Middleware and Distributed Systems

Inter-Process Communication With XML

Martin v. Löwis

XML Protocols

- Use XML as presentation layer (similar to XDR or BER)
- Client and server exchange request and response by XML documents
- Typically on-top-of lower layer transport protocol (e.g. session handling)
- Rationales
  - Easy to implement, can rely on existing tools
  - No marshaling for already present XML payload data
- Drawbacks
  - Young technology
  - Lack of language mappings

XML

- Markup language, subset of SGML (like HTML)
- HTML defines presentation, XML describes structured, hierarchical data
- XML schema: Definition of meaning of the tags within a XML document
- Scoping of elements through namespaces
- XML 1.0
  - Second edition of W3C Recommendation
  - Started in 1996, first published in 1998

XML-RPC

- Invented by Dave Winer (www.xml-rpc.com)
- Fixed XML vocabulary for RPC, with fixed set of data types
  - Four-byte integer (<int> / <i4>), <boolean>, <double>, <string>, <base64>
  - <dateTime.iso8601>, e.g. 19980717T14:08:55
  - <struct>: seq of **member** elements, each with **name** and **value** element
  - <array>: single **data** element, with a sequence of **value** elements
    - No need for homogenous data types in an array
- Based on HTTP
- Multiple implementations

XML-RPC Example Request

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>
<methodCall>
  <methodName>examples.getStateName</methodName>
  <params>
    <param><value><i4>41</i4></value></param>
  </params>
</methodCall>
```

XML-RPC Example Response

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
  <params>
    <param> <value><string>South Dakota</string></value>
</param>
  </params>
</methodResponse>
```

XML-RPC Fault Response

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 426
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:02 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?>
<methodResponse>
<fault>
  <value>
    <struct>
       <member><name>faultCode</name>
         <value><int>4</int></value>
      </member>
      <member><name>faultString</name>
        <value><string>Too many parameters.</string></value>
     </member>
   </struct>
  </value>
</fault>
</methodResponse>
```

Web Services

- Gartner Group:
  „Web services are software technologies, making it possible to build bridges between IT systems that otherwise would require extensive development efforts.“

- World Wide Web Consortium (2003):
  „A web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.“

- Oracle:
  “A Web service is a software system identified by a URI, whose public interfaces and bindings are defined and described using XML.”

- Matthew McDonald:
  „Web Services are a modified version of COM with a little difference.”

Architecture

- Service interface for applications in the web
  - „Human-readable CORBA through port 80“
- Roles: service provider, service registry, service consumer
- Operations: publishing, finding, binding, usage of services
- Web services are layered on-top-of existing protocols (HTTP, TCP/IP)
  - Relies on proven web concepts
  - URI: Uniform Resource Identifier (either URL or URN)
  - XML: Extensible Markup Language

Web Service Components

- SOAP: XML-based message format for communication
- WSDL: XML-based description language for web services (endpoints, operations, messages)
- UDDI: Directory for web services

- Roles and activities
  - WSDL document is published to UDDI by the web service provider
  - Service consumer looks for services in UDDI
    - Obtains URI to the WSDL document as result
  - Consumer calls the service through the SOAP protocol

Web Services Definition Language

- Description of Web Services - everything needed for consumption (structure, connectivity, message flow)
- Interface definition: endpoints, operations, messages
- XML vocabulary for describing services

- Definition of *services* as a collection of *ports* (network endpoints)
  - Definition of *messages*: data that are exchanged
  - Definition of *port types*: abstract collections of operations
- Definition of *bindings*: specification of protocol and data format

C	IBM, Microsoft, Ariba
S	1.1: W3C Recommendation
I	All

WSDL Example

```
<?xml version="1.0"?>
<definitions name="StockQuote“

   targetNamespace="http://example.com/stockquote.wsdl"
        xmlns:tns="http://example.com/stockquote.wsdl"
        xmlns:xsd1="http://example.com/stockquote.xsd"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
<types>…</types>
<message>…</message>
<portType>…</portType>
<binding>…</binding>
<service>…</service>
</definitions>
```

Definition of data types used by the Web Service

Messages used by the Web Service

Endpoints which offer the Web Service

Communication protocols used by the Web Service

Operations performed by the Web Service

WSDL Types

- Based on W3C XML Schema definitions
- Declared types are referenced in the message specification part

```
<types>
   <xs:schema targetNamespace="http://example.com/stockquote.xsd"
              xmlns:xs="http://www.w3.org/2000/10/XMLSchema">
      <xs:element name="TradePriceRequest">
         <xs:complexType name="TickerDesc"><xs:sequence>
            <xs:element name="tickerSymbol" type="string"/>
         </xs:sequence></xs:complexType>
      </xs:element>
      <xs:element name="TradePrice">
         <xs:complexType><xs:sequence>
            <xs:element name="price" type="float"/>
         </xs:sequence></xs:complexType>
      </xs:element>
   </schema>
</types>
```

WSDL Messages

- Message element describes the data elements of an operation
- Each message can consist of one ore more parts
  - Each part specified either as an element or a type
  - Actual (XML) representation depends on the binding
- Parts refer to function call parameters in RPC scenario

```
<message name="GetLastTradePriceInput">
   <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<message name="GetLastTradePriceOutput">
   <part name="body" element="xsd1:TradePrice"/>
</message>
```

WSDL Port Types

- Describes the list of abstract operations for a Web Service
- Each operation has optional input, output, fault messages
- 4 possible transmission primitives / operation types:
  - One-way: Endpoint receives a message
  - Request-response: Endpoint receives a message, and sends a message
  - Solicit-response: Endpoint sends a message, receives a correlated message
  - Notification: The endpoint sends a message

```
<portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
           <input message="tns:GetLastTradePriceInput"/>
           <output message="tns:GetLastTradePriceOutput"/>
        </operation>
</portType>
```

WSDL Bindings

- Message format and protocol details for each port
  - Binding type attribute relates to *portType* definition
- In case of SOAP binding, attributes for SOAP encoding style and transport protocol (defined in WSDL SOAP binding)
  - Mapping of WSDL operations, specification of SOAP encoding for each one

