocol
- Simple and easy to utilize

- Overview
- Core Web Service Technologies
- Web Services For Mobile Domain
- Standardization
- Conclusion

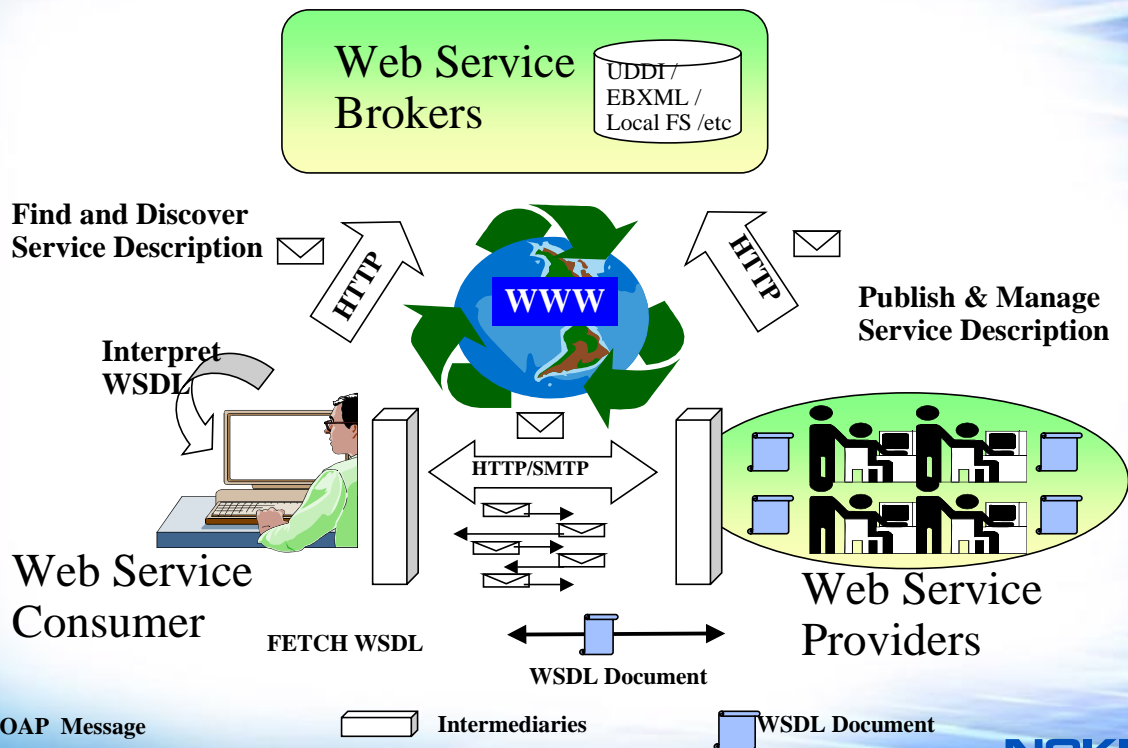
## Core Web Services technologies

- Web Services Architectures
- Core Web Service Protocols
  - SOAP
  - WSDL
- Other Crucial Web Services Technologies
  - Web Services Choreography
  - Web Services Security
  - Web Services & REST

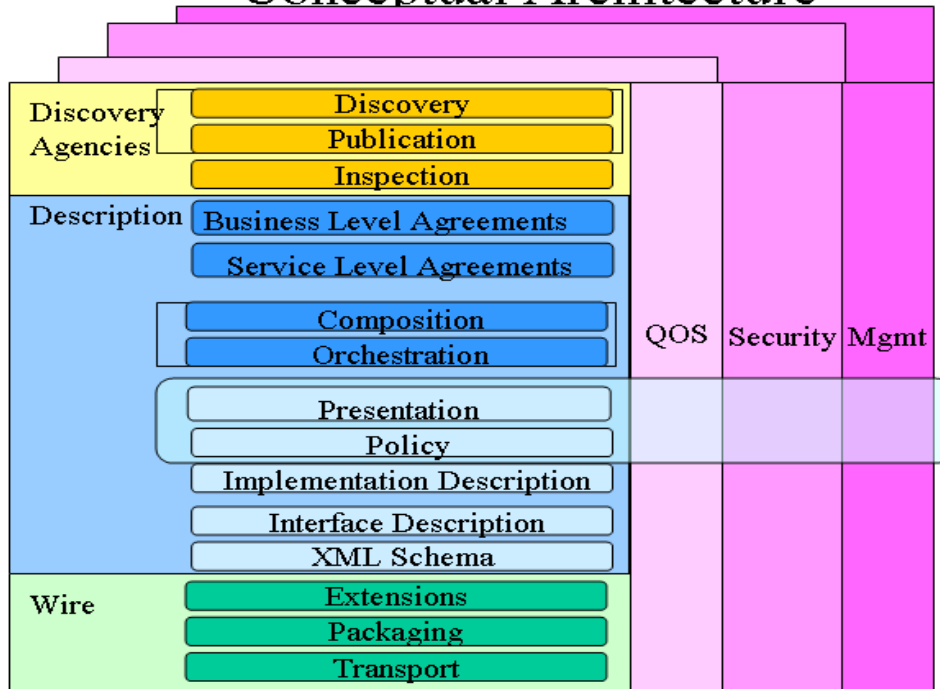
# Web Services Architecture

- The basic Web Service Architecture is defined around three main roles for its participants
  - **Web Services Client (WSC):** The Web Service (WS) consuming client system.
  - **Web Services Providers (WSP):** are participants who desire to publish their service as a Web Service
  - **Web Services Brokers (WSB):** Provide WSC a discovery mechanism to find the required service and at the same time provided WSP means of publish and manage the description about the hosted WS.

# Web Services Architecture



## W3C's Complete Web Services Stack Conceptual Architecture



## Key Web Service technologies

- XML
  - Utilized to encapsulate Data in a platform neutral format
- XML Schema
  - Utilized as XML based data modeling and typing systems
- SOAP
  - An XML Message exchange protocol for application-service communication and exchange of messages
- WSDL
  - A XML based Service description protocol , which defines the service operations one could invoke by exchanging XML based messages (SOAP).

# SOAP

## What is SOAP (formerly known as Simple Object Access Protocol) ?

### Definition :

- Simple 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 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."

### Focus:

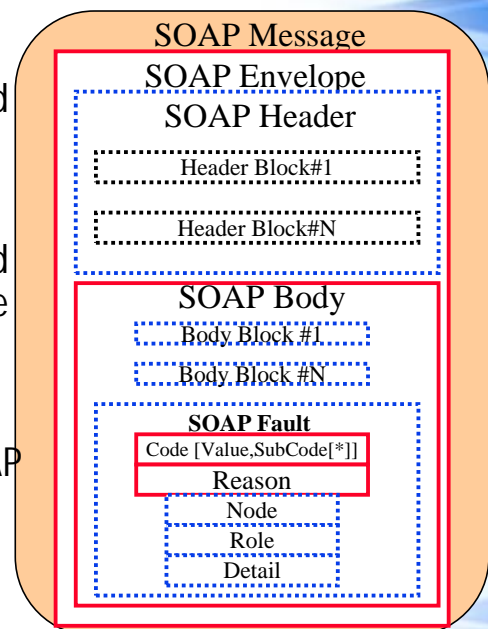
- SOAP looks at the WEB to enable :
  - Invocation of software functionality over the internet
  - **Communication between heterogeneous systems**
  - **Exposure of OO or component based systems**
  - Exchange of inter-application messages
  - EDI / B2B over the internet
- **It is centered around the RPC model**, though it also **supports the document oriented usage of the framework.**

