INFOH509 XML & Web Technologies

Lecture 11

BIGWS-* WEB SERVICES

SERVICES

Two competing technology stacks

- **Big Web Services (WS-*)**
  - Various (complex) protocols on top of HTTP (SOAP, UDDI, WSDL, WS-Addressing, …)
  - Is mostly used to implement RPC-style services, but can be used to implement any of the three
  - Lots of standards! Primarily meant to create web services that involve more than 2 peers.

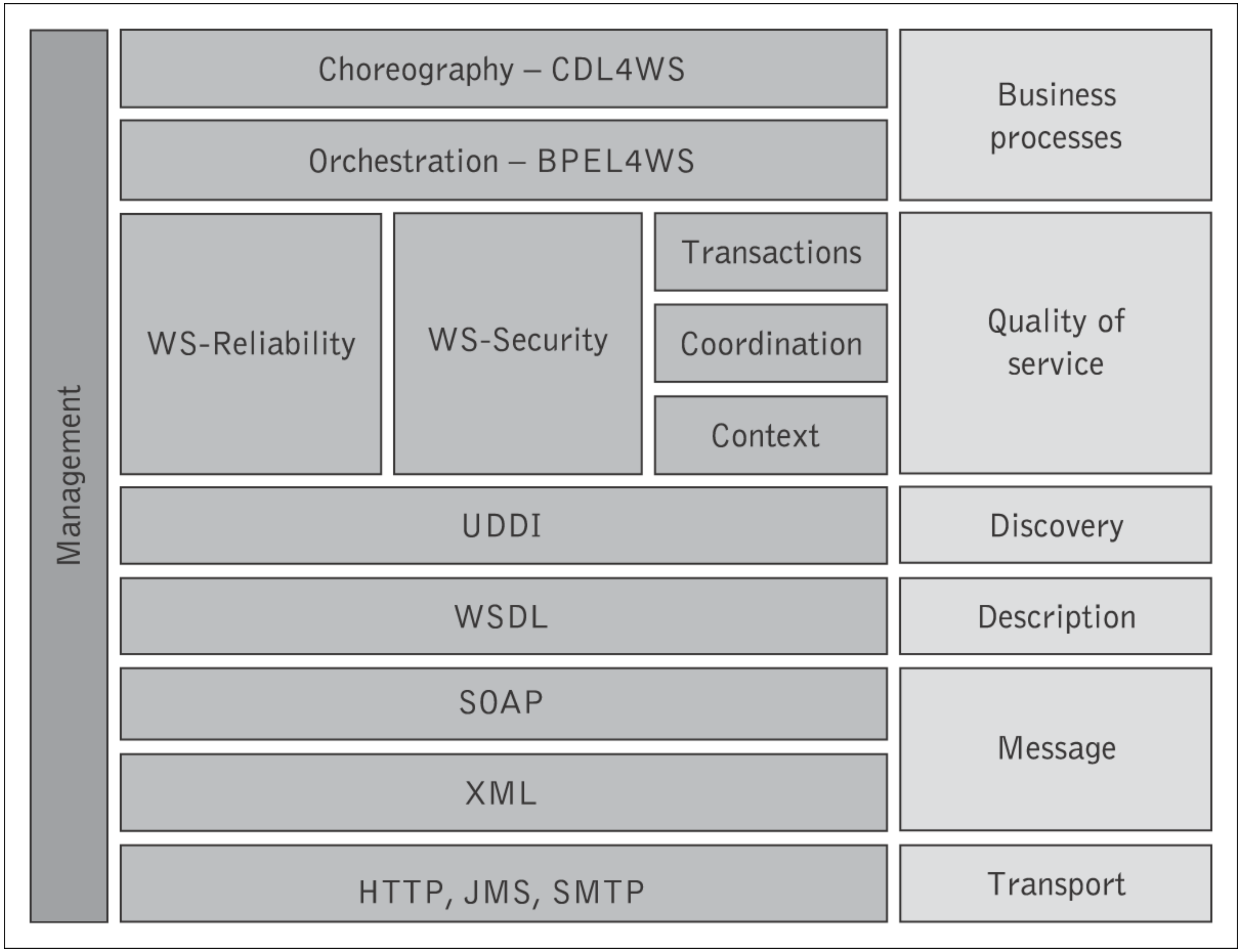
- **RESTful Web Services**
  - Use ONLY HTTP and standard media types
  - Restricted to Resource-style services
  - Conceptually simpler, but mainly restricted to web services that are limited to two endpoints

Two competing technology stacks

- **Big Web Services (WS-*)** $\longrightarrow$ **This lecture**
  - Various (complex) protocols on top of HTTP (SOAP, UDDI, WSDL, WS-Addressing, …)
  - Is mostly used to implement RPC-style services, but can be used to implement any of the three
  - Lots of standards! Primarily meant to create web services that involve more than 2 peers.

- **RESTful Web Services** $\longrightarrow$ **Previous lecture**
  - Use ONLY HTTP and standard media types
  - Restricted to Resource-style services
  - Conceptually simpler, but mainly restricted to web services that are limited to two endpoints

WS-* Technology Stack



Web Services Standards Overview

Interoperability Issues

Basic Profile
1.1
WS-I
Final Specification

▲ **Basic Profile** – The Basic Profile 1.1 provides implementation guidelines for how related set of non-proprietary Web Service specifications should be used together for best interoperability.

Basic Profile
1.2
WS-I
Working Group Draft

▲ **Basic Profile** – The Basic Profile 1.2 builds on Basic Profile 1.1 by incorporating Basic Profile 1.1 errata, requirements from Simple SOAP Binding Profile 1.0, and adding support for WS-Addressing and MTOM.

Basic Profile
2.0
WS-I
Working Group Draft

▲ **Basic Profile** – The Basic Profile 2.0 is an update of WS-I BP that includes a profile of SOAP 1.2.

Attachments Profile
1.0
WS-I
Final Specification

▲ **Attachments Profile** – The Attachment Profile 1.0 complements the Basic Profile 1.1 to add support for interoperable SOAP Messages with attachments-based Web Services.

Simple SOAP Binding Profile
1.0
WS-I
Final Specification

▲ **Simple SOAP Binding Profile** – The Simple SOAP Binding Profile consists of those Basic Profile 1.0 requirements related to the serialization of the envelope and its representation in the message.

Basic Security Profile
1.0
WS-I
Board Approval Draft

▲ **Basic Security Profile** defines the WS-I Basic Security Profile 1.0, based on a set of non-proprietary Web services specifications, along with clarifications and amendments to those specifications which promote interoperability.

REL Token Profile
1.0
WS-I
Working Group Draft

▲ **REL Token Profile** is based on a non-proprietary Web services specification, along with clarifications and amendments to that specification which promote interoperability.

SAML Token Profile
1.0
WS-I
Working Group Draft

▲ **SAML Token Profile** is based on a non-proprietary Web services specification, along with clarifications and amendments to that specification which promote interoperability.

Conformance Claim Attachment Mechanism (CCAM)
1.0
WS-I
Final Specification

▲ **Conformance Claim Attachment Mechanism (CCAM)** catalogues mechanisms that can be used to attach WS-I Profile Conformance Claims to Web services artefacts (e.g., WSDL descriptions, UDDI registries).

Reliable Asynchronous Messaging Profile (RAMP)
1.0
WS-I
Working Draft

▲ **Reliable Asynchronous Messaging Profile (RAMP)** is a profile, in the fashion of the WS-I profiles, that enables, among other things, basic B2B integration scenarios using Web services technologies.

Standards Bodies

OASIS The **Organization for the Advancement of Structured Information Standards (OASIS)** is a non-for-profit, international consortium that drives the development, convergence, and adoption of e-business standards. The consortium produces more Web services standards than any other organization along with standards for security, e-business, and standardization efforts in the public sector and for application-specific markets. Founded in 1993, OASIS has more than 4,000 participants representing over 600 organizations and individual members in 100 countries.

W3C The **World Wide Web Consortium (W3C)** was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has over 350 Member organizations from all over the world and has earned international recognition for its contributions to the growth of the Web. W3C is designing the infrastructure, and defining the architecture and the core technologies for Web services. In September 2000, W3C started the XML Protocol Activity to address the need for an XML-based protocol for application-to-application messaging. In January 2002, the Web Services Activity was launched, subsuming the XML Protocol Activity and extending its scope.

WS-I The **Web Services Interoperability Organization (WS-I)** is an open industry organization chartered to promote Web services interoperability across platforms, operating systems and programming languages. The organization's diverse community of Web services leaders helps customers to develop interoperable Web services by providing guidance, recommended practices and supporting resources. Specifically, WS-I creates, promotes and supports generic protocols for the interoperable exchange of messages between Web services.

I E T F The **Internet Engineering Task Force (IETF)** is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet.

Version 3.0 · February 2007

Business Process Specifications

Business Process Execution Language for Web Services 1.1
(BPEL4WS) · 1.1 · BEA Systems, IBM, Microsoft, SAP, Siebel Systems · OASIS-Standard

▲ **Business Process Execution Language for Web Services 1.1** (BPEL4WS) provides a language for the formal specification of business processes and business interaction protocols using Web Services.

WS-Choreography Model Overview
1.0 · W3C
Working Draft

▲ **WS-Choreography Model Overview** defines the format and structure of the (SOAP) messages that are exchanged, and the sequence and conditions in which the messages are exchanged.

Web Service Choreography Interface
(WSCI) · 1.0 · W3C
Sun Microsystems, SAP, BEA Systems and Intalio · Note

▲ **Web Service Choreography Interface** (WSCI) describes how Web Service operations can be choreographed in the context of a message exchange in which the Web Service participates.

Web Service Choreography Description Language
(CDL4WS) · 1.0 · W3C
Candidate Recommendation

▲ **Web Service Choreography Description Language** (CDL4WS) is to specify a declarative, XML based language that defines from a global viewpoint the common and complementary observable behaviour, where message exchanges occur, and when the jointly agreed ordering rules are satisfied.

Business Process Execution Language for Web Services 2.0
(BPEL4WS) · 2.0
OASIS, BEA Systems, IBM, Microsoft, SAP, Siebel Systems · Committee Draft

▲ **Business Process Execution Language for Web Services 2.0** (BPEL4WS) provides a language for the formal specification of business processes and business interaction protocols using Web Services.

Business Process Management Language (BPML)
1.1
BPMI.org
Final Draft

▲ **Business Process Management Language** (BPML) provides a meta-language for expressing business processes and supporting entities.

XML Process Definition Language (XPDL)
2.0
Final

▲ **XML Process Definition Language** (XPDL) provides an XML file format that can be used to interchange process models between tools.