```
<binding name="StockQuoteBinding‟ type="tns:StockQuotePortType">
 <soap:binding style="document"
               transport="http://schemas.xmlsoap.org/soap/http"/>
 <operation name="GetLastTradePrice">
   <soap:operation soapAction="http://example.com/GLTrdPrice"/>
   <input>
      <soap:body use="literal"/>
   </input>
   <output>
      <soap:body use="literal"/>
   </output>
 </operation>
```

WSDL Services

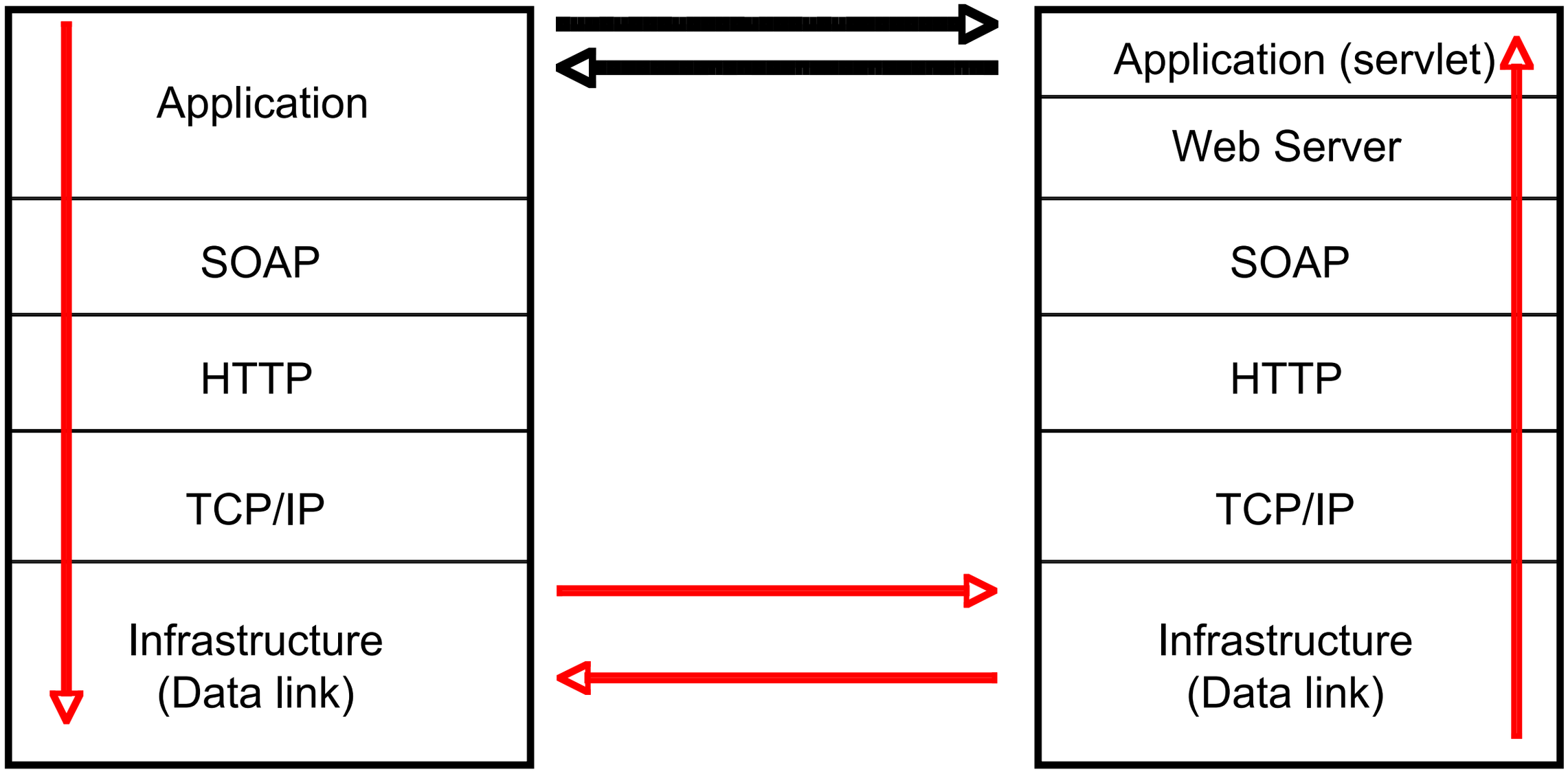
- List of ports
- Defines physical location for a communication end-point
- Name, message format and protocol details for each port
- Again specific elements from WSDL SOAP binding

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

SOAP

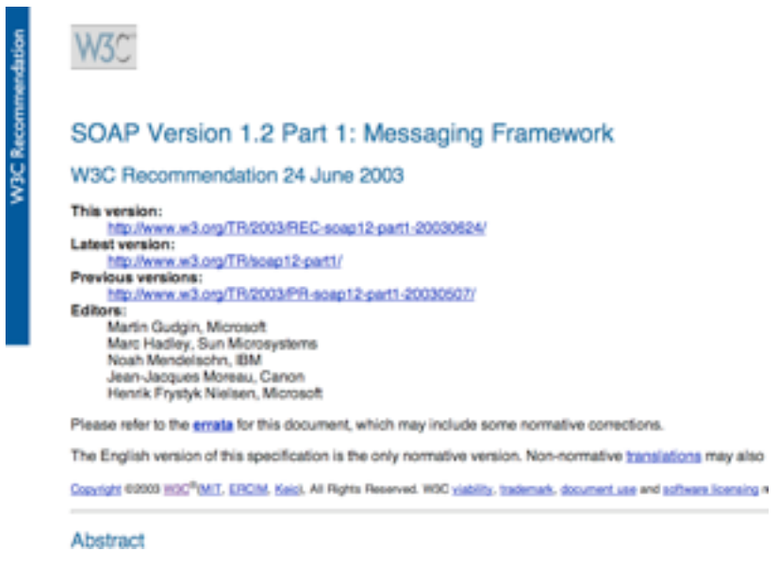
- XML-based messaging protocol, specified by W3C
- “lightweight protocol for exchange of information in a decentralized, distributed environment”
- Current version 1.2 (W3C), version 1.1 widely used
- Processing model (roles, relays)
- Fault information
- Three parts in the SOAP specification:
  - Envelope structure for defining messages
  - Rules for encoding application-specific data types
  - Convention for doing RPC

SOAP Protocol Stack



SOAP Usage Scenarios

- Fire-and-forget to single/multiple receiver (notifications)
- RPC, Request/response asynchronous communication
- (Multiple) Third party intermediary
- Request with acknowledgment
- Request with encrypted payload
- Multiple asynchronous responses
- Caching
- Routing


W3C Recommendation

W3C

SOAP Version 1.2 Part 1: Messaging Framework

W3C Recommendation 24 June 2003

This version:
http://www.w3.org/TR/2003/REC-soap12-part1-20030624/
Latest version:
http://www.w3.org/TR/soap12-part1/