# 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
- **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)

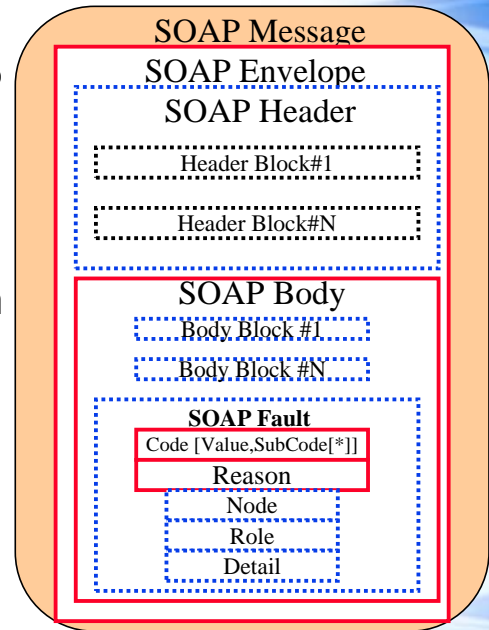
- **Envelope:** This is top level root element of a SOAP Message, which contains the Header and Body element.
- **Header:** A collection of zero or more SOAP header blocks each of which might be targeted at any SOAP receiver within the SOAP message path.
- **Body:** A collection of zero or more *element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

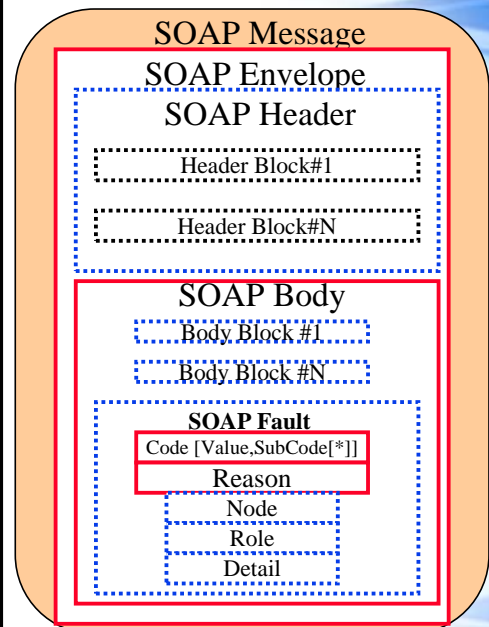
All the above SOAP1.2 Message elements must be defined by the Namespace : <http://www.w3.org/2002/12/soap-envelope>



## Example: SOAP Request Message

```
<?xml version=1.0 encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
<env:Header>
<m:authentication
  xmlns:m="http://www.mobileservices.org/identity"
  env:role="http://www.w3.org/2002/12/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-egg63et9n2f</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

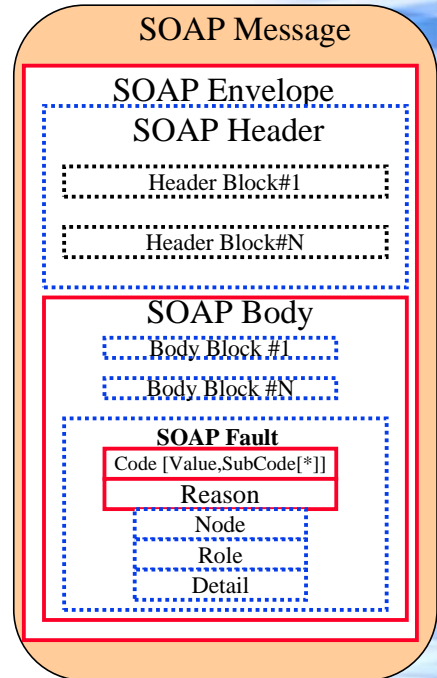


# Example: SOAP Response Message

```

<?xml version=1.0 encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">
<env:Header>
<m:acknowledgement
  xmlns:m="http://www.mobileservices.org/identity"
  env:role="http://www.w3.org/2002/12/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

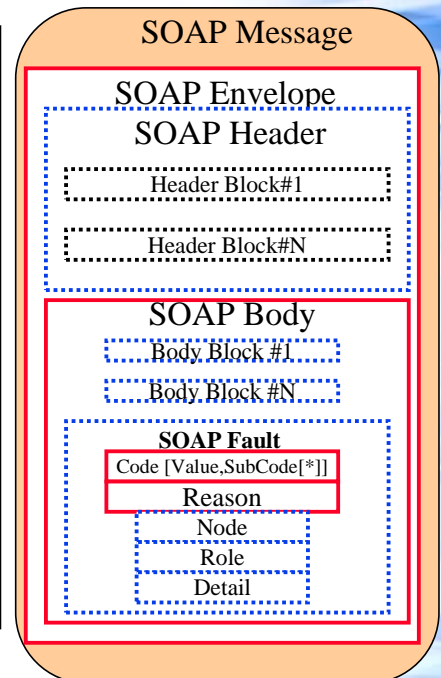


# Example: SOAP Fault Response Message

```

<?xml version=1.0 encoding="UTF-8"?>
<env:Envelope xmlns:env="http://www.w3.org/2002/12/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, without any major efforts from the developer in translating the message to underlying system specifics
- **Document Style:**
  - This enables the SOAP engine to deliver the Message to the corresponding Message Handler. The developer takes the responsibility to handle the messages accordingly within such handlers.

Note: The closer the system is to the Message structure the better the overall performance. If it needs further transformation or translation it will affect the overall system performance.

## 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/2002/12/soap-envelope">
<env:Header>
<m:authentication xmlns:m="http://www.mobileservices.org/identity"
  env:role="http://www.w3.org/2002/12/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

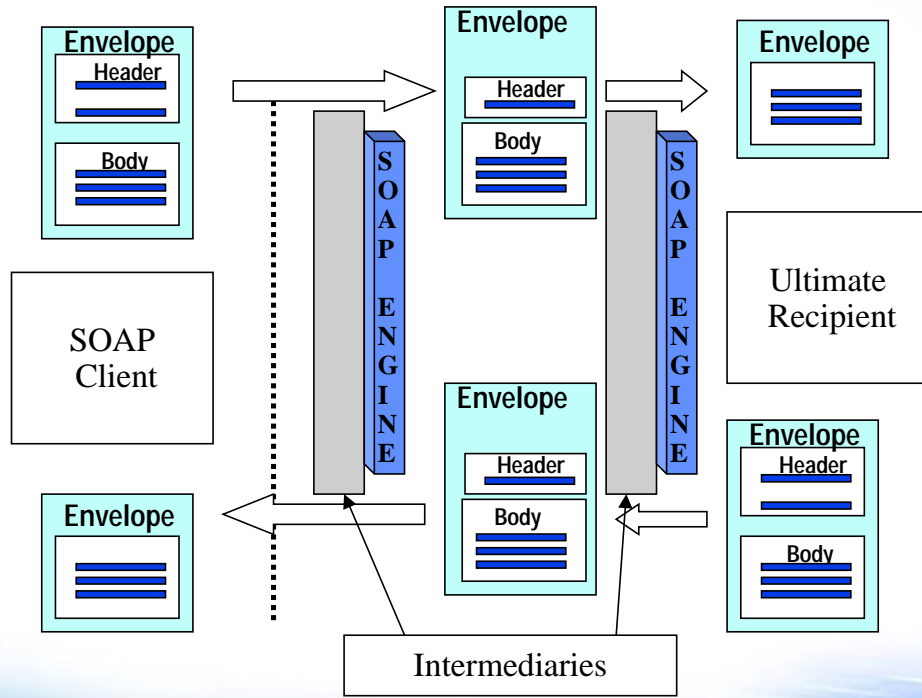
```
<?xml version=1.0>  
<env:Envelope xmlns:env="http://www.w3.org/2002/12/soap-envelope">  
<env:Header>  
<m:authentication xmlns:m="http://www.example.org/identity"  
  env:role="http://www.w3.org/2002/12/soap-envelope/role/next"  
  env:mustUnderstand="true">  
  <m:username>suresh</m:username>  
  <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