Management Specifications

Management Using Web Services (WSDM-MUWS)
1.0
OASIS
OASIS-Standard

▲ **Web Service Distributed Management:** Management Using Web Services (WSDM-MUWS) defines how an IT resource connected to a network provides manageability interfaces such that the IT resource can be managed locally and from remote locations using Web services technologies.

Management Of Web Services (WSDM-MOWS)
1.0
OASIS
OASIS-Standard

▲ **Web Service Distributed Management:** Management Of Web Services (WSDM-MOWS) addresses management of the components that form the network, the Web services endpoints, using Web services protocols.

WS-Management
AMD, Dell, Intel, Microsoft and Sun Microsystems
Published Specification

▲ **WS-Management** describes a general SOAP-based protocol for managing systems such as PCs, servers, devices, Web services and other applications, and other manageable entities.

Service Modeling Language
IBM, BEA, BMC, Cisco, Dell, HP, Intel, Microsoft, Sun
Draft Specification

▲ **Service Modeling Language** (SML) is used to model complex IT services and systems, including their structure, constraints, policies, and best practices.

Presentation Specifications

Web Services for Remote Portlets (WSRP)
2.0
OASIS
Committee Draft

▶ **Web Services for Remote Portlets** (WSRP) defines a set of interfaces and related semantics which standardize interactions with components providing user-facing markup, including the processing of user interactions with that markup.

Metadata Specifications

WS-Policy
1.5
W3C
Working Draft

▲ **WS-Policy** describes the capabilities and constraints of the policies on intermediaries and endpoints (e.g. business rules, required security tokens, supported encryption algorithms, privacy rules).

WS-PolicyAssertions
1.1
BEA Systems, IBM, Microsoft, SAP
Public Draft

▲ **WS-PolicyAssertions** provides an initial set of assertions to address some common needs of Web Services applications.

WS-PolicyAttachment
1.2
W3C
W3C Member Submission

▲ **WS-PolicyAttachment** defines two general-purpose mechanisms for associating policies with the subjects to which they apply; the policies may be defined as part of existing metadata about the subject or the policies may be defined independently and associated through an external binding to the subject.

WS-Discovery
Microsoft, BEA Systems, Canon, Intel and webMethods
Draft

▲ **WS-Discovery** defines a multicast discovery protocol for dynamic discovery of services on ad-hoc and managed networks.

WS-MetadataExchange
1.1
BEA Systems, Computer Associates, IBM, Microsoft, SAP, Sun Microsystems and webMethods
Public Draft

▲ **WS-MetadataExchange** enables a service to provide metadata to others through a Web services interface. Given only a reference to a Web service, a user can access a set of WSDL/SOAP operations to retrieve the metadata that describes the service.

Universal Description, Discovery and Integration (UDDI)
3.0.2
OASIS
OASIS-Standard

▲ **Universal Description, Discovery and Integration (UDDI)** defines a set of services supporting the description and discovery of businesses, organizations, and other Web services providers, the Web services they make available, and the technical interfaces which may be used to access those services.

Web Service Description Language 2.0 SOAP Binding
2.0
W3C · Working Draft

▲ **Web Service Description Language SOAP Binding** describes the concrete details for using WSDL 2.0 in conjunction with SOAP 1.1 protocol.

Web Service Description Language 2.0 Core
2.0
W3C
Candidate Recommendation

▲ **Web Service Description Language 2.0 Core** is an XML-based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.

Web Service Description Language 1.1
1.1
W3C
Note

▲ **Web Service Description Language 1.1** is an XML-based language for describing Web services and how to access them. It specifies the location of the service and the operations (or methods) the service exposes.

Reliability Specifications

WS-ReliableMessaging
1.1
OASIS
Committee Draft

▲ **WS-ReliableMessaging** describes a protocol that allows Web services to communicate reliable in the presence of software component, system, or network failures. It defines a SOAP binding that is required for interoperability.

WS-Reliable Messaging Policy Assertion (WS-RM Policy)
1.1
OASIS
Committee Draft

▲ **Web Services ReliableMessaging Policy Assertion** (WS-RM Policy) describes a domain-specific policy assertion for WS-ReliableMessaging that that can be specified within a policy alternative as defined in WS-Policy Framework.

WS-Reliability
1.1
OASIS
OASIS-Standard

▲ **WS-Reliability** is a SOAP-based protocol for exchanging SOAP messages with guaranteed delivery, no duplicates, and guaranteed message ordering. WS-Reliability is defined as SOAP header extensions and is independent of the underlying protocol. This specification contains a binding to HTTP.

Security Specifications

WS-Security
1.1
OASIS
OASIS-Standard

▲ **WS-Security** is a communications protocol providing a means for applying security to Web Services.

WS-SecurityPolicy
1.1
IBM, Microsoft, RSA Security, VeriSign
Public Draft

▲ **WS-SecurityPolicy** defines how to describe policies related to various features defined in the WS-Security specification.

WS-Security: SOAP Message Security
1.1
OASIS
Public Review Draft

▲ **WS-Security: SOAP Message Security** describes enhancements to SOAP messaging to provide message integrity and confidentiality. Specifically, this specification provides support for multiple security token formats, trust domains, signature formats and encryption technologies. The token formats and semantics for using these are defined in the associated profile documents.

WS-Security: Username Token Profile
1.1
OASIS
Public Review Draft

▲ **WS-Security: Username Token Profile** describes how a Web Service consumer can supply a Username Token as a means of identifying the requestor by username, and optionally using a password (or shared secret, etc.) to authenticate that identity to the Web Service producer.

WS-Security: Kerberos Binding
1.0
Microsoft, IBM, OASIS
Working Draft

▲ **WS-Security: Kerberos Binding** defines how to encode Kerberos tickets and attach them to SOAP messages. As well, it specifies how to add signatures and encryption to the SOAP message, in accordance with WS-Security, which uses and references the Kerberos tokens.

WS-Federation
1.0
IBM, Microsoft, BEA Systems, RSA Security, and VeriSign
Initial Draft

▲ **WS-Federation** describes how to manage and broker the trust relationships in a heterogeneous federated environment including support for federated identities.

WS-Security: SAML Token Profile
1.1
OASIS
Public Review Draft

▲ **WS-Security: SAML Token Profile** defines the use of Security Assertion Markup Language (SAML) v1.1 assertions in the context of WSS: SOAP Message Security including for the purpose of securing SOAP messages and SOAP message exchanges.

WS-Trust
BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology · Initial Draft

▲ **WS-Trust** describes a framework for trust models that enables Web Services to securely interoperate. It uses WS-Security base mechanisms and defines additional primitives and extensions for security token exchange to enable the issuance and dissemination of credentials within different trust domains.

WS-Security: X.509 Certificate Token Profile
1.1
OASIS
Public Review Draft

▲ **WS-Security: X.509 Certificate Token Profile** describes the use of the X.509 authentication framework with the WS-Security: SOAP Message Security specification.

WS-SecureConversation
BEA Systems, Computer Associates, IBM, Layer 7 Technologies, Microsoft, Netegrity, Oblix, OpenNetwork, Ping Identity Corporation, Reactivity, RSA Security, VeriSign and Westbridge Technology · Public Draft

▲ **WS-SecureConversation** specifies how to manage and authenticate message exchanges between parties including security context exchange and establishing and deriving session keys.

Transaction Specifications

WS-Coordination
1.1
OASIS
Working Draft

▲ **WS-Coordination** describes an extensible framework for providing protocols that coordinate the actions of distributed applications.

WS-Business Activity
1.1
OASIS
Working Draft

▲ **WS-Business Activity** provides the definition of the business activity coordination type that is to be used with the extensible coordination framework described in the WS-Coordination specification.

WS-Atomic Transaction
1.1
OASIS
Committee Draft

▲ **WS-Atomic Transaction** defines protocols that enable existing transaction processing systems to wrap their proprietary protocols and interoperate across different hardware and software vendors.

WS-Composite Application Framework (WS-CAF)
1.0 · Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems
Committee Specification

▲ **WS-Composite Application Framework** (WS-CAF) is a collection of three specifications aimed at solving problems that arise when multiple Web Services are used in combination. It proposes standard, interoperable mechanisms for managing shared context and ensuring business processes achieve predictable results and recovery from failure.

WS-Context (WS-CTX)
1.0
Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems
Committee Draft

▲ **WS-Context** (WS-CTX) is intended as a lightweight mechanism for allowing multiple Web Services to share a common context.

WS-Coordination Framework (WS-CF)
1.0 · Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems
Committee Draft

▲ **WS-Coordination Framework** (WS-CF) allows the management and coordination in a Web Services interaction of a number of activities related to an overall application.

WS-Transaction Management (WS-TXM)
1.0 · Arjuna Technologies, Fujitsu, IONA, Oracle and Sun Microsystems
Committee Draft

▲ **WS-Transaction Management** (WS-TXM) defines a core infrastructure service consisting of a Transaction Service for Web Services.

Resource Specifications

Web Services Resource Framework (WSRF)
1.2
OASIS
OASIS-Standard

▲ **Web Services Resource Framework** (WSRF) defines a family of specifications for accessing stateful resources using Web Services.

WS-BaseFaults (WSRF)
1.2
OASIS
Working Draft

▲ **WS-BaseFaults** (WSRF) defines a base set of information that may appear in fault messages. WS-BaseFaults defines an XML schema type for base faults, along with rules for how this base fault type is used and extended by Web Services.

WS-ServiceGroup (WSRF)
1.2
OASIS
Working Draft

▲ **WS-ServiceGroup** (WSRF) defines a means by which Web Services and WS-Resources can be aggregated or grouped together for a domain specific purpose.

WS-ResourceProperties
1.2
OASIS
Working Draft

▲ **WS-ResourceProperties** specifies the means by which the definition of the properties of a WS-Resource may be declared as part of the Web Service interface. The declaration of the WS-Resource properties represents a projection of or a view on the WS-Resource state.

WS-ResourceLifetime
1.2
OASIS
Working Draft

▲ **WS-ResourceLifetime** is to standardize the terminology, concepts, message exchanges, WSDL and XML needed to monitor the lifetime of, and destroy WS-Resources. Additionally, it defines resource properties that may be used to inspect and monitor the lifetime of a WS-Resource.

WS-Transfer
W3C
W3C Member Submission

▲ **WS-Transfer** describes a general SOAP-based protocol for accessing XML representations of Web service-based resources.

Resource Representation SOAP Header Block (RRSHB)
W3C
Recommendation

▲ **Resource Representation SOAP Header Block** (RRSHB) complements MTOM by defining mechanisms for describing and conveying non-XML resource representations in a SOAP 1.2 message.

Messaging Specifications

WS-Notification
1.3
OASIS
OASIS-Standard

▶ **WS-Notification** is a family of related white papers and specifications that define a standard Web services approach to notification using a topic-based publish/subscribe pattern.