Previous versions:
http://www.w3.org/TR/2003/PR-soap12-part1-20030507/
Editors:
Martin Gudgin, Microsoft
Marc Hadley, Sun Microsystems
Noah Mendelsohn, IBM
Jean-Jacques Moreau, Canon
Henrik Frystyk Nielsen, Microsoft

Please refer to the errata for this document, which may include some normative corrections.

The English version of this specification is the only normative version. Non-normative translations may also

Copyright ©2003 W3C®(MIT, ERCIM, Keio). All Rights Reserved. W3C viability, trademark, document use and software licensing

Abstract

SOAP Non-Goals

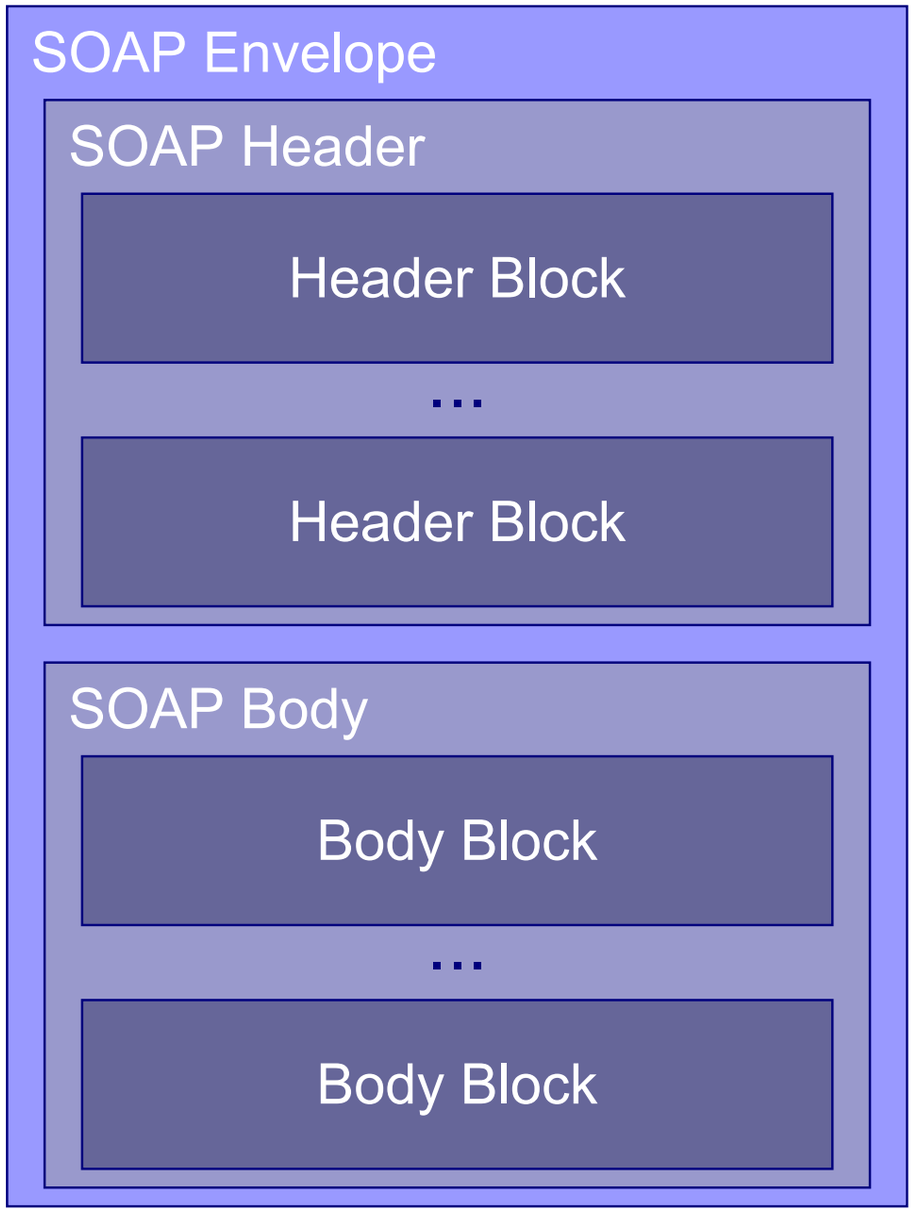
- Distributed Garbage Collection
  - Resource-related WS standards
- Batching of messages
- Objects-by-reference
  - But WS-Addressing introduces EPR’s
- Activation
  - SOAP is a protocol specification

SOAP Companions

- WSDL: Web Services Definition Language
  - Interface definition: endpoints, operations, and messages
  - XML vocabulary for describing services (SOAP, HTTP, MIME)
- UDDI: Universal Description Discovery and Integration
  - Repository service for discovery of services
  - XML schemas for various information models

SOAP Messages

- Emitted by sender, targeted at ultimate receiver, through intermediaries (loosely coupled, point-to-point)
- Header blocks can be specific to above actors
- Might be required to understand header blocks
- Body blocks are always for the ultimate receiver
- Intermediary processes and removes appropriate header blocks, may add new header blocks



SOAP Envelope & 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</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>
 <env:Body> ... </env:Body>
</env:Envelope>
```