# WSDL

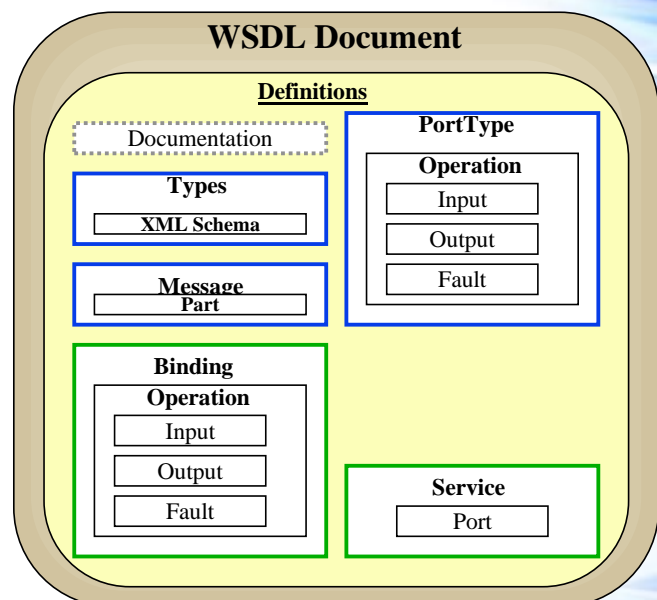
# Web Services Description Language (WSDL)

Originally submitted to W3C on 15 Mar 2001 by the Ariba, Microsoft and IBM and WSDL1.2 is currently a Working draft at W3C

- WSDL provides an XML based grammar to define web services as a collection of the communication end points capable of exchanging messages (procedural or document oriented) which follow a specified message structures in order to satisfy the functioning of a software system over a networked environment.
- The description is modeled into
  - Abstract : containing end points and messages
  - Concrete parts : binds these endpoints and messages to a concrete network deployment and data format bindings
- In principle this split in the service description promotes **multiple deployment models of the baseline service definitions.**

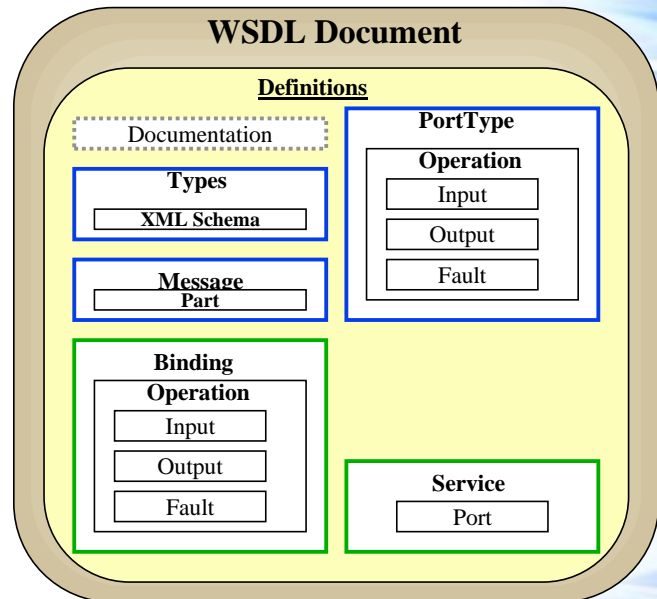
## WSDL1.2 Document Structure

- **Definitions:** WSDL Root element which contains the complete service definition
- **Documentation:** Optional sub element used for documentation which is human readable
- **Types:** Data type definitions (XML Schema data types)
- **Messages:** Abstract definition of the data being transmitted
- **PortType:** Set of Abstract Operations,
  - Operation : each operation refers to Input, Output and Faults
    - Input
    - Output
    - Fault



# WSDL1.2 Document Structure

- **Binding:** Specifies concrete protocol and data format specifications for the operations and messages defined by a particular portType.
  - Operation
    - Input
    - Output
    - Fault
- **Service:** Utilized to aggregate a set of related ports
  - **Port:** Specifies an address for a binding



## WSDL Bindings

WSDL Provides 3 different bindings currently, namely:

- **SOAP**
  - An indication that a binding is based on SOAP 1.2 protocol.
  - Address specification for SOAP endpoints.
  - A SOAP Action URI basically ending up as an HTTP header for the HTTP binding of SOAP.
  - Header, Body and Fault definitions
- **HTTP 1.1 GET and POST Bindings**
  - Non -SOAP based binding
  - Specify the base URI as a port (<http://www.mobileservices.org>) and the specific functionality to be invoked executed in the operations (`/services/delivery/deliver`)
- **MIME Binding**
  - Allows the specification of multipart content being either part of input/output/fault.
  - Relies on either SOAP or HTTP binding as defined above.

## Example: WSDL with SOAP Binding - Delivery WSI

```
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns:http="http://www.w3.org/2003/01/wsd/soap12"
  xmlns:soap="http://www.w3.org/2003/01/wsd/soap12"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:m="http://www.mobileservices.org"
  xmlns:d="http://www.mobileservices.org/delivery"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:mime="http://www.w3.org/2003/01/wsd/mime/"
  targetNamespace="http://www.mobileservices.org/delivery"
  xmlns="http://www.w3.org/2003/03/wsd/">
  <documentation> ... </documentation>
  <types> ... </types>
  <message> ... </message> <message> ... </message> ... N
  <portType> ... </portType> <portType> ... </portType> ... N
  <binding> ... </binding> <binding> ... </binding> <binding> ... </binding> ... N
  <service>
    <port> ... </port> <port> ... </port> ...N
  </service>
</definitions>
```

## Example: WSDL with SOAP Binding - Delivery Web Services Interface [Contd..]

```
<types>
  <xs:element name="sendMessage">
    <xs:complexType>
      <xs:annotation><xs:documentation> ... </xs:documentation></xs:annotation>
      <xs:sequence> <xs:element name="message" type="d:messageType"/> </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="messageType">
    <xs:complexType>
      <xs:annotation><xs:documentation> ... </xs:documentation></xs:annotation>
      <xs:sequence>
        <xs:element name="ID" type="m:UUidType"/>
        <xs:element name="Recipients" type="d:RecipientListType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="RecieipientListType">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MSISDN" type="m:msidType"minOccurs="0" maxOccurs="10"/>
        <xs:element name="email" type="xs:string" minOccurs="0" maxOccurs="10"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  .....
</types>
```

## Example: WSDL with SOAP Binding - Delivery Web Services Interface [Contd..]

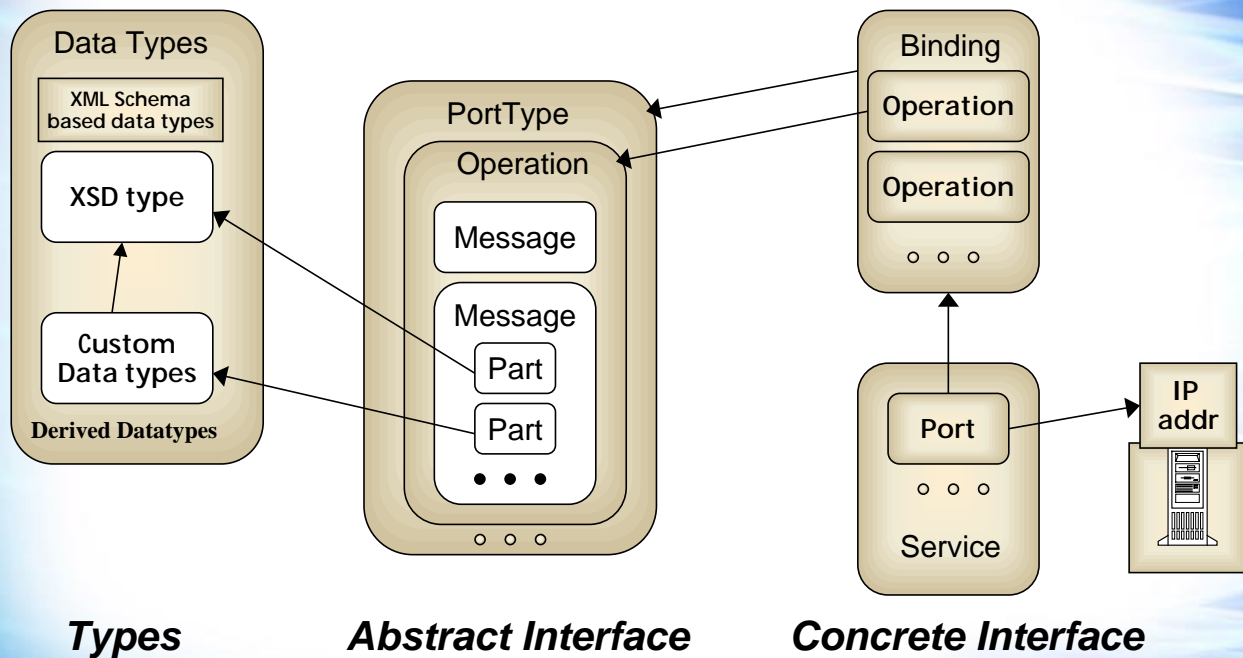
```
<message name="sendMessageIn">
  <part name="parameters" element="d:sendMessage"/>
</message>
<message name="messageDeliveredResponseOut">
  <part name="parameters" element="d:messageDeliveredResponse"/>
</message>
<message name="messageDeliveryFailureOut">
  <part name="parameters" element="d:messageDeliveryFailure"/>
</message>

<portType name="DeliveryService">
  <operation name="sendMessage">
    <documentation>....</documentation>
    <input message="d:sendMessageIn"/>
    <output message="d:messageDeliveredResponseOut"/>
    <fault name="deliveryFailed" message="d:messageDeliveredResponseOut"/>
  </operation>
</ portType >
```