WS-BrokeredNotification
1.3
OASIS
OASIS-Standard

▶ **WS-BrokeredNotification** defines the interface for the NotificationBroker. A NotificationBroker is an intermediary, which, among other things, allows publication of messages from entities that are not themselves service providers.

WS-BaseNotification
1.3
OASIS
OASIS-Standard

▶ **WS-BaseNotification** standardizes the terminology, concepts, operations, WSDL and XML needed to express the basic roles involved in Web services publish and subscribe for notification message exchange.

WS-Eventing
W3C
Public Draft

▶ **WS-Eventing** defines a baseline set of operations that allow Web services to provide asynchronous notifications to interested parties.

WS-Addressing – Core
1.0
W3C
Recommendation

▶ **WS-Addressing – Core** provides transport-neutral mechanisms to address Web services and messages. This specification defines XML elements to identify Web service endpoints and to secure end-to-end endpoint identification in messages.

WS-Enumeration
Systinet, Microsoft, Sonic Software, BEA Systems and Computer Associates
Public Draft

▶ **WS-Enumeration** describes a general SOAP-based protocol for enumerating a sequence of XML elements that is suitable for traversing logs, message queues, or other linear information models.

WS-Topics
1.3
OASIS
OASIS-Standard

▶ **WS-Topics** defines three topic expression dialects that can be used as subscription expressions in subscribe request messages and other parts of the WS-Notification system.

WS-Addressing – WSDL Binding
1.0
W3C
Candidate Recommendation

▶ **WS-Addressing – WSDL Binding** defines how the abstract properties defined in Web Services Addressing – Core are described using WSDL.

WS-Addressing – SOAP Binding
1.0
W3C
Recommendation

▶ **WS-Addressing – SOAP Binding** provides transport-neutral mechanisms to address Web services and messages.

SOAP

SOAP
1.2
W3C
Recommendation

SOAP Message Transmission Optimization Mechanism (MTOM)
1.0 · W3C
Recommendation

▶ **SOAP Message Transmission Optimization Mechanism** describes an abstract feature for optimizing the transmission and/or wire format of a SOAP message.

SOAP
1.1
W3C
Note

▶ **SOAP** is a lightweight, XML-based protocol for exchange of information in a decentralized, distributed environment.

XML Specifications

XML 1.1
1.1
W3C
Recommendation

▶ **XML** – Extensible Markup Language is a pared-down version of SGML, designed especially for Web documents. It allows one to create own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

XML 1.0
1.0
W3C
Recommendation

▶ **XML** – Extensible Markup Language is a pared-down version of SGML, designed especially for Web documents. It allows one to create own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations.

Namespaces in XML
1.1
W3C
Recommendation

▶ **Namespaces in XML** provides a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by IRI references.

XML Information Set
1.0
W3C
Recommendation

▶ **XML Information Set** is an abstract data set to provide a consistent set of definitions for use in other specifications that need to refer to the information in a well-formed XML document.

XML Schema
1.1
W3C
Working Draft

▶ **XML Schema** – XML Schema Definition Language is an XML language for describing and constraining the content of XML documents.

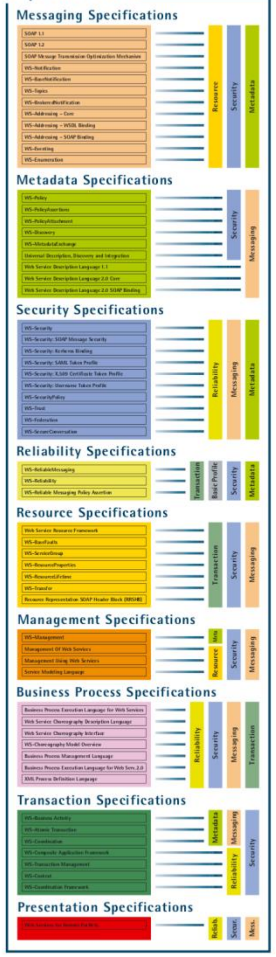
XML binary Optimized Packaging (XOP)
1.0
W3C
Recommendation

▶ **XML binary Optimized Packaging** (XOP) is an XML language for describing and constraining the content of XML documents.

Describing Media Content of Binary Data in XML (DMCBDX)
W3C
Note

▶ **Describing Media Content of Binary Data in XML** (DMCBDX) specifies how to indicate the content-type associated with binary element content in an XML document and to specify, in XML Schema, the expected content-type(s) associated with binary element content.

Dependencies



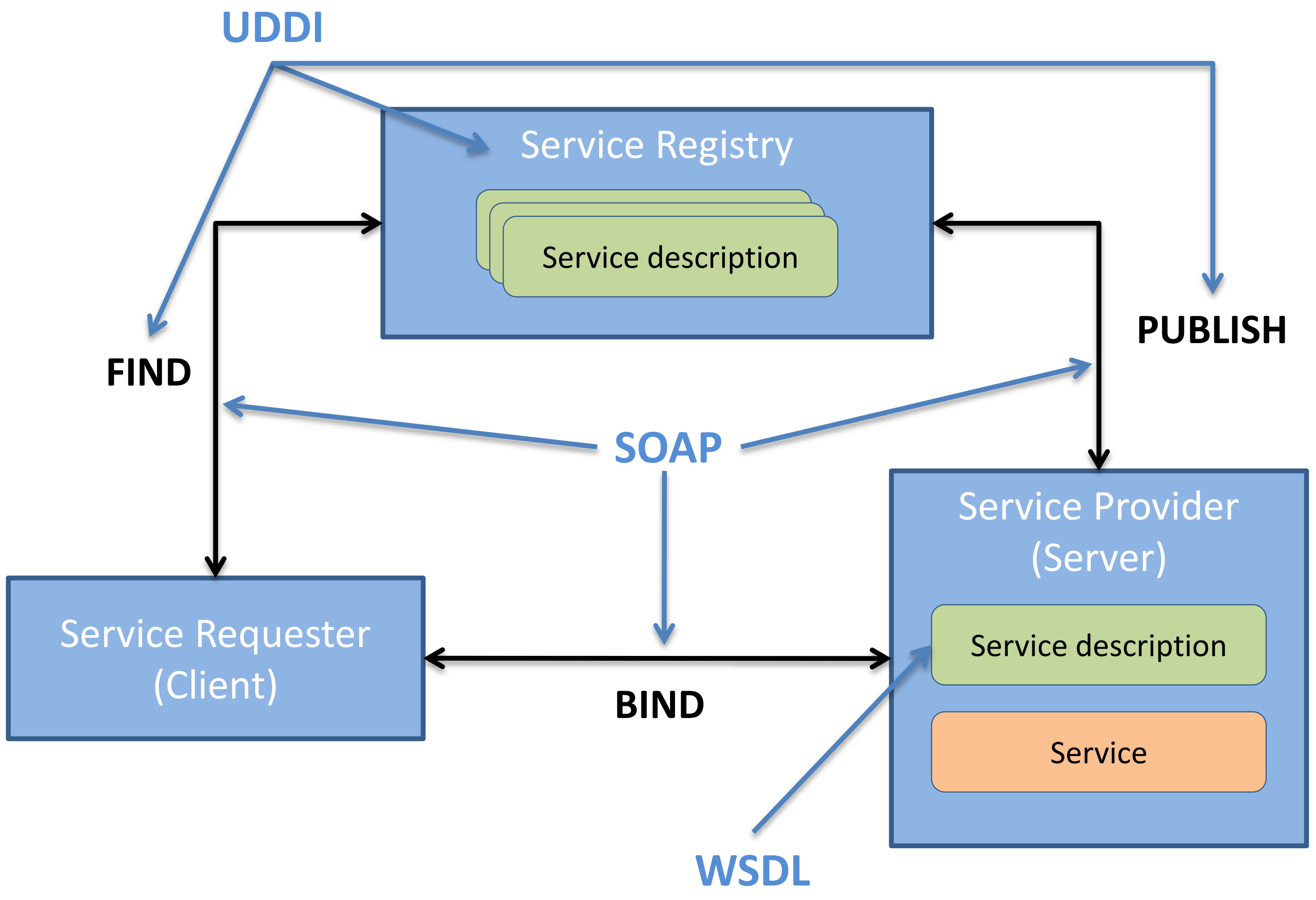
Messaging Specifications
SOAP 1.1
SOAP 1.2
SOAP Message Transmission Optimization Mechanism
WS-Notification
WS-BaseNotification
WS-Topics
WS-BrokeredNotification
WS-Addressing – Core
WS-Addressing – WSDL Binding
WS-Addressing – SOAP Binding
WS-Eventing
WS-Enumeration
Resource
Security
Metadata
Metadata Specifications
WS-Policy
WS-PolicyAssertions
WS-PolicyAttachment
WS-Discovery
WS-MetadataExchange
Universal Description, Discovery and Integration
Web Service Description Language 1.1
Web Service Description Language 2.0 Core
Web Service Description Language 2.0 SOAP Binding
Security
Messaging
Security Specifications
WS-Security
WS-Security: SOAP Message Security
WS-Security: Kerberos Binding
WS-Security: SAML Token Profile
WS-Security: X.509 Certificate Token Profile
WS-Security: Username Token Profile
WS-SecurityPolicy
WS-Trust
WS-Federation
WS-SecureConversation
Reliability
Messaging
Metadata
Reliability Specifications
WS-ReliableMessaging
WS-Reliability
WS-Reliable Messaging Policy Assertion
Transaction
Basic Profile
Security
Metadata
Resource Specifications
Web Service Resource Framework
WS-BaseFaults
WS-ServiceGroup
WS-ResourceProperties
WS-ResourceLifetime
WS-Transfer
Resource Representation SOAP Header Block (RRSHB)
Transaction
Security
Messaging
Management Specifications
WS-Management
Management Of Web Services
Management Using Web Services
Service Modeling Language
Meta
Resource
Security
Messaging
Business Process Specifications
Business Process Execution Language for Web Services
Web Service Choreography Description Language
Web Service Choreography Interface
WS-Choreography Model Overview
Business Process Management Language
Business Process Execution Language for Web Serv. 2.0
XML Process Definition Language
Reliability
Security
Messaging
Transaction
Transaction Specifications
WS-Business Activity
WS-Atomic Transaction
WS-Coordination
WS-Composite Application Framework
WS-Transaction Management
WS-Context
WS-Coordination Framework
Metadata
Messaging
Reliability
Security
Presentation Specifications
Web Services for Remote Portlets
Reliab.
Secur.
Mess.

innoQ

innoQ Deutschland GmbH
Halskestraße 17
D-40880 Ratingen
Phone +49 21 02 77 162-100
info@innoq.com · www.innoq.com

innoQ Schweiz GmbH
Gewerbestrasse 11
CH-6330 Cham
Phone +41 41 743 01 11

WS-* Message, Description & Discovery



- While SOAP & WSDL are frequently used, UDDI has never caught on.