SOAP Header Roles

- Predefined roles
  - *next*: Each SOAP intermediary and the ultimate SOAP receiver MUST act in this role
  - *none*: SOAP nodes MUST NOT act in this role
    - never formally processed
    - may carry data that is required for processing of other header blocks
  - *ultimateReceiver*: The ultimate receiver MUST act in this role
- *env:mustUnderstand* is “true” or “false”
- Intermediary may drop, modify, or add headers
  - Headers targeted at intermediary are removed, unless *env:relay* is true (and unless processing of the header reinserts it)

SOAP Body

- Mandatory part of the SOAP message, predefined tag name
  - Payload content must be well-formed XML, without XML prolog
  - Information is interpreted by the application
  - Special encoding of binary data (SOAP with attachments, MTOM)
- Body processing
  - Targeted at ultimate receiver
  - Structure completely application-defined, except for faults
  - Structure of body specified in *env:encodingStyle* attribute
    - No default value

SOAP Body

```
<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>
</env:Body>
```

SOAP Faults

- SOAP fault block (*env:Fault*) as only child element of the SOAP body in the response message
- *env:Code* element with *env:Value* child
  - Possible values are *env:{VersionMismatch, MustUnderstand, DataEncodingUnknown, Sender, Receiver}*, possible additional *env:Subcode* child
- *env:Reason*, with optional *env:Text* children
- Optional *env:Node*, with URI content
- Optional *env:Role*, with URI content
- Optional *env:Detail*, with arbitrary attributes and child elements

RPC with SOAP

- Special use case, since transfered XML document now represents RPC
- SOAP Version 1.2, Part 2: Adjuncts
- Demands transport of request parameters and response values / fault information
- Different XML serialization approaches for graphs of untyped data structures
  - SOAP encoding, based on Section 5 from the SOAP spec
  - Literal encoding, where the structure is defined by XML Schema in the WSDL
- Different message styles
  - RPC style - body must conform to structure that indicates RPC data
    - Mapping of incoming data to RPC paradigm by SOAP stack
  - Document style - opaque payload, mapping to RPC by application

SOAP Encoding Example

```
<xsd:complexType name="Point">
  <xsd:sequence>
    <xsd:element name="x" type="xsd:int" />
    <xsd:element name="y" type="xsd:int" />
  </xsd:sequence>
</xsd:complexType>
...
<wsdl:message name="DistInput">
 <wsdl:part name="p1" type="tns:Point" />
 <wsdl:part name="p2" type="tns:Point" />
</wsdl:message>
<wsdl:message name="DistOutput">
 <wsdl:part name="result" type="xsd:float" />
</wsdl:message>
<wsdl:portType name="Geometry">
 <wsdl:operation name="Dist">
  <wsdl:input message="tns:DistInput" />
  <wsdl:output message="tns:DistOutput" />
 </wsdl:operation>
</wsdl:portType>
...
```

SOAP Encoding: Resulting Example Request

```
...
    <ns:Dist xmlns:ns="...">
      <p1>
        <x>10</x>
        <y>20</y>
      </p1>
      <p2>
        <x>100</x>
        <y>200</y>
      </p2>
    </ns:Dist>
...
```

```
...
    <ns:Dist xmlns:ns="...">
      <p1 href="#id1" />
      <p2 href="#id1" />
    </ns:Dist>
    <ns:Point id="id1"
     xmlns:ns="...">
      <x>10</x>
      <y>20</y>
    </ns:Point>
...
```

```
Point one = new Point(10, 20);
Point two = new Point(100, 200);
float f = proxy.Distance(one, two);
```

```
Point one = new Point(10, 20);
float f = proxy.Distance(one, one);
```