## Example: WSDL with SOAP Binding - Delivery Web Services Interface [Contd..]

```
<binding name="DeliveryServiceSoapBinding" type="d:DeliveryServicePort">
  <soap:binding transport="http://www.w3.org/2003/01/wsd/soap" style="document" />
  <operation name="sendMessageRequest">
    <soap:operation soapAction="http://www.mobileservices.org/services/delivery"/>
    <input>
      <soap:body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </input>
    <output>
      <soap:body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </output>
    <fault>
      <soap:body use="literal" encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
    </fault>
  </operation >
</binding>
<service name="DeliveryService">
  <port name="DeliveryServiceSoap" binding="d:DeliveryServiceSoapBinding">
    <soap:address location="http://www.mobileservices.org/services/delivery"/>
  </port>
</service>
</definitions>
```

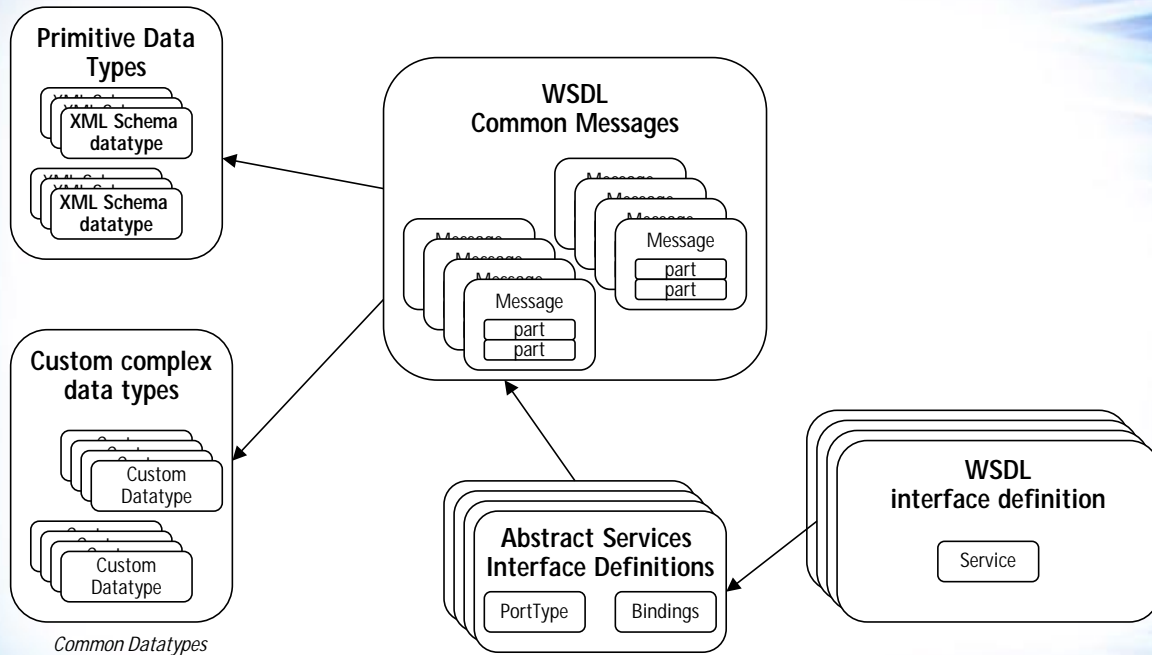
# WSDL Modular Extensions



## WSDL definition Reusability

- WSDL through its "import" element construct enables reusability
  - Helps writing clean and manageable service descriptions in a modular fashion separating the definitions according to their level of abstraction.
  - It increases the ability to reuse definitions from a common set of service definitions.
  - This is particularly beneficial when considering a family of WSI specification which are related to a particular industry domain.
    - Enables definition of reusable Web Services Core libraries which address issues such as: Security, Reliability, Authorization & Authentication, many more

## WSDL Reusability - Enables definition of core WS Libraries



## Other Crucial Web Services Standards

- WS-Choreography / Orchestration [W3C <-> OASIS TC]
- WS-Security [OASIS TC] / WS-Reliability [OASIS TC]
- Web Services and REST

# Web Services Choreography / Orchestration

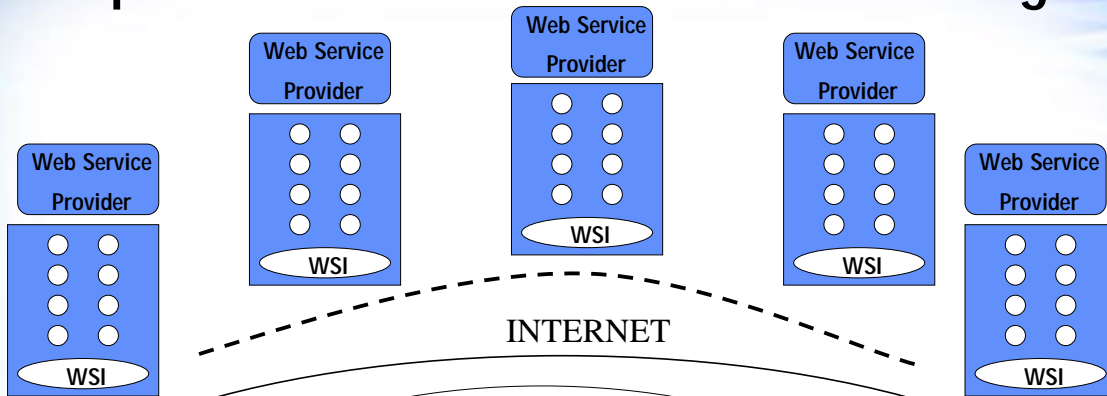
A Web Services technology which enables the definition of a composite Web Service (or referred to as Business Process definition), which can utilize existing static WS specifications (WSDL) and allow the consumption of them in a (semi-)dynamic fashion. This is done so by providing additional syntactic and semantic definitions to specify the consumption patterns.