Part I

SOAP

The history of SOAP

- Originally, SOAP was conceived as a minimalistic infrastructure to perform remote procedure calls (RPC) over the Web:
  - SOAP messages use XML as a format to exchange data between systems.
  - SOAP messages can be sent over HTTP (but also other protocols like SMTP) as a transport protocol. (This avoids system heterogeneity & firewall issues.)

- Currently, SOAP is no longer limited to RPC.

The history of SOAP (cont.)

- Originally proposed by Microsoft (then: the Object Access Protocol).
- Version 1.0 standardized by the W3C in 1999
- Currently at version 1.2

What is SOAP?

- SOAP = a messaging framework (much like HTTP)

- The SOAP specification covers the following four main areas:
  - **Message format:** A format for one-way communication describing how a message can be packed into an XML document.
  - **Processing model:** A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. Describes what parts of the messages should be read by whom and how to react in case of failure.
  - **Extensibility model:** How the basic message constructs can be extended with application-specific constructs
  - **Protocol binding framework:** A description of how SOAP messages can be transported using different protocols (e.g., HTTP, SMTP, …).
  - A set of conventions on how to turn an RPC call into a SOAP message and back. (Since version 1.1. SOAP is not restricted to the RPC programming model).

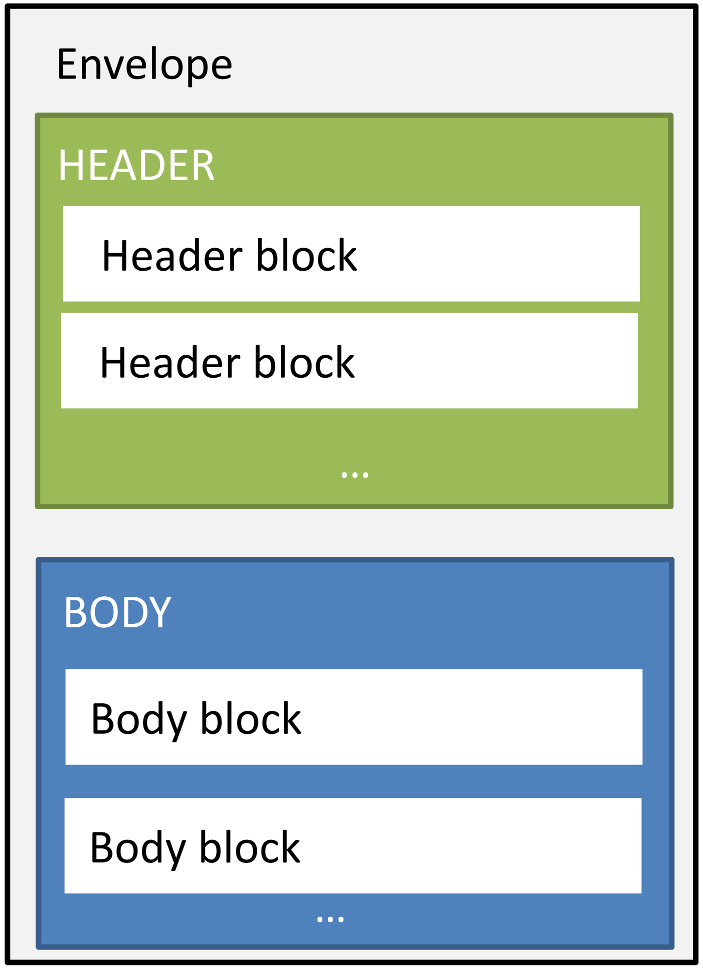
What is SOAP?

- SOAP = a messaging framework (much like HTTP)

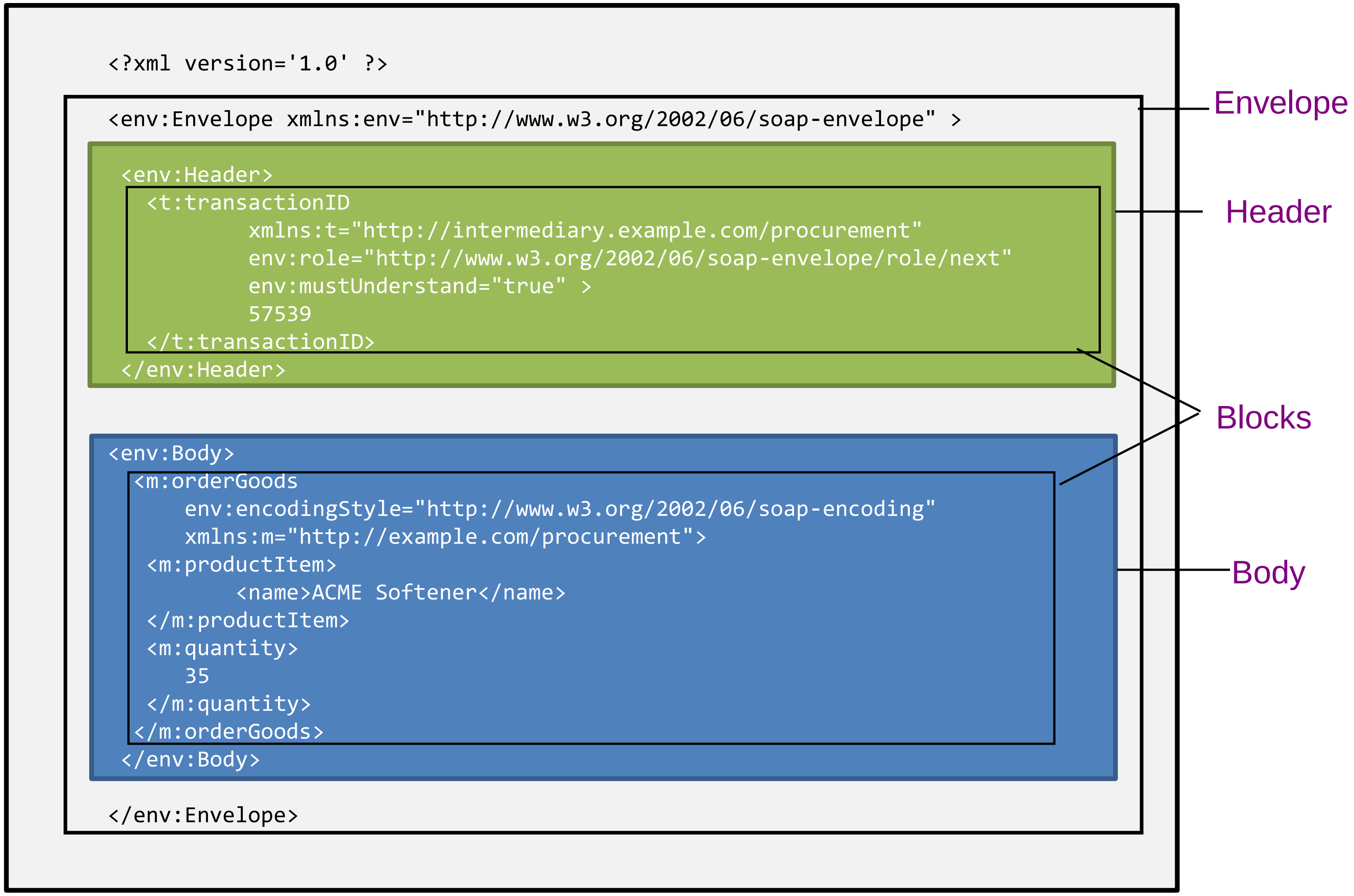
- The SOAP specification covers the following four main areas:
  - **Message format:** A format for one-way communication describing how a message can be packed into an XML document.
  - **Processing model:** A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. Describes what parts of the messages should be read by whom and how to react in case of failure.
  - **Extensibility model:** How the basic message constructs can be extended with application-specific constructs
  - **Protocol binding framework:** A description of how SOAP messages can be transported using different protocols (e.g., HTTP, SMTP, …).
  - A set of conventions on how to turn an RPC call into a SOAP message and back. (Since version 1.1. SOAP is not restricted to the RPC programming model).

The SOAP message structure

- SOAP Message
  = envelope in XML format
- Two parts:
  - Headers (optional)
    "The stickers"
    Contains meta data (security, transaction data, …)
  - Body
    "Application data"
- SOAP only specifies the structure of the message, not the semantics of header/body



An example SOAP message



```
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2002/06/soap-envelope" >
  <env:Header>
    <t:transactionID
            xmlns:t="http://intermediary.example.com/procurement"
            env:role="http://www.w3.org/2002/06/soap-envelope/role/next"
            env:mustUnderstand="true" >
            57539
    </t:transactionID>
  </env:Header>

 <env:Body>
   <m:orderGoods
       env:encodingStyle="http://www.w3.org/2002/06/soap-encoding"
       xmlns:m="http://example.com/procurement">
    <m:productItem>
           <name>ACME Softener</name>
    </m:productItem>
    <m:quantity>
       35
    </m:quantity>
   </m:orderGoods>
  </env:Body>

 </env:Envelope>
```

The SOAP header

- The header is intended as a generic place holder for information that is not necessarily application dependent (the application may not even be aware that a header was attached to the message).

- Typical uses of the header are: coordination information, identifiers (e.g., for transactions), security information (e.g., certificates)

- SOAP provides mechanisms to specify who should deal with headers and what to do with them. For this purpose it includes:
  - Role attribute: who should process that particular header block.
  - Boolean mustUnderstand attribute: indicates whether it is mandatory to process the header. If a header is directed at a node (as indicated by the actor attribute), the mustUnderstand attribute determines whether it is mandatory to do so.
  - SOAP 1.2 adds a relay attribute (forward header if not processed)

The SOAP body

- The SOAP body is the area of the SOAP message, where the application specific XML data (payload) being exchanged in the message is placed.

- The **<Body> element** must be present and is an immediate child of the envelope. It may contain a number of child elements, called body entries, but it may also be empty. The **<Body> element** contains either of the following:
  - **Application-specific data** is the information that is exchanged with a Web service. The SOAP <Body> is where the method call information and its related arguments are encoded. It is where the response to a method call is placed, and where error information can be stored.
  - A **fault message** is used only when an error occurs.

- A SOAP message may carry either application-specific data or a fault, but not both.