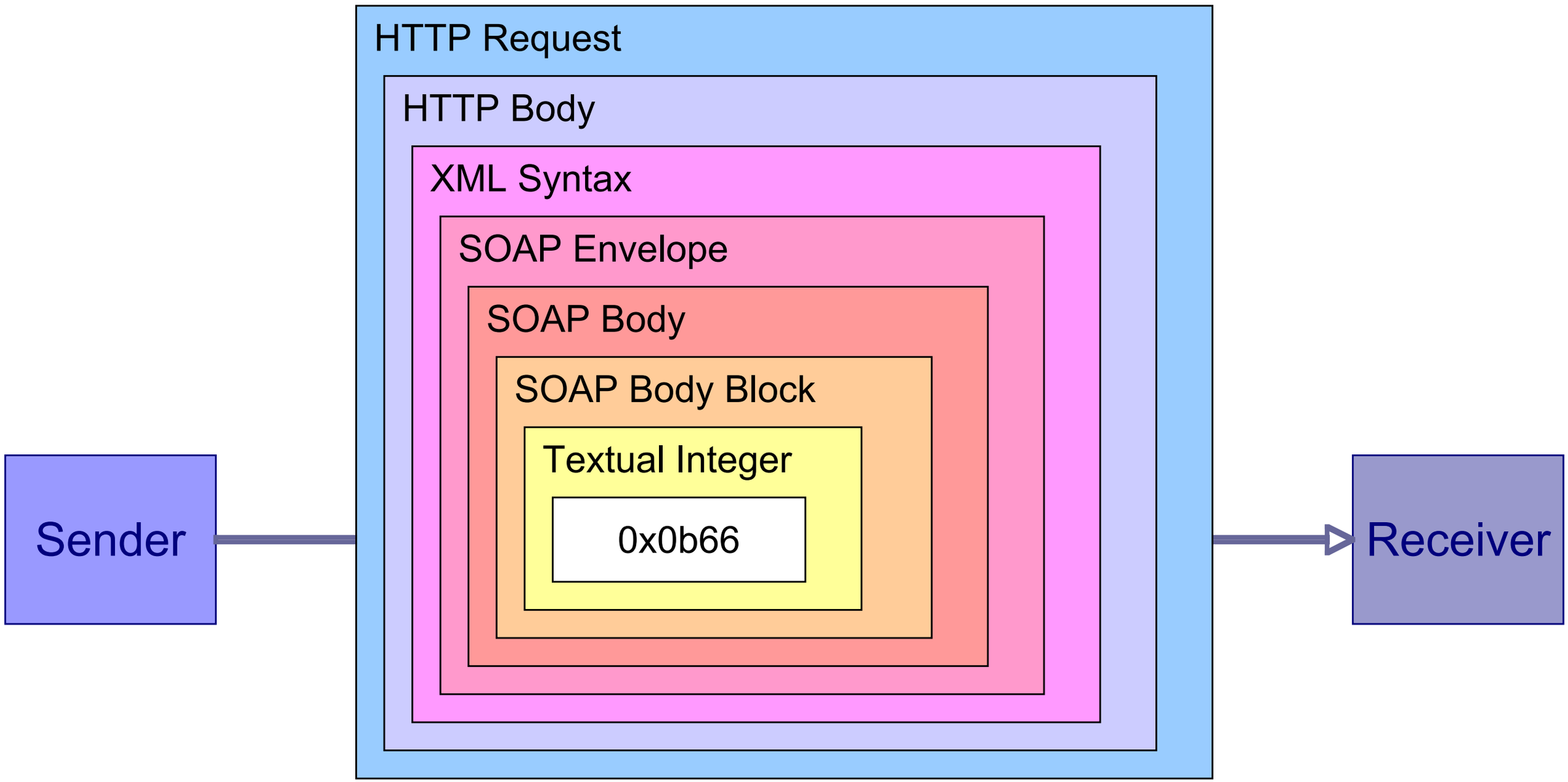
Document Style SOAP

- Document/literal is based on XML schema definition
  - More powerful description, also for complex parameter types
- Demands usage of XML parsing technology in applications
- RPC encodings lacks mapping from XML schema type to object graph (e.g. xsd:sequence -> graph)
- RPC- vs. Document-style was the primary source of WS implementation interoperability problems in the past
  - RPC/encoded (JAX-RPC) vs. Document/literal (.NET)
  - Document/wrapped as special literal style, to express the operation name
- WSDL defines encoding style for communication with the specified Web Service