- The specification will allow choreograph the pattern of consumption of a set of WSI's, by providing a mechanism to specify the following features:
  - Order of execution of individual WSI's,
  - Exception Handling,
  - Correlation between the messages exchanged
  - Map between the messages and service interfaces
  - Invoke WSI's on behalf of the client of this choreography
  - Handle and manage different roles between the WSIs
  - Maintain a state and session during the execution of the Choreography

# Web Services Choreography / Orchestration

- Two different activities in this direction :
  - Web Services Choreography [W3C->WSC1+others]
    - Working group was created January 2003 and work has started here.
  - Web Services Orchestration [OASIS->WSBPEL TC / BPEL4WS] :
    - Formed on 29th April, first meeting on 16th May 2003

# Gaps in the Core Web Service technologies



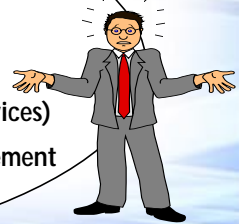
- Order of
- Mes
- In
- Au
- Status

Not all issues are required to be addressed for the non business critical services, Some are addressed by :

- Developer Guides
- Message Level Conventions
- Some are missing
- Some not required

## Usage Model ?

(Composite Services)  
 Context management



# Benefits Choreography adds to Web Services

- Enables the exposure of a **process oriented model** to Web Service Interfaces
- Provide a **simplified interface** to a set of interfaces, Facilitate simplified packaging of a set of interfaces.
- WS with choreography defined can be **better managed** when compared to current WSIs, specially when there exists inter-dependent set of WSIs (Payment, Profile, Delivery & Presence).
- Provides a clear **usage model** for a set of WSIs.
- Allows creation of **composite services**
- Enables specify **design patterns** depicting the best practices of using a set of WSI's
- Allows Provisioning of an **execution environment** which can guarantee execution of the complex/composite service invocation with compensations etc(for e.g. Delivery & Payment)

## Benefits Choreography adds to Web Services [Contd..]

- Provides a good environment for Long running middleware infrastructure. Where there can be pre-defined set of operations that need to be performed over a period of time (even Scheduled). This will **simplify the client interactions** with the WSIs, but in turn needs to interact with the process to manage and monitor the status of such execution.
- The most important factor is also that it helps define a **coarse grained interfaces to the existing granular interfaces (RPC / Document style)**, reducing the interactions between the WSC and Choreography provider.
- Helps **monitor that status** of the request and even **manage the lifecycle** of the process

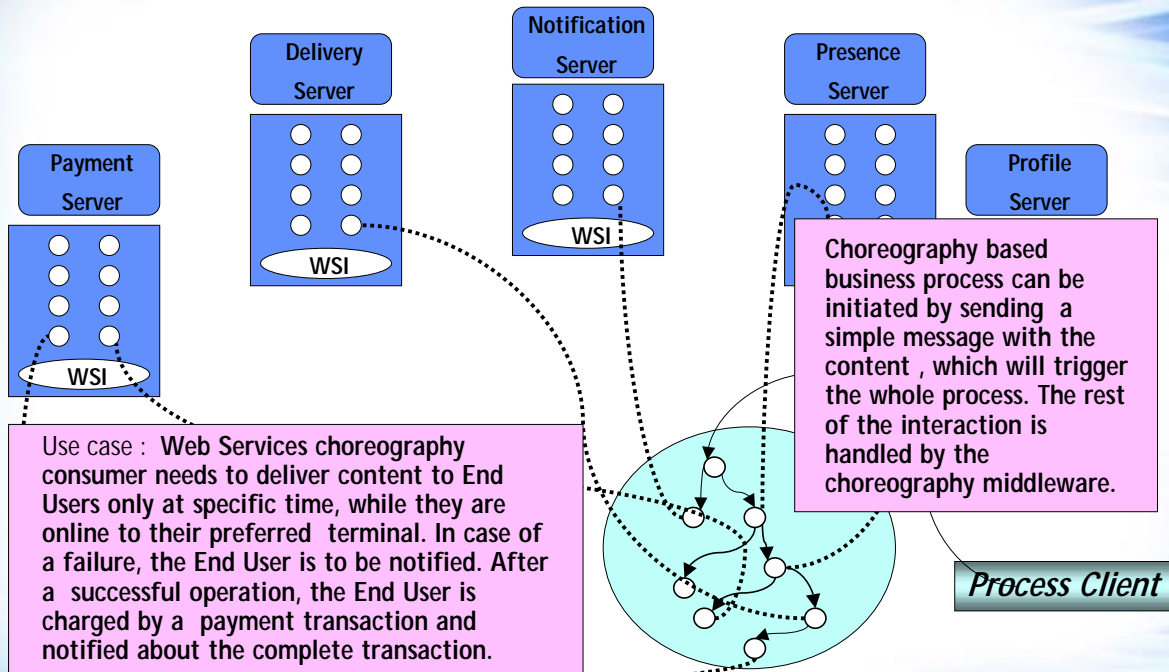
## When not to use WS Choreographies ?

- The WSPs provide a **standalone sets of interfaces** which do not need any correlation between each other
- Each of the Web Service Provider (WSP) handles all the **conversational aspects** in its own internal model
- Each WSP **hides the complexity** of the composite WSI utilization
- The **WSC explicitly handles the complexity of exceptions** and correspondingly manages the compensation handling

### However:

For a set of industry standardized WSIs, WS Choreography might serve inter-operability purposes across multi-vendor implementations of these WSIs.

# An Example: Mobile Web Services Choreography



## Choreography \ Orchestration - Key features

- **Process** : A construct is available to represent a Business process (a complete choreography)
- **Atomic/Composite Activities** : These are individual steps involved in the execution of a choreographed business process
- **Correlation** : Allows correlation between the messages exchanged within the execution of a business process

### Transitions

- **Loop**: Allows iterative execution of each of the steps involved within a process
- **Parallel** :Allows parallel execution of process or sub process within a process
- **Conditional** : Allows a conditional execution of a set of steps involved within a process
- **Sequential** : Allows sequential execution of each of the steps involved within a process.

## Choreography \ Orchestration - Key features

- **Property:** This functionality allows alter and refer the values of the property in context of the execution of the Business process
- **Transactions :** Provides the transactional support within a Process
- **Exception Handling :** Handling of error conditions during the execution of the Business Process Steps
- **Compensation Behavior:** Enables compensate the affects of transaction once it has as successfully executed.
- **Fault Triggers :** Allows to raise a fault condition during the execution of the Business Process
- **Multiple Service Interactions :** Enables the definition of the interconnection between the different Services (operations within as well )
- **Service Lookup :** Facilitates dynamic lookup to identify the next activity within the execution of the Business Process

## Web Services Security

# Basic WS security requirements

- Authentication: **who is requesting** the service (client authentication), who is **providing** the service (server authentication)
- Authorization/Access control: who is **allowed to do what**
- Confidentiality: protect information **against eaves-dropping**
- Integrity: protect information **against modification and replay**
- Audit trail and non-repudiation: prevent **denial of legally binding transactions**
- Key management: distribute, locate and validate public keys (certificates)
- Availability: protect service **against denial-of-service**
- Correctness: proper operation against malfunction and attack
  - NOTE: this is usually an implementation level issue!

# Three generations of Web Services security

(0. None)

- Transport level security:  
SSL/TLS with HTTP authentication
- Message-based security:  
WITHOUT a reusable security context
- Message-based security:  
WITH a reusable security context

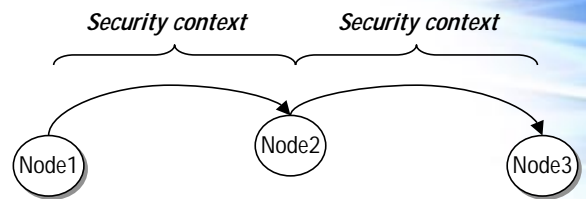


*current technology  
sits about here*

# Transport security vs. message security

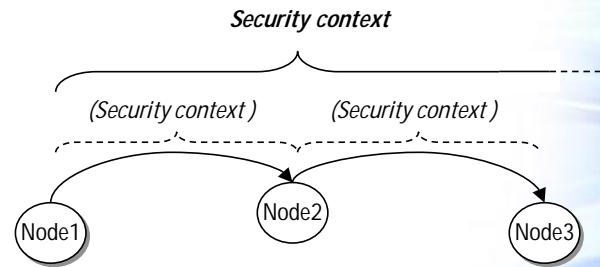
- **Transport security**

- Between two communication endpoints
- Point-to-point security contexts
- Secured path ends when the message exits the transport channel
- Intermediate nodes can read the whole message
- Intermediate nodes must re-establish the secure communication when forwarding the message