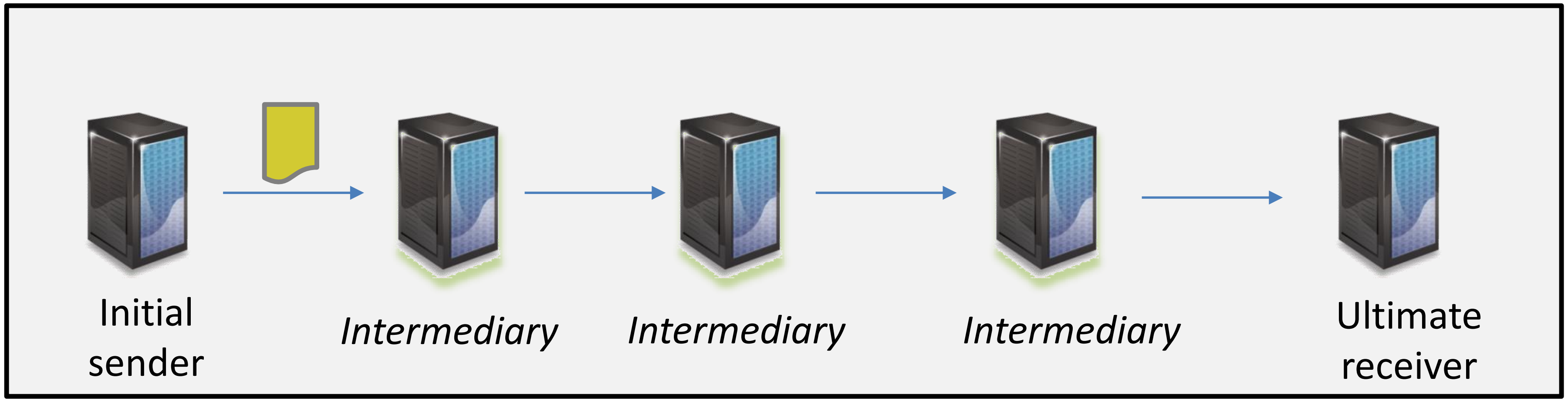
What is SOAP?

- SOAP = a messaging framework (much like HTTP)

- The SOAP specification covers the following four main areas:
  - **Message format:** A format for one-way communication describing how a message can be packed into an XML document.
  - **Processing model:** A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. Describes what parts of the messages should be read by whom and how to react in case of failure.
  - **Extensibility model:** How the basic message constructs can be extended with application-specific constructs
  - **Protocol binding framework:** A description of how SOAP messages can be transported using different protocols (e.g., HTTP, SMTP, …).
  - A set of conventions on how to turn an RPC call into a SOAP message and back. (Since version 1.1. SOAP is not restricted to the RPC programming model).

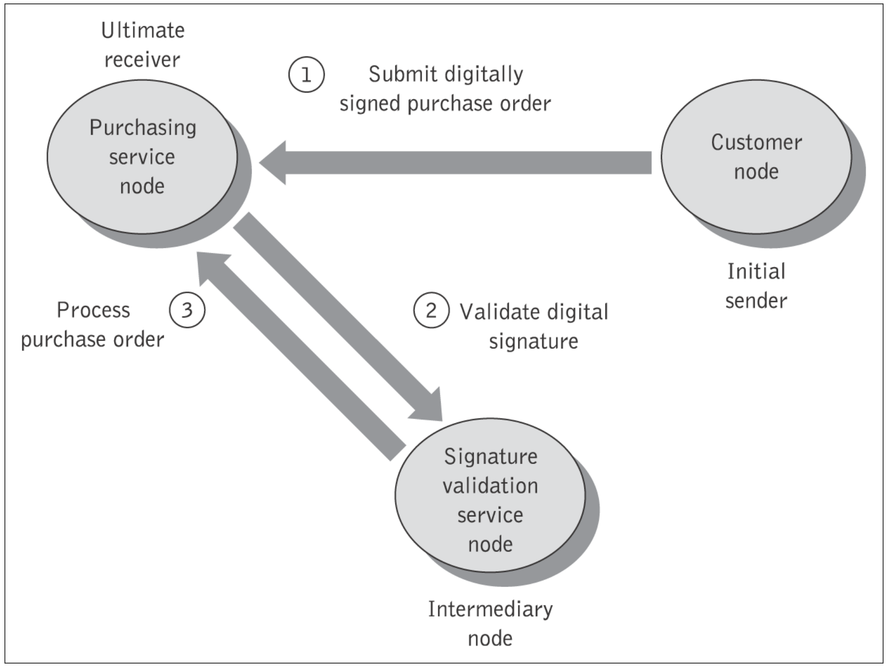
The SOAP message path

- A SOAP message can pass through multiple hops on the way from the initial sender to the ultimate receiver
- The entities involved in transporting the message are called **SOAP nodes**
- SOAP Intermediaries can manipulate and forward the message
- Every SOAP node assumes a certain role which influences how the message is processed at the node



SOAP Intermediaries

- SOAP headers have been designed in anticipation of participation of other SOAP processing nodes – called **SOAP *intermediaries*** – along a **message's path** from an initial SOAP sender to an ultimate SOAP receiver.
- A SOAP message travels along the message path from a sender to a receiver.
- All SOAP messages start with an initial sender, which creates the SOAP message, and end with an ultimate receiver.



Message Processing Model

- For each message received, every SOAP node on the message path must **process the message** as follows
  - Decide in which roles to act (standard roles: `next` or `ultimateReceiver`, or other application-defined roles).
    These roles may also depend on the contents of the message.
  - Identify the mandatory header blocks targeted at the node (matching role, mustUnderstand=true)
  - If a mandatory header block is not understood by the node, a fault must be generated. The message must not be processed further.
- Process the mandatory header blocks and, in case of the ultimate receiver, the body. Other header blocks targeted at the node may be processed. The order of processing is not significant.
- SOAP intermediaries will finally forward the message
  - Processed header blocks may be removed depending on the specification for the block.
  - Header blocks which were targeted at the intermediary but not processed are relayed only if the the relay attribute is set to true.
- Active SOAP intermediaries may also change a message in ways not described here (e.g., encrypt the message).

Example of a header with routing

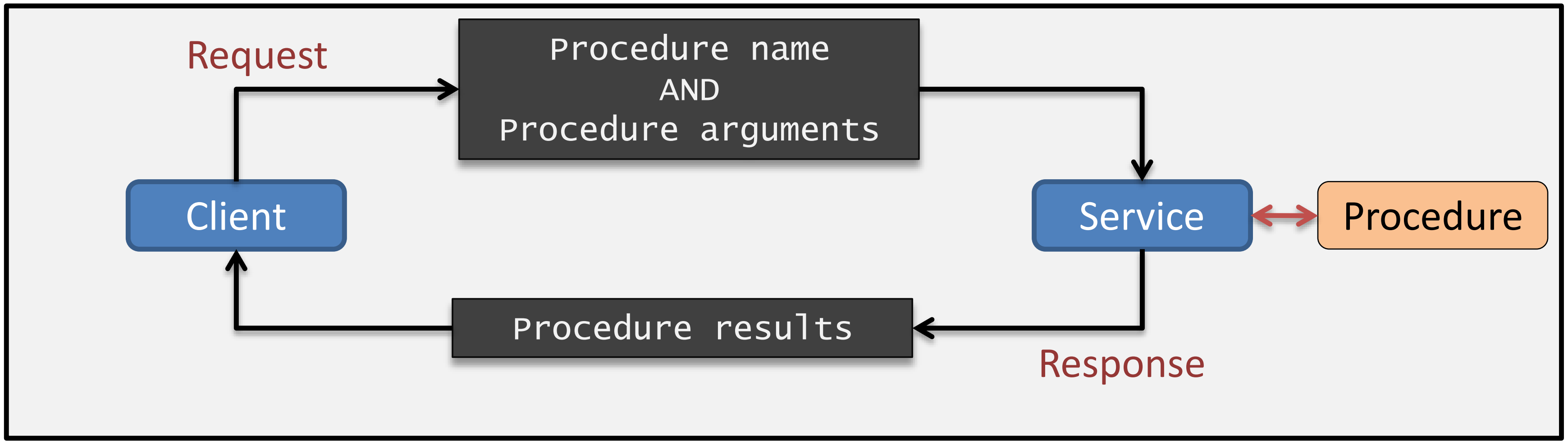
```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope  xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Header>
  <m:order xmlns:m="http://www.plastics_supply.com/purchase-order"
           env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
           env:mustUnderstand="true">
    <m:order-no >uuid:0411a2daa</m:order-no>
    <m:date>2004-11-8</m:date>
  </m:order>
  <n:customer xmlns:n="http://www.supply.com/customers"
             env:role="http://www.w3.org/2003/05/soap-envelope/role/next"
             env:mustUnderstand="true">
      <n:name> Marvin Sanders </n:name>
  </n:customer >
 </env:Header>
 <env:Body>
     <-- Payload element goes here -->
 </env:Body>
</env:Envelope>
```

What is SOAP?

- SOAP = a messaging framework (much like HTTP)

- The SOAP specification covers the following four main areas:
  - **Message format:** A format for one-way communication describing how a message can be packed into an XML document.
  - **Processing model:** A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. Describes what parts of the messages should be read by whom and how to react in case of failure.
  - **Extensibility model:** How the basic message constructs can be extended with application-specific constructs
  - **Protocol binding framework:** A description of how SOAP messages can be transported using different protocols (e.g., HTTP, SMTP, …).
  - A set of conventions on how to turn an **RPC call** into a SOAP message and back. (Since version 1.1. SOAP is not restricted to the RPC programming model).