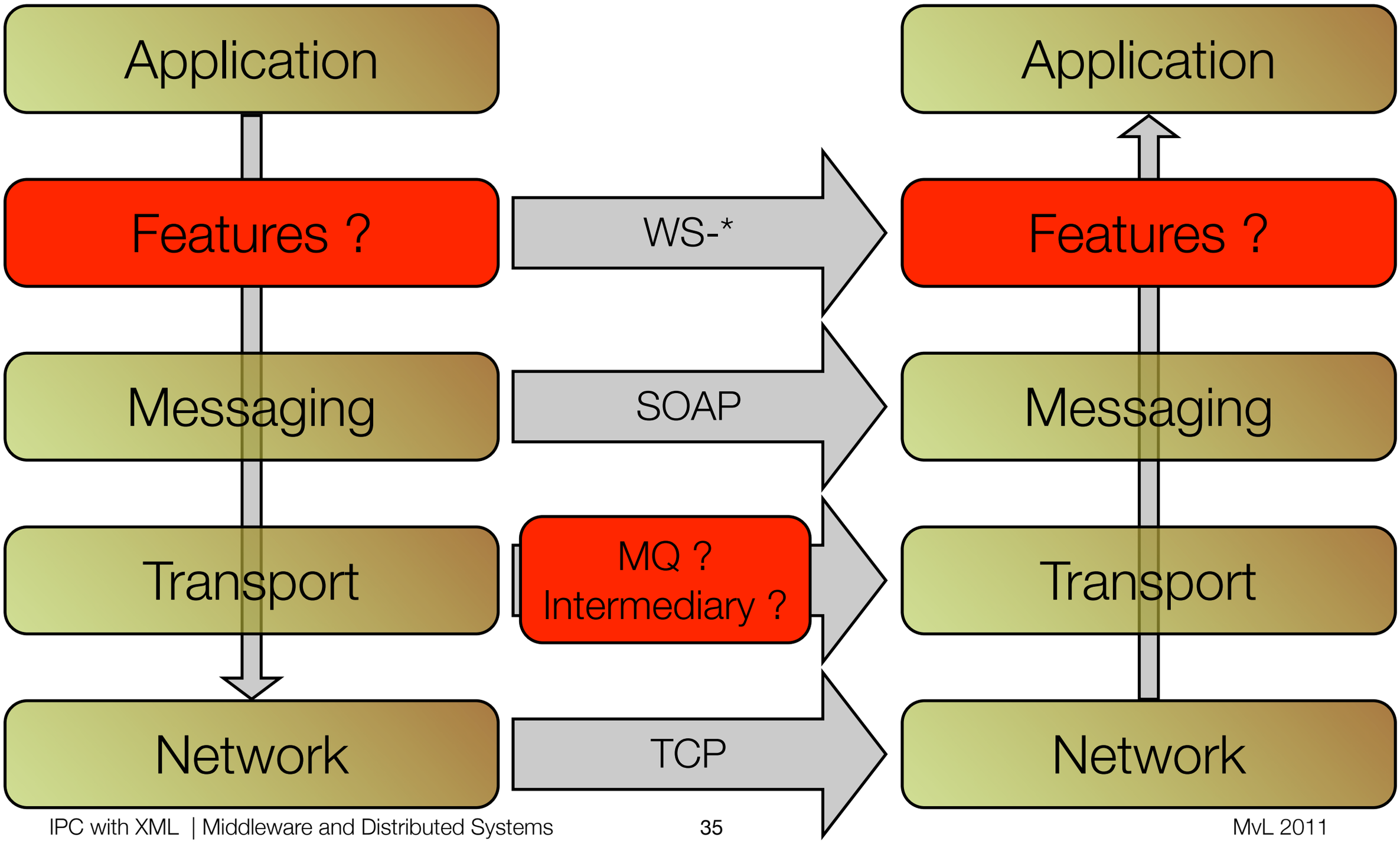
Encodings Today

- RPC/literal allows formulation of data structures with Schema
  - Still demands ‚RPC‘ - style organization of body
  - Schema does not tell the message body infoset, no validation possible
- RPC/literal is a subset of Document/literal
  - WS-I allows both
- Tools meanwhile agreed on document / wrapped
  - Convention to have operation name as child to <soap:body>

SOAP (In)Efficiency



Web Service Middleware Communication



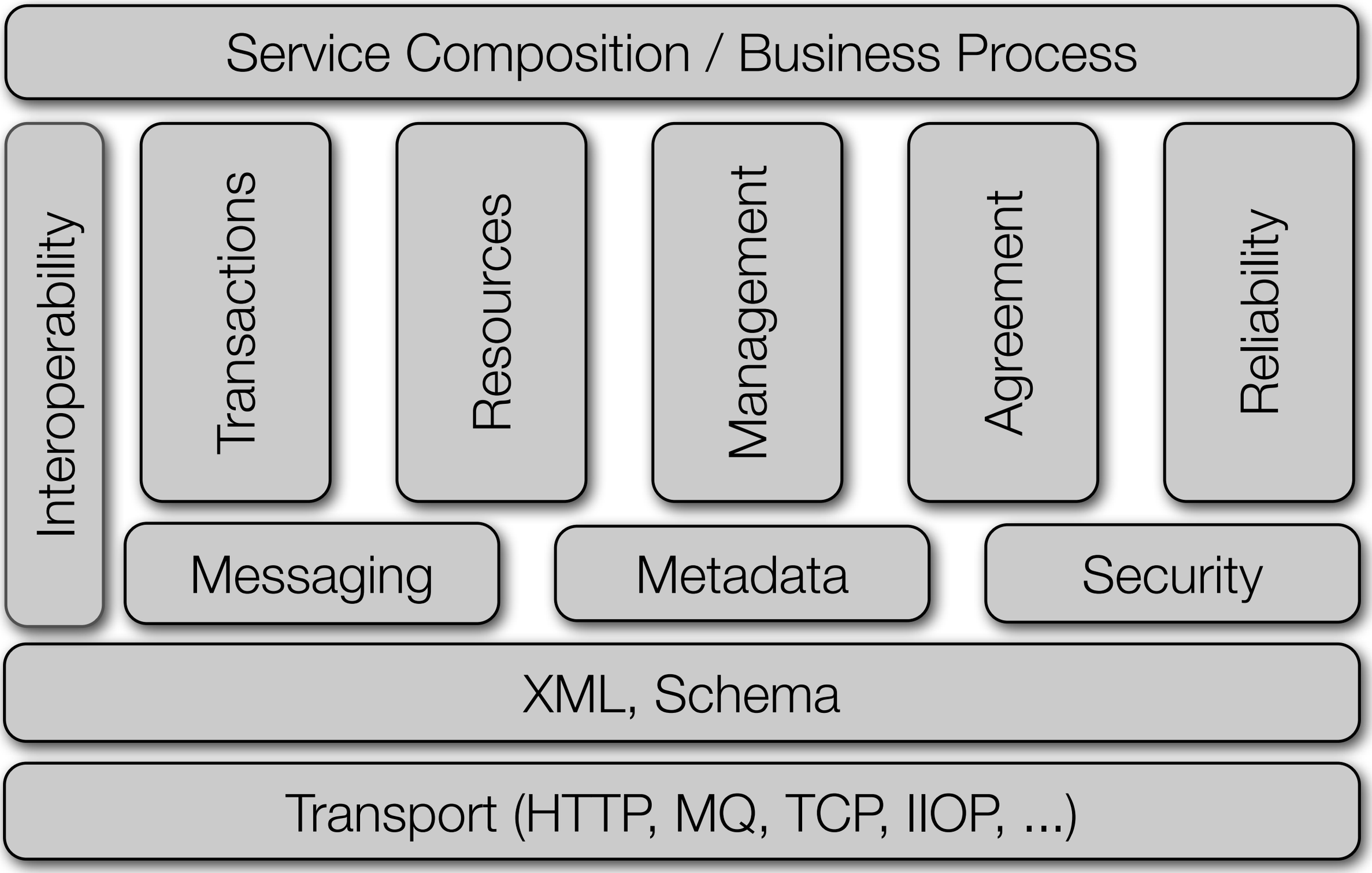
WS-* Standards

- Everything on-top-of SOAP, WSDL, and UDDI
- Huge number of documents, topics, and versions
- Different organizations with different influence and power
- Concurrent specifications, competing implementations