- **Message security**

- Message-level security structures decoupled from the comms infra
- End-to-end security contexts
- Secured path continues even when the message exits the transport channel
- Intermediate nodes can read only those parts of the message that are targeted to them
- Secure messaging is guaranteed even if intermediate nodes do not participate in security mechanism
- Possibility of multiple security context in one message



## WS-Security

- IBM, Microsoft & Verisign specification 05 April 2002
- Submitted to OASIS July 2002
- A framework specification
- Defines a standard set of SOAP extensions as message headers that can be used to implement Web Services integrity and confidentiality
  - How to use **XML Signatures** in SOAP messages (integrity)
  - How to use **XML Encryption** in SOAP messages (confidentiality)
  - How to attach or reference **security tokens** in SOAP messages
  - How to **carry security context** for potentially multiple actors
  - How to **associate signatures with security tokens**
  - **Encoding** format for the following **binary security tokens**:
    - X.509 certificates
    - Kerberos tickets

# Web services and REST

## Web Services and REST (web) Principles

- **RE**presentational State Transfer(REST) is a term coined by Roy Thomas Fielding in his PhD Thesis [<http://www.ebuilt.com/fielding/pubs/dissertation/top.htm>] , REST is an **architectural style and not a standard**
- Web consists of addressable resources, which is an item of interest. When a user utilizing an applications selects a specific address(URL) a specific **representation** of that resource is returned over the Web. This representation places the client application into a specific **state**. On accessing another URL gets another representation to client application and in turn **transferring** the state from the current to the new state.
- "Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use."(Roy T Fielding)

# REST principles

- **Stateless Conversation:** All the requests originating from a client and server should contain all the required context for the communication and can not take advantage of any stored data.
- **Cacheable Representations:** Resource representations should be able to be flagged as cacheable/non-cacheable in order to improve the network efficiency.
- **Resource Addressability:** The resources are addressable using URL's and are in turn interconnected via URLs
- **Intermediaries:** There could be several intermediaries introduced between the client and the ultimate resource, such as proxies, gateways, for caching, security, performance, etc
- **Resource Discovery:** Standardized manner of discovering using distributed naming authorities(DNS) and referring to resources (URI)

# REST Guidelines

- View any web based system as a set of **resources addressable as URLs**
- Use **URIs to refer to specific resources** (refer to resources as **nouns and not as verbs**)
  - Noun example: [http://www.mobile.services/ringtone/NokiaTunev1\\_0](http://www.mobile.services/ringtone/NokiaTunev1_0)
  - Verb Example: [http://www.mobile.services/getRingTone?tune=NokiaTunev1\\_0](http://www.mobile.services/getRingTone?tune=NokiaTunev1_0) X
- **Prefer an URI which are logical instead of physical**
  - Logical: [http://www.mobile.services/ringtone/NokiaTunev1\\_0](http://www.mobile.services/ringtone/NokiaTunev1_0)
  - Physical: [http://www.mobile.services/ringtone/NokiaTunev1\\_0.html](http://www.mobile.services/ringtone/NokiaTunev1_0.html)
- **Minimize the use of Query Strings**
- Utilize the appropriate **generic operations** to perform the resource manipulation
  - GET for simple read operations
  - POST for modifying the state of the resource

# REST Guidelines

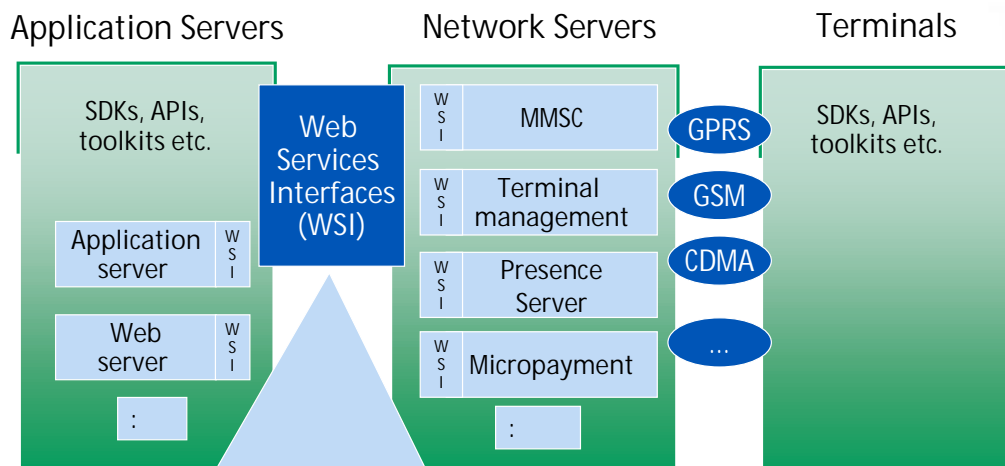
- **Represent resources** using **schemas** with inbuilt capability for extension included.
- Servers and Intermediaries should be **stateless**
- Clients need to maintain the state and state transitions

- Overview
- Core Web Service Technologies
- Web Services For Mobile Domain
- Standardization
- Conclusion

# Rational behind Web Services for Mobile Domain

- The Role of Mobile content is ever growing
- High costs and low development cycle in deploying content Mobile, due to :
  - Fragmentation of frameworks and interfaces ( billing etc)
  - Different technologies in web and mobile domains
  - High integration costs
- 3rd party content and services strongly contributes to operator revenues and complements operators own services
- Ability to charge 3rd party service and content provider for unique mobile assets such as MMS, micro payment, delivery of Java etc
- Faster integration work and lower costs
- Easier access to more users / new revenues via operator partnerships

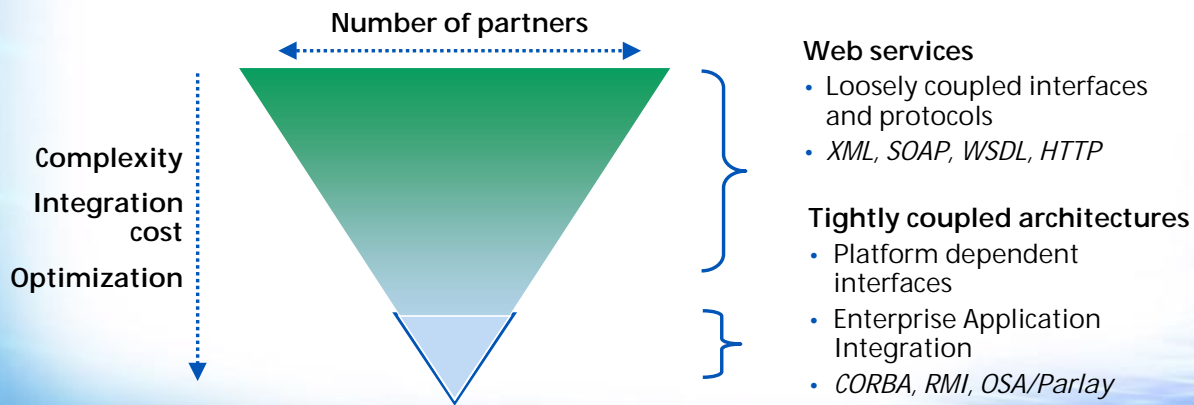
## Web services in mobile industry: Opportunity to lower integration costs between operators and content partners



The key opportunity of Web Services in the mobile context lies in open standard server-to-server interfaces enabling more efficient service and content deployment