RPC Style (1/2)



- Client sends message to a remote server and blocks while waiting for response
- Request message identifies the procedure to be executed and its arguments
- Server decodes message, maps message arguments directly to input parameters, executes procedure, and sends (serialized) results back to client

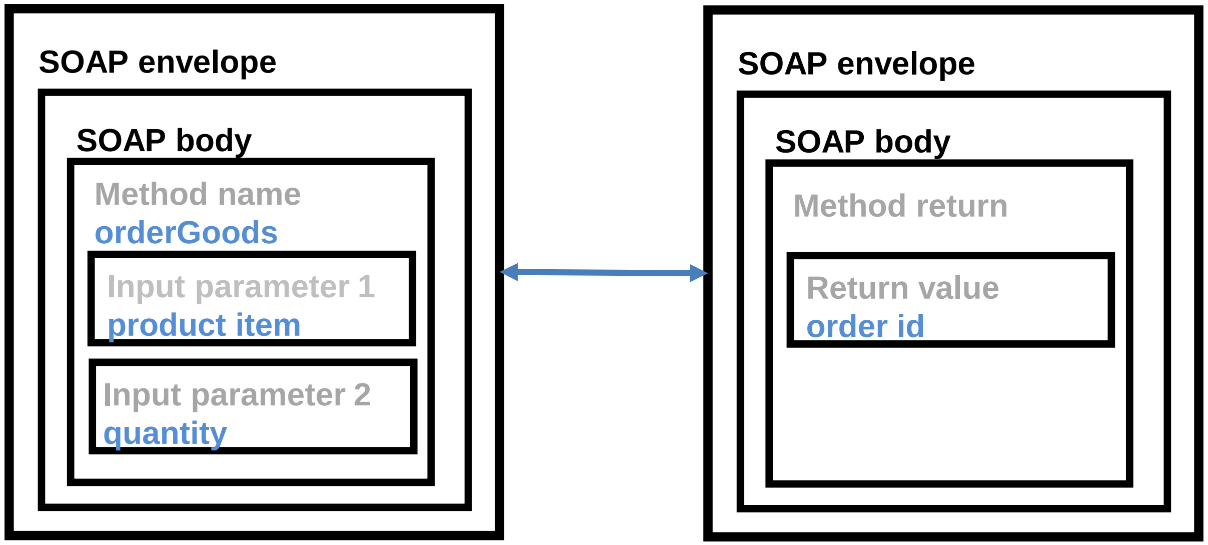
RPC Style (2/2)

- Pros:
  - Very easy to implement (lots of frameworks that automate the process, e.g. AX-WS framework for Java)

- Cons:
  - Usually inflexible and fragile: tight coupling between client and service, if procedure needs to change (e.g., number of arguments), all clients need to be rewritten.
  - Usually restricted to **synchronous communication** (client blocks while waiting for response)

RPC style SOAP services

- In a remote procedure call (RPC)-style Web service clients express their request as a method call with a set of arguments, which returns a response containing a return value.



RPC style SOAP services (cont.)

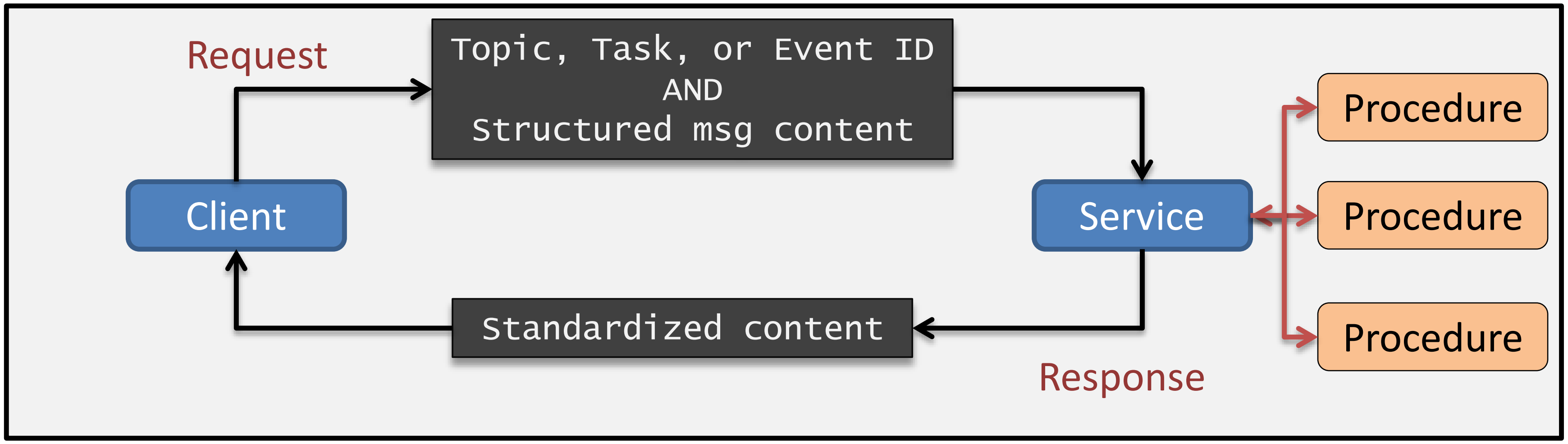
```
<env:Envelope
            xmlns:env=“http://www.w3.org/2003/05/soap-envelope”
            xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
           <tx:Transaction-id xmlns:t=”http://www.transaction.com/transactions”
                              env:mustUnderstand=“true”>
               512
           </tx:Transaction-id>
    </env:Header>
    <env:Body>
           <m:GetProductPrice>
              <product-id>  450R6OP  </product-id >
           </m:GetProductPrice >
    </env:Body>
</env:Envelope>
```

Example Request

```
<env:Envelope xmlns:env=“http://www.w3.org/2003/05/soap-envelope”
              xmlns:m="http://www.plastics_supply.com/product-prices">
    <env:Header>
           <!-- Optional context information -->
    </env:Header>
    <env:Body>
           <m:GetProductPriceResponse>
              <product-price>   134.32  </product-price>
           </m:GetProductPriceResponse>
    </env:Body>
</env:Envelope>
```

Example Response

Message-based style



- In a message-based API, messages are not derived from the signatures of remote procedures.
- Instead, messages may carry information on specific topics, tasks to execute, and events.
- The server selects the correct procedure to execute based on the message content

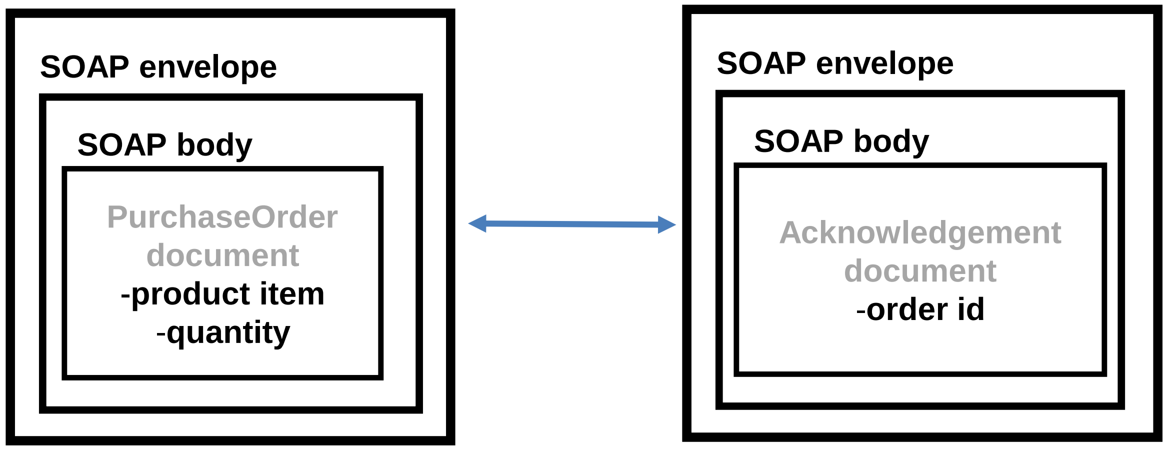
Message-based Style (2/2)

- Pros:
  - Looser coupling between clients and servers
  - Support for asynchronous communication [necessary on web-scale networks]

- Cons:
  - Messages must be standardized somehow. This is easy if communication is within the same organization, but more difficult when many parties are involved.

Message-style SOAP Services

- In the document-style of messaging, the SOAP <Body> contains an XML document fragment. The <Body> element reflects no explicit XML structure.
- The SOAP run-time environment accepts the SOAP <Body> element as it stands and hands it over to the application it is destined for unchanged. There may or may not be a response associated with this message.



Message-style SOAP Services (cont.)

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope”>

    <env:Header>
          <tx:Transaction-id xmlns:t=”http://www.transaction.com/transactions”
                             env:mustUnderstand="true”>
               512
          </tx:Transaction-id>
    </env:Header>
    <env:Body>
          <po:PurchaseOrder oderDate=”2004-12-02”
                         xmlns:po="http://www.plastics_supply.com/POs">
           <po:from>
             <po:accountName>   RightPlastics   </po:accountName>
             <po:accountNumber>   PSC-0343-02   </po:accountNumber>
           </po:from>
           <po:to>
             <po:supplierName>  Plastic Supplies Inc.  </po:supplierName>
             <po:supplierAddress>  Yara Valley Melbourne  </po:supplierAddress>
           </po:to>
           <po:product>
               <po:product-name> injection molder </po:product-name>
               <po:product-model> G-100T </po:product-model>
               <po:quantity> 2 </po:quantity>
           </po:product>
       </po:PurchaseOrder >
    </env:Body>
</env:Envelope>
```

SOAP Fault Element

- SOAP provides a model for handling faults arise.
- It distinguishes between the conditions that result in a fault, and the ability to signal that fault to the originator of the faulty message or another node. The SOAP <Body> is the place where fault information is placed.

```
<env:Envelope xmlns:env=“http://www.w3.org/2003/05/soap-envelope”
             xmlns:m="http://www.plastics_supply.com/product-prices">
  <env:Header>
   <tx:Transaction-id  xmlns:t=”http://www.transaction.com/transactions”
                      env:mustUnderstand='1'>
        512
   </tx:Transaction-id>
  </env:Header>
  <env:Body>
     <env:Fault>
        <env:Code>
           <env:Value>env:Sender</env:Value>
           <env:Subcode> <env:Value> m:InvalidPurchaseOrder </env:Value> </env:Subcode>
        </env:Code>
        <env:Reason>
           <env:Text xml:lang="en-UK"> Specified product did not exist </env:Text>
        </env:Reason>
        <env:Detail>  arbitrary XML here </env:Detail>
       </env:Fault>
    </env:Body>
</env:Envelope>
```

What is SOAP?

- SOAP = a messaging framework (much like HTTP)

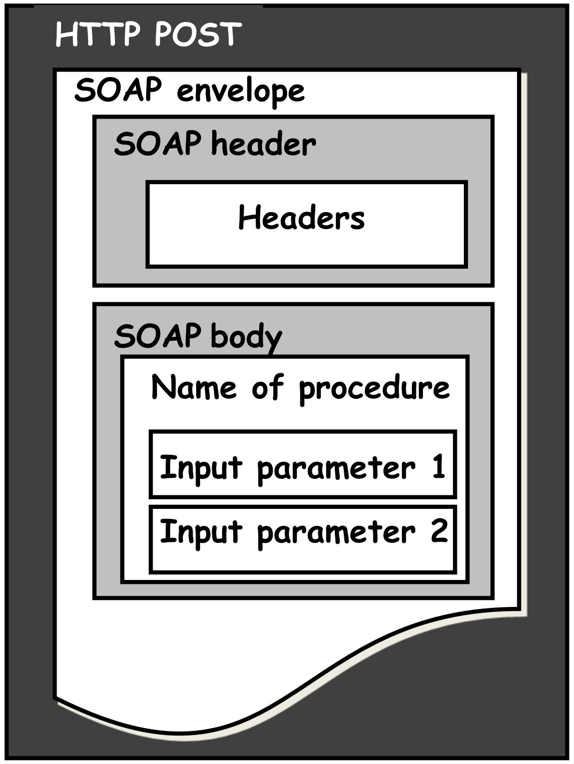
- The SOAP specification covers the following four main areas:
  - **Message format:** A format for one-way communication describing how a message can be packed into an XML document.
  - **Processing model:** A set of rules that must be followed when processing a SOAP message and a simple classification of the entities involved in processing a SOAP message. Describes what parts of the messages should be read by whom and how to react in case of failure.
  - **Extensibility model:** How the basic message constructs can be extended with application-specific constructs
  - **Protocol binding framework:** A description of how SOAP messages can be transported using different protocols (e.g., HTTP, SMTP, …).
  - A set of conventions on how to turn an RPC call into a SOAP message and back. (Since version 1.1. SOAP is not restricted to the RPC programming model).

SOAP Protocol Binding Framework

- SOAP messages can be transferred using any protocol,
- A binding of SOAP to a transport protocol is a description of how a SOAP message is to be sent using that transport protocol.
- A binding specifies how response and request messages are correlated.
- The SOAP binding framework expresses guidelines for specifying a binding to a particular protocol

SOAP HTTP Binding

- SOAP Messages are typically transferred using HTTP.
- The binding to HTTP defined in the SOAP specification
- SOAP can use GET or POST. With GET, the request is not a SOAP message but the response is a SOAP message, with POST both request and response are SOAP messages (in version 1.2, version 1.1 mainly considers the use of POST).
- SOAP uses the same error and status codes as those used in HTTP so that HTTP responses can be directly interpreted by a SOAP module.



SOAP over HTTP Example