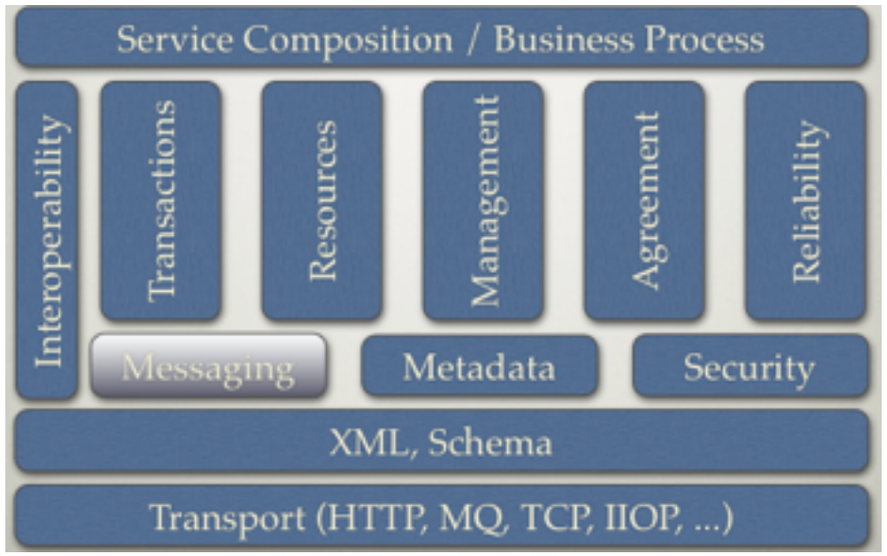
innoq.com

- Microsoft with / against IBM against the rest

Web Service Specification Landscape



Asynchronous Messaging

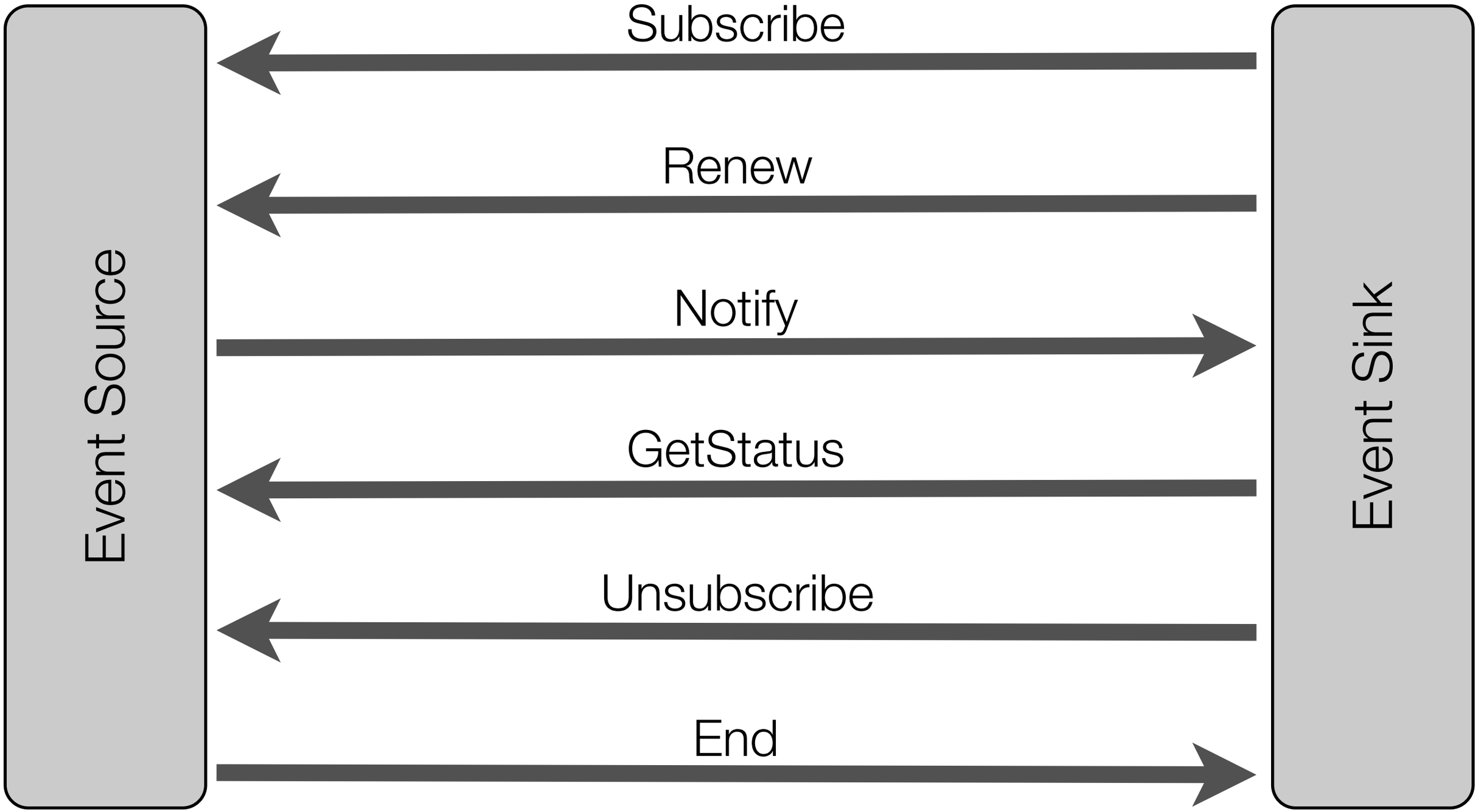


- WS-Notification 1.2 (IBM)
  - WS-BaseNotification
  - WS-BrokeredNotification
- WS-Eventing (IBM, MS)

C	IBM, Tibco, HP
S	OASIS Standard
I	ETTK, Axis, ...

C	IBM, Tibco, MS
S	W3C Member Sub.
I	ETTK, Axis, ...