# Web services are most useful for fast, simple and flexible low cost integration

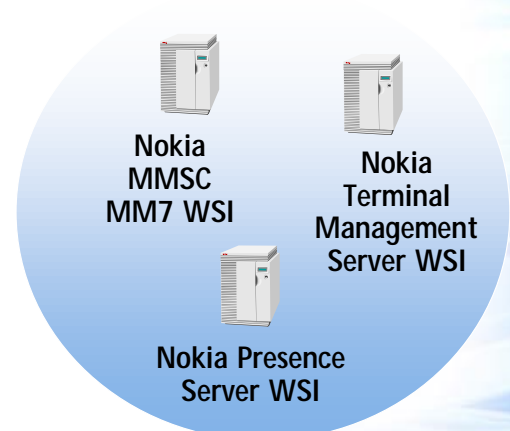
- **Web services:** Optimized for mass partnering, simple low cost integration of different systems / can be build on top of existing systems
- **Tightly coupled architectures:** Optimized for internal operations or limited number of close high volume partners



# Nokia is committed to making Web services happen in the mobile industry

- Nokia's focus now is to demonstrate the viability of the approach and gather learning(s) for Web services standardization in OMA
- Therefore Nokia has now announced the 1st Web services interfaces (WSI) in it's server product portfolio
- Once OMA Web services standards are ready Nokia is planning to implement these throughout the server portfolio

## Nokia's 1st Web services Interface (WSI) Announcements



- Overview
- Core Web Service Technologies
- Web Services For Mobile Domain
- **Standardization**
- Conclusion

## Web Service Standardization Efforts

- **W3C**
  - Core Web Services Technologies
- **OASIS**
  - Web Services Security, Reliable Messaging, Business Process Execution Language,
- **OMA**
  - Open and Inter-operable Mobile Web Services
- **Liberty**
  - Single Signon and Web Services Identify framework
- **WS-I**
  - Web Services Inter-operability

# W3C Web Services Standardization

## Web Services Co-ordination Group

- Coordinate deliverables and dependencies between WG/ Creation and Dissolution of Working Groups/ Manage crises/ Gather and forward requests for additional requirements to the appropriate WG(s). Suggest/propose items relating to architecture to the Web Services Architecture Working Group for its consideration.
- Web Services Architecture Working Group.
- XML Protocol Working Group.
- Web Services Description Working Group.
- Web Services Choreography Working Group.

## W3C

- SOAP1.2 is in candidate recommendation
- WSDL1.2 is working draft currently
- Nokia is active in the development of SOAP 1.2, WSDL 1.2, and Web Services architecture related standards

# OASIS

- OASIS tries to complement the work of standard bodies
- OASIS is participating Web Service standardizations, specifically in the areas of :
  - Web Services Security TC (Nokia Participates in this)
  - Web Services Business Process Execution Language TC
  - Web Services Reliable Messaging TC (Nokia Participates in this)

Others being:

- Web Services Distributed Management TC
- Web Services Interactive Applications TC
- Web Services Remote Portal TC

# Open Mobile Alliance (OMA)

- Drives Open standards for inter-operable Mobile Services
- Mobile web services working group within OMA will create guidelines on how the interfaces should be specified
- The Mobile Web Services group is addressing :
  - A specification Suite that will aid developer to apply Web Services :
    - Web Services Discovery
    - Access &
    - Leverage Service enablers with OMA Framework
- OMA mobile Web Services framework standardisation
  - How to use Web Service technologies and specifications in the mobile domain
  - Use existing specifications and work as much as possible (don't re-invent the wheel)
    - Actual specification done in other forums (e.g. WS-I and W3C)
  - Nokia is an active member in the OMA mobile Web Services framework standardisation

# Liberty Alliance Project

- Specifies an Identity Web Services Framework (Nokia takes an active role here)
- Based on SOAP1.1, WSDL1.1, SAML and WS-Security
- Three distinct Liberty Identity efforts :
  - Federation Framework (ID-FF)
    - Provides core protocols, schemata and profiles. This allows implementers to create standardized, multi-vendor identity federation network.
  - Web Services Framework (ID-WSF)
    - Provides a set of protocols, schemata and profiles to provide a basic framework of identity services, such as: Identity Service discovery and invocation.
  - Service Instance Specification
    - Utilize the ID-FF and ID-WSF to provide network Identity services, such as contacts, presence detection or wallet services that depend on networked identity.

Ref: <http://www.projectliberty.org/specs/draft-lib-arch-idwsf-primer-v1.0-04.pdf>

## WS-I

- Assist in creations and deployment of inter-operable Web Services
- Development of common best practices for Web Services usage in the development, deployment and integration of business applications
- Deliverables for WS-I being :
  - Interoperability Profile
  - Testing tools
  - Sample Applications
- The Basic Profile1.0 consists of SOAP1.1, WSDL1.1, UDDI2.0
- WS-I and OMA do complementary work in mobile web services IOP area
- Nokia is active in the use cases and scenarios work, interoperability testing tool definitions, and sample application work

- Overview
- Core Web Service Technologies
- Web Services For Mobile Domain
- Standardization
- Wrap-up and Conclusion

## Future Web Services Challenges

- Mature Standards and proof of interoperability
- Web Services Reliability
- Web Services Performance
- Web Services Management (Lifecycle management)
- Maintain Interoperability (Surprising !!)
- Open and Royalty Free Standards
- Web Services Choreography and Business Process Execution Language interoperability and collaboration between W3C and OASIS.
- Web Services specifications to extend support for handheld mobile client devices.

# Conclusion

- Web services is nothing new to the Web World but it has taken up different tools to address the previously addressed problems. It has taken up XML as the core tool to ensure inter-application/services communication.
- Web Service technologies will emerge to be more dynamic than it currently exists in the longer term(Semantic Web)
- Web Services will slowly emerge into several industry domains, for instance, we will see the mobile industry embracing this technology.
- The tools are getting mature and one can expect very soon stable and fully standards complaint tool set available from the market.
- Several bodies will together play an important role to ensure Web Services promise for inter-operability, which includes both the standards body and as well as the vendors.
- W3C, OASIS, WS-I, OMA, Liberty, 3GPP, IETF and advocates of the Web have a key role to play here to ensure successful extension of the Web

# Backup Slides

# References

- OASIS Web Services Security TC : [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss)
- Liberty : <http://www.projectliberty.org/>
- WS-I : <http://www.ws-i.org/>
- Open Mobile Alliance : <http://www.openmobilealliance.org>
- OASIS's Technical Committees: <http://www.oasis-open.org/committees/committees.php>
- W3C's Web Services Activities: <http://www.w3.org/2002/ws/>
- Web Services Security : <http://www-106.ibm.com/developerworks/library/ws-secure/>
- Roy T Fielding's PhD Thesis: <http://www.ebuilt.com/fielding/pubs/dissertation/top.htm>
- Roger L. Costello's REST overview: <http://www.xfront.com/REST.ppt>
- REST & SOAP overview, DevelopMentor: <http://www.razorsoft.net/slides/RESTfulSOAP.ppt>
- REST discussion group at Yahoo Groups : <http://groups.yahoo.com/group/rest-discuss/>

## Web Services Overview- Current Deployment

- **Business Integration solutions**
  - Business to Business Integration solutions
  - Enterprise Application Integration
  - Business to Consumers : Yahoo, Google and Amazon in-order to provide basic building block services to business partners.
- **Middleware Solutions:** J2EE, .NET
- **Client Solutions**
  - Web client solutions (Apache Axis, etc).
  - Mobile device clients: Lightweight clients infrastructure for Handheld devices
- **Tools**
  - Integrated Development Environment (IDE's),
  - Automatic code generators
  - WS Interface (WSI) generators from the existing code.

# SOAP Message Framework - Principles

- Modular encapsulation of information such that it is suitable for :
  - Intermediaries
  - Ultimate Recipient
  - Flexibility through extensibility.
- Semi-automatically expose the existing infrastructure as a Web service, through the help of tools. (This is achieved by keeping the framework close to the programmatic model such as RPC)
- Service centric rather than Web Centric:
  - URI is used to locate the service
  - Application functionality is hidden behind the API's, instead of the URI
  - Message's encapsulate the communication intention instead of the underlying protocol (HTTP/SMTP)

# SOAP Message Framework - Principles [Contd.]

- Utilizes Web as transportation
  - SOAP message contains all the application interpretable information
  - Web and protocols below SOAP are meant for transportation purposes.
- Bindings can be altered without major effect on the functionality
  - Addressing is done at the message level and is application controlled.
  - Not tightly integrated to the transport binding protocols
    - + Can support wider transport protocols with minimal efforts (SMTP,HTTP, SIP, FTP,others.)
    - - Could cause inefficient utilization of the transport and the general infrastructure if not applied properly.