```
POST /order/billing HTTP/1.1
Host: billing.eserver.com
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 12354

<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Body>
   <order xmlns="http://orders.com/">
            <clientID>1892AxF1</clientID>
            <itemID>456D1</itemID>
   </order>
</env:Body>
</env:Envelope>
```

SOAP over HTTP Example

SOAP over HTTP Example

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

<?xml version="1.0" encoding="utf-8"?>
<env:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
              xmlns:env="http://www.w3.org/2003/05/soap-envelope">
 <env:Body>
   <orderCreated xmlns="http://orders.com/"/>
 </env:Body>
</env:Envelope>
```

Part II

WSDL:
WEB SERVICE DESCRIPTION LANGUAGE

Service Description

- Any service that you provide, others will have to program and use.

- The purpose of a service = abstract from system & programming language heterogeneity
  - For instance, neither the service requestor nor the provider should be aware of each other's technical infrastructure, programming language, or distributed object framework (if any).

- **A Service description** is a machine understandable document that
  - describes the operations of a web service,
  - describes how the web service should be accessed (transport protocol, access point),
  - describes what the content of the messages that are exchanged is

It hence reduces the amount of required common understanding and custom programming and integration.

WSDL

- WSDL = Web Service Description Language

- A WSDL document is an XML document that specifies the details of the complete interfaces exposed by web services:
  - ***what*** a service does, i.e., the operations the service provides,
  - ***where*** it resides, i.e., details of the protocol-specific address (e.g., its URI)
  - ***how*** to invoke it, i.e., details of the data formats (e.g., XML) and protocols (e.g., SOAP over HTTP) necessary to access the service’s operations.

WSDL History

- Evolution of the standard:
  - WSDL 1.0 (first release) – 2001
  - WSDL 1.1. – 2003
  - WSDL 2.0 (current version) – 2007

- Originally, WSDL was primarily meant to describe SOAP web services; with WSDL 2.0 it is also possible to describe REST web services.

WSDL History

- Evolution of the standard:
  - WSDL 1.0 (first release) – 2001
  - WSDL 1.1. – 2003
  - WSDL 2.0 (current version) – 2007

- Originally, WSDL was primarily meant to describe SOAP web services; with WSDL 2.0 it is also possible to describe REST web services.

CAUTION:
What follows is an overview of WSDL 2.0, which is a significant (non-compatible) change with earlier WSDL versions!

WSDL document structure

- The compontents of a WSDL file parallel those of an interface in a programming language

```
public interface ReservationInterface {
   Result checkAvailability(BookingDesiderata bd)
                    throws InvalidDataException;

   Result book(BookingDesiderata bd)
                    throws InvalidDataException,
                          NotAvailableException;
}
```

WSDL document structure

- The compontents of a WSDL file parallel those of an interface in a programming language

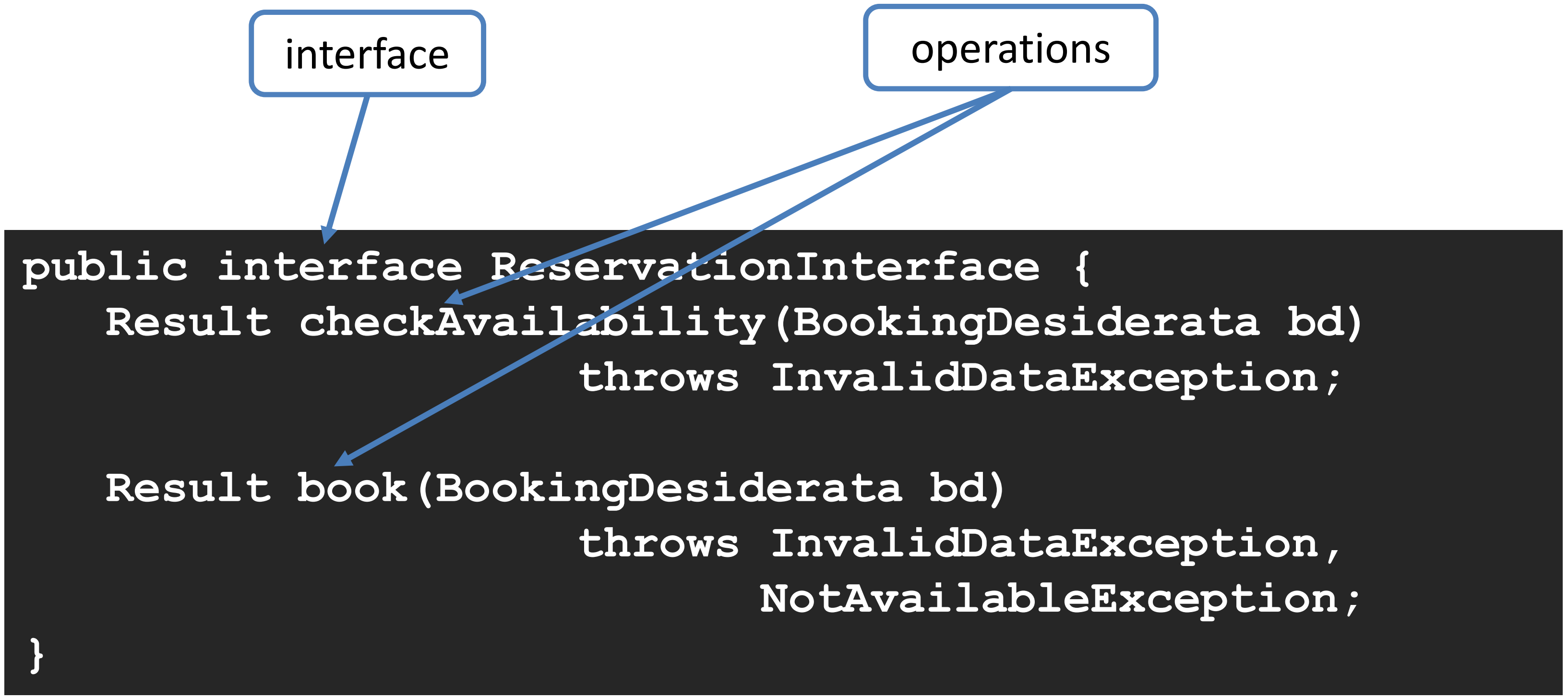
interface

```
public interface ReservationInterface {
   Result checkAvailability(BookingDesiderata bd)
                        throws InvalidDataException;

   Result book(BookingDesiderata bd)
                        throws InvalidDataException,
                               NotAvailableException;
}
```

WSDL document structure

- The compontents of a WSDL file parallel those of an interface in a programming language

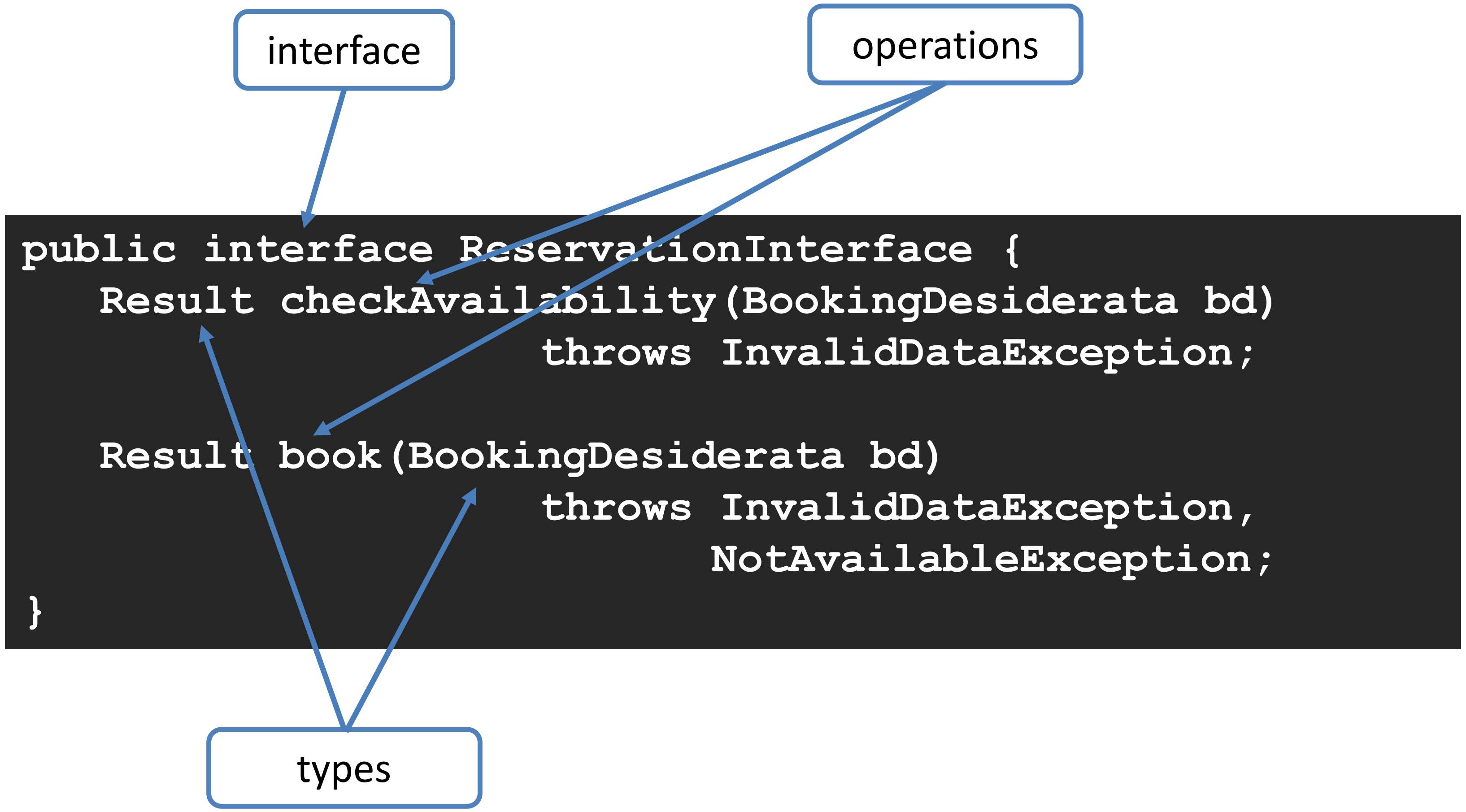


```
public interface ReservationInterface {
   Result checkAvailability(BookingDesiderata bd)
                     throws InvalidDataException;

   Result book(BookingDesiderata bd)
                     throws InvalidDataException,
                            NotAvailableException;
}
```