Push Notification With Web Services



WS-Notification

- **Web Service Base Notification 1.3**
  - Interfaces for consumers and producers
- **Web Services Topics 1.3 (WS-Topics)**
  - Mechanisms to organize and categorize items of interest for subscription
- **Web Services Brokered Notification 1.3**
  - Interface for Notification Broker

WS-BaseNotification

- Different roles with their endpoints -
  producer, consumer, subscription manager, subscriber
- Two ways of notification (no response expected)
  - Producer sends raw application-specific content
  - Producer send special Notify message
    - Subscription reference
    - Topic and topic dialect
    - Producer reference

Notify Message Infoset

```
...
<wsnt:Notify>
  <wsnt:NotificationMessage>
    <wsnt:SubscriptionReference>
      wsa:EndpointReferenceType
    </wsnt:SubscriptionReference> ?
    <wsnt:Topic Dialect="xsd:anyURI">
      {any} ?
    </wsnt:Topic>?
    <wsnt:ProducerReference>
      wsa:EndpointReferenceType
    </wsnt:ProducerReference> ?
    <wsnt:Message>
      {any}
    </wsnt:Message>
  </wsnt:NotificationMessage> +
  {any} *
</wsnt:Notify>
```

SOAP Notify Example

```
<s:Envelope><s:Header>
    <wsa:Action>
      http://docs.oasis-open.org/wsn/bw-2/NotificationConsumer/Notify
    </wsa:Action>
    ...
</s:Header><s:Body>
    <wsnt:Notify><wsnt:NotificationMessage>
      <wsnt:SubscriptionReference>
        <wsa:Address>http://www.example.org/SubManager</wsa:Address>
      </wsnt:SubscriptionReference>
      <wsnt:Topic Dialect=
        "http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple">
        npex:SomeTopic
      </wsnt:Topic>
      <wsnt:ProducerReference>
        <wsa:Address>http://www.example.org/NotiProd</wsa:Address>
      </wsnt:ProducerReference>
      <wsnt:Message>
        <npex:NotifyContent>exampleNotifyContent</npex:NotifyContent>
      </wsnt:Message>
    <wsnt:NotificationMessage></wsnt:Notify>
</s:Body></s:Envelope>
```

WS-Notification Subscription

- Several options for subscription
  - Notification consumer EPR
  - Boolean filter expressions
    - Order and timing for tests defined by producer
    - Topic, or XPath expression to check message content
  - Initial termination time
  - Subscription policy (e.g. # of messages)
  - Indicator for raw subscription
- Subscription response with EPR and termination time

Subscription Example

```
<s:Envelope><s:Header>
  <wsa:Action>
    http://docs.oasis-open.org/wsn/bw- 2/NotificationProducer/SubscribeRequest
  </wsa:Action>
  ...
</s:Header><s:Body>
  <wsnt:Subscribe>
    <wsnt:ConsumerReference>
      <wsa:Address>http://www.example.org/NotificationConsumer</wsa:Address>
    </wsnt:ConsumerReference>
    <wsnt:Filter>
      <wsnt:TopicExpression
          Dialect="http://docs.oasis-open.org/wsn/t-1/TopicExpression/Simple">
        npex:SomeTopic
      </wsnt:TopicExpression>
      <wsnt:MessageContent
          Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
        boolean(ncex:Producer="15")
      </wsnt:MessageContent>
    </wsnt:Filter>
    <wsnt:InitialTerminationTime>
      2005-12-25T00:00:00.00000Z
    </wsnt:InitialTerminationTime>
  </wsnt:Subscribe>
</s:Body></s:Envelope>
```

WS-Notification vs. WS-Eventing

- WS-Notification features
  - Support for small devices
    (restricted set of mandatory features)
  - Support for direct and brokered notification
  - Transformation and aggregation of Topics
  - Runtime metadata (e.g. available subscription types)
  - Broker federations
  - Based on *WS-ResourceProperties* and *WS-ResourceLifetime* (from WSRF)

Implementations

- SOAP, WSDL, UDDI - Endless number of implementations
- WS-* specifications
  - Apache projects (Axis, WSFX, Muse, ...)
  - IBM Emerging Technologies Toolkit (ETTK)
  - Microsoft Web Services Enhancements toolkit (WSE) / Windows Communication Foundation (WCF)
  - Sun Java Java Web Services Developer Pack (WSDP)
  - Verisign Trust Service Integration Kit (TSIK)

W3C Standardization Body

- Founded 1994 by Tim Berners-Lee
- Internet standards (HTTP, HTML, XML)
- All ratified standards must be royalty-free
- Standardization track:
  - Working group note and working draft
  - Candidate recommendation
  - Proposed / W3C recommendation
- Member submission

WS-I Standardization Body

- Web Services Interoperability Organization
- Founded in 2002 by Microsoft, IBM and others
- Clarifies ambiguities and restricts WS specifications
- Profiles for basic specs (SOAP, WSDL, UDDI) and Security
- Conformance test tool chain (Java, C#)

WS-I Details

- WS-I profiles
  - Basic Profile Working Group
  - Basic Security Profile Working Group
  - Requirements gathering, sample scenarios, testing tools, XML schema issues
- Clarifications: Missing details, interop problems, attachments, SOAP binding, security token, ...
- Sun: “Shadow government for standards”

Enterprise Service Bus

- Term coined by Dave Chappell in his book (2004)
  - Part of SOA concepts for service orchestration
- Primary task as message broker, which transforms and maps data
  - Implementations based on XML-powered MOM
    - Application developer formulates XSLT, based on XPath
- Support for debugging and testing of transformation steps

XML/XSLT Support

	XML parser	XSLT engine	Pluggable
BEA Systems AquaLogic Service Bus 2.1	Proprietary	Proprietary	Y
Cape Clear Software ESB 6.5	Proprietary	Proprietary	N
Fiorano Software SOA 2006 Platform 3.7	Xerces	Xalan	Y
IBM WebSphere Message Broker 6.0	XML4C	Xalan	N
Oracle SOA Suite	Xerces	Xalan	Y
Software AG Enterprise Service Integrator 2.1	Xerces	Xalan	N
Sonic Software SOA Suite 6.1	Xerces	Saxon	N
TIBCO Software BusinessWorks 5.3	Proprietary	Proprietary	N

Y=Yes, N=No

Discussion

- XML for RPC / Messaging Middleware
  - Pros ?
  - Cons ?
- CORBA and XML; conflict or cooperation?
  - http://www.omg.org/news/whitepapers/watsonwp.htm