## SOAP Message Framework - Principles [Contd.]

- Developer friendly
  - Programmatic representation makes it developer friendly due to similarity to the deployed environment's programming model.
  - RPC style :
    - + Removes the need for additional code to interpret the XML documents
    - + Directly invokes the components or legacy system (this is usually generated automatically).
    - - too granular.
    - - expose the complexity to the client implementation.

## SOAP Message Framework - Disadvantages

- The application level API complexity is exposed to the client system.
- Any header that needs to be interpreted requires parsing of the entire message at each intermediary level
- Performance relies on the number of intermediaries and message complexity.
- Does not utilize the underlying protocols and relevant infrastructure to its fullest extent.

# Java Community Process

- JSR-172: Web Services for Mobile Devices
  - Is based on the JAX-RPC technology , but for the J2ME java platform for mobile devices.
  - Provides tools such as :
    - XML Parsing APIs
    - WSDL to java mapping,
    - API to support generated stubs,
    - Investigation of a suitable and compact encoding mechanism for XML based RPC messages

## Changes in SOAP 1.2 from 1.1

- SOAP1.2 is on a candidate 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/2002/12/soap-envelope"/>
<env:SupportedEnvelope qname="ns2:Envelope" xmlns:ns2="http://schemas.xmlsoap.org/soap/envelope"/>
</env:Upgrade>
```

# Web Services Development Model

## Top Down Approach

### A. WSDL Creation:

- Identify & define the data types to be used (Custom as well as from the XML Schema DT base)
- Define the interfaces to be used
- Define the different operations & their bindings

### B. Utilize tool to create stubs :

- Run through tool to create required automatically generated parts of the code

### C. Implementation:

- Complete the implementation to receive the SOAP requests, process and deliver a Response incase required to do so.

### D. Deploy the WS:

- Utilize the tools to deploy and host the WS
- Publish the WS into a registry like using UDDI, WSIL (optional)

# Web Services Development Model

## Bottom Up Approach (Implementation exists)

### A. Implementation

- If not already existing Develop a system using Java, C++ or any other programming language.

### B. Create WSDL :

- Define the WSDL Manually and then map the SOAP requests to the API calls using the standard APIs or then
- Run through tool to create required automatically generated parts of the WSDL and the bindings (for example Java2WSDL, ApacheAxis).
- Integrate the generated parts

### C. Deploy the WS

- Utilize the tools to deploy and host the WS
- Publish the WS into a registry like using UDDI, WSIL (optional)

## Most important mobile Web services interfaces to standardize from business perspective



- MMS
- Delivery (downloading Java, polyphonic ring tones etc)
- Payment
- Location
- Presence
- Device management
- Authentication

# Good start in Web services standardization for the mobile industry

- Promising 1st steps to standardize how to apply Web services technology in mobile
  - 3GPP in MMS: MM7
  - Mobile Web Services Group in Open Mobile Alliance (OMA) started in August 2002
- Nokia views that the Open Mobile Alliance (OMA) is well positioned to lead Web Services standardization in the Mobile Industry
  - Over 220 member companies representing the whole value chain needed for mobile services
  - Cooperation between OMA and other key forums is crucial (3GPP, W3C, OASIS, Liberty etc)

## XML Signature

- **W3C Recommendation 12 February 2002**
- Detection of message tampering
  - partial or full signing of a message
  - multiple signatures for a single message
  - nonce & recipient can be included for nonrepudiation
- Detection of forgery
  - certificate chain validation
- Support for multiple encryption algorithms
- Minimal canonicalization required
  - make equivalent documents equal at byte level before signature creation and validation
- Verifying a signature
  - binding of document-to-key (signature validation against the doc)
  - binding of key-to-identity (certificate)
  - verifying the current validity of key-to-document binding

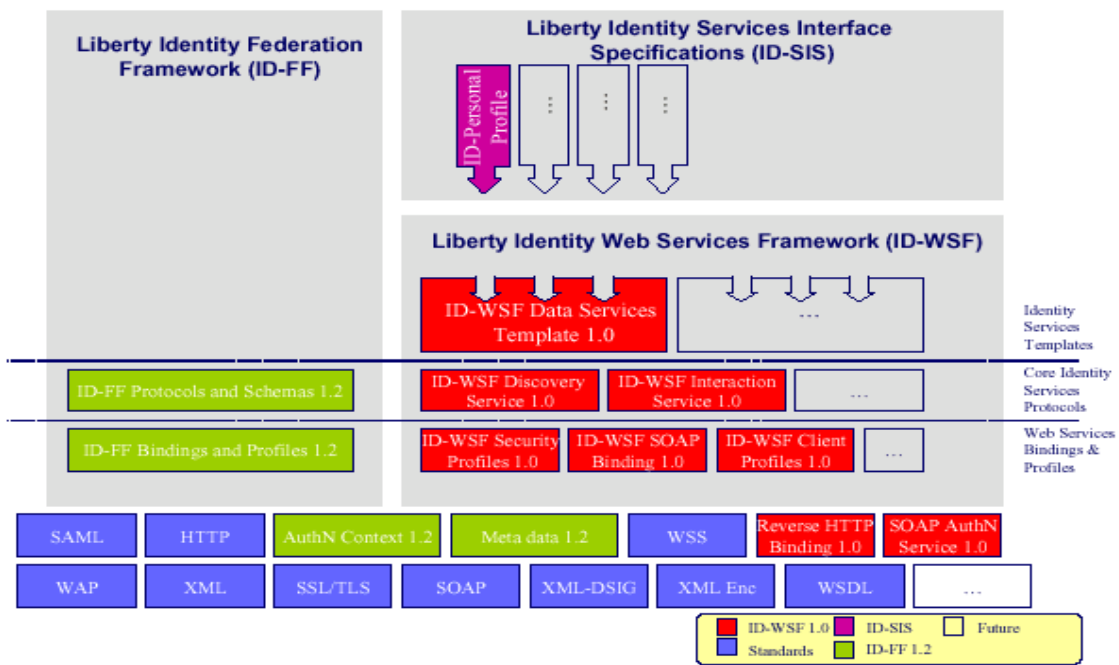
# XML Encryption

- **W3C Candidate Recommendation 02 August 2002**
- Message confidentiality
  - partial or full encryption of a message
- Support for multiple encryption algorithms
- Encryption key can be further encrypted
  - useful when encryption is done by a randomly generated symmetric key that is in turn encrypted by the recipient's public key
- Minimal canonicalization required
  - make equivalent documents equal at byte level before signature creation and validation
- Encrypted data cannot be validated until decrypted
- If data is signed before encryption, signature should be encrypted as well
  - plain text signature exposes some deciphering vulnerabilities

# XML Key Management Specification (XKMS)

- **W3C Working Draft 18 March 2002**
- SOAP based PKI operations against some PKI provider
- Functions
  - Registering (and revoking) public keys with an XKMS-compliant PKI service
    - X-KRSS Key Registration Service Specification
    - Certificate creation and registration
    - Certificate revocation
    - Recovery of public (and optionally private) keys
  - Locating or validating a public key already registered with an XKMS-compliant PKI service
    - X-KISS Key Information Service Specification
    - Location and retrieval of public keys
    - Certificate validation
- Used in conjunction with XML Encryption and XML Digital Signature
  - For example, a signature containing a certificate could be sent to an XKMS service to extract the public key

# The Liberty Alliance Project



**Figure 1: Liberty Modules**

Ref: <http://www.projectliberty.org/specs/draft-lib-arch-idwsf-primer-v1.0-04.pdf>