WSDL document structure

- The compontents of a WSDL file parallel those of an interface in a programming language

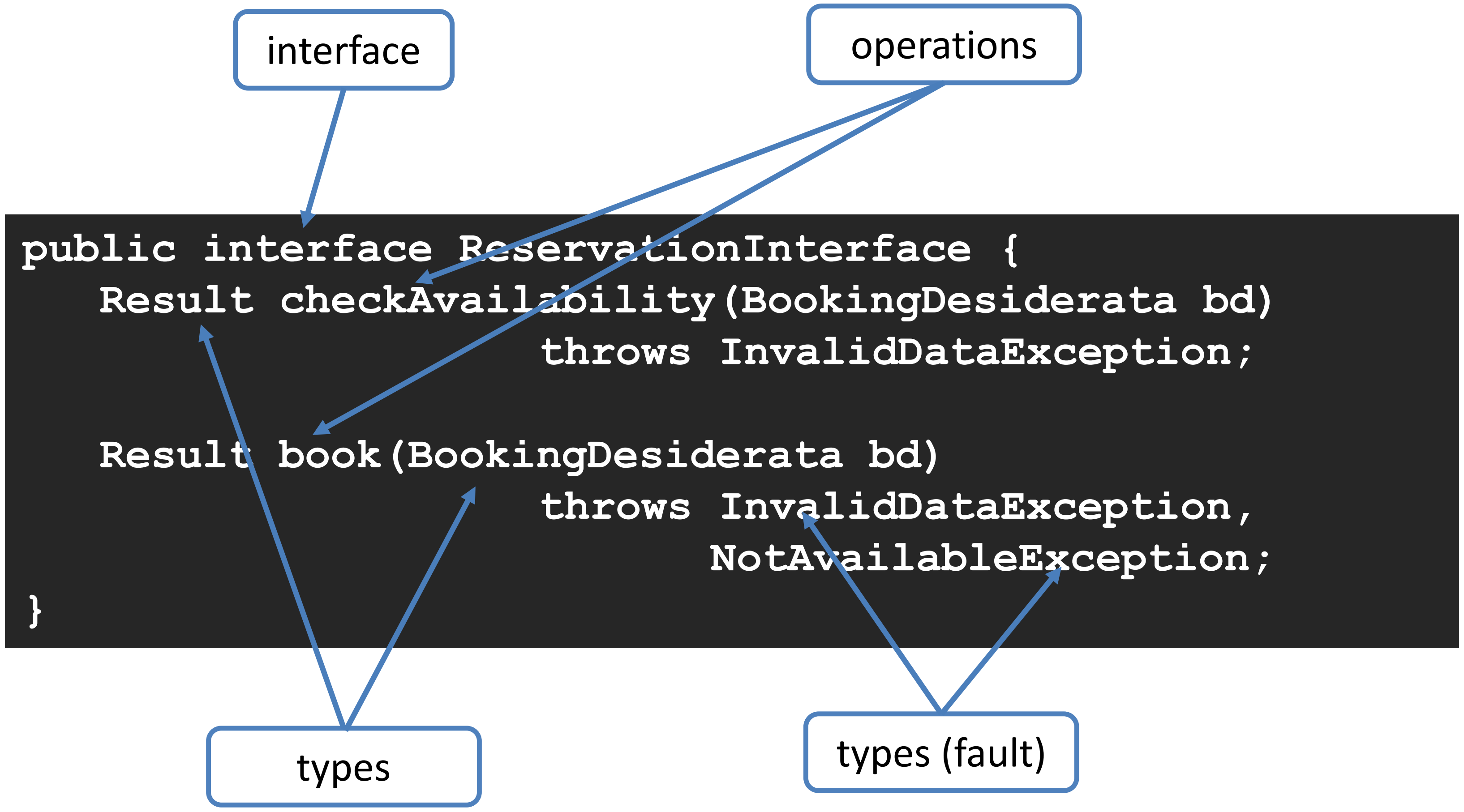


```
public interface ReservationInterface {
   Result checkAvailability(BookingDesiderata bd)
                    throws InvalidDataException;

   Result book(BookingDesiderata bd)
                    throws InvalidDataException,
                           NotAvailableException;
}
```

WSDL document structure

- The compontents of a WSDL file parallel those of an interface in a programming language



```
public interface ReservationInterface {
   Result checkAvailability(BookingDesiderata bd)
                    throws InvalidDataException;

   Result book(BookingDesiderata bd)
                    throws InvalidDataException,
                           NotAvailableException;
}
```

WSDL document structure

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl">
   <wsdl:types> … </wsdl:types>
   <wsdl:interface> … <wsdl:interface>
   <wsdl:binding> … </wsdl:binding>
   <wsdl:service> … </wsdl:service>
</wsdl:description>
```

- The `<wsdl:types>` element defines the « types » of the messages that are to be exchanged.
- By default, WSDL assumes that the messages will be in XML, and it uses XML Schema to describe the exact structure of the XML that needs to be exchanged.

WSDL document structure

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl">
   <wsdl:types> … </wsdl:types>
   <wsdl:interface> … <wsdl:interface>
   <wsdl:binding> … </wsdl:binding>
   <wsdl:service> … </wsdl:service>
</wsdl:description>
```

- The `<wsdl:types>` element defines the « types » of the messages that are to be exchanged.
- By default, WSDL assumes that the messages will be in XML, and it uses XML Schema to describe the exact structure of the XML that needs to be exchanged.

Note
In principle, WSDL allows other schema languages or type systems (one could imagine, an other language to describe the structure of JSON content to be exchanged), but this seems to be not widely implemented.

Defining message types

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    … >
  . . .
  <types>
    <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        targetNamespace="http://greath.example.com/2004/schemas/resSvc">

      <xs:element name="bookingDesiderata" type="tBookingDesiderata"/>
      <xs:complexType name="tBookingDesiderata">
        <xs:sequence>
          <xs:element  name="checkInDate" type="xs:date"/>
          <xs:element  name="checkOutDate" type="xs:date"/>
          <xs:element  name="roomType" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>

      <xs:element name="checkAvailabilityResult" type="xs:double"/>
      <xs:element name="invalidDataError" type="xs:string"/>
    </xs:schema>
  </types>
  .  .  .
</description>
```

WSDL document structure

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl">
   <wsdl:types> … </wsdl:types>
   <wsdl:interface> … <wsdl:interface>
   <wsdl:binding> … </wsdl:binding>
   <wsdl:service> … </wsdl:service>
</wsdl:description>
```

- The `<wsdl:interface>` element defines the « interfaces » provided by the web service (as collections of operations).
- For each operation, it specifies the kind of message exchange pattern (MEP). Popular MEPS are: request + response, request-only, response-only, …

Defining interfaces

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">

 <types> … </types>

 <interface  name = "reservationInterface" >
   <fault name = "invalidDataFault" element = "ghns:invalidDataError"/>

   <operation name="opCheckAvailability"
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
     <input messageLabel="In" element="ghns:bookingDesiderata " />
     <output messageLabel="Out" element="ghns:checkAvailabilityResult" />
     <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
 </interface>
 …
</description>
```

Defining interfaces

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">

 <types> … </types>

 <interface  name = "reservationInterface" >
   <fault name = "invalidDataFault" element = "ghns:invalidDataError"/>

   <operation name="opCheckAvailability"
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
     <input messageLabel="In" element="ghns:bookingDesiderata " />
     <output messageLabel="Out" element="ghns:checkAvailabilityResult" />
     <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
 </interface>
 …
</description>
```

Request+response MEP

Defining interfaces

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">

 <types> … </types>

 <interface  name = "reservationInterface" >
   <fault name = "invalidDataFault" element = "ghns:invalidDataError"/>

   <operation name="opCheckAvailability"
           pattern="http://www.w3.org/ns/wsdl/in-out"
           style="http://www.w3.org/ns/wsdl/style/iri"
           wsdlx:safe = "true">
     <input messageLabel="In" element="ghns:bookingDesiderata " />
     <output messageLabel="Out" element="ghns:checkAvailabilityResult" />
     <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
 </interface>
 …
</description>
```

Call is "safe": does not induce client obligation (like buying something)

Defining interfaces

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">

 <types> … </types>

 <interface  name = "reservationInterface" >
   <fault name = "invalidDataFault"

   <operation name="opCheckAvailabi
            pattern="http://www.w3.org/ns/wsdl/in-out"
            style="http://www.w3.org/ns/wsdl/style/iri"
            wsdlx:safe = "true">
     <input messageLabel="In" element="ghns:bookingDesiderata " />
     <output messageLabel="Out" element="ghns:checkAvailabilityResult" />
     <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
 </interface>
 …
</description>
```

Message type to send for operation call (element defined in types)

Defining interfaces

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">

 <types> … </types>

 <interface  name = "reservationInterface" >
   <fault name = "invalidDataFault" element = "ghns:invalidDataError"/>

   <operation name="opCheckAvailability"
           pattern="http://www.w3.org/ns/wsdl/in-out"
           style="http://www.w3.org/ns/wsdl/style/iri"
           wsdlx:safe = "true">
     <input messageLabel="In" element="ghns:bookingDesiderata " />
     <output messageLabel="Out" element="ghns:checkAvailabilityResult" />
     <outfault ref="tns:invalidDataFault" messageLabel="Out"/>
    </operation>
 <
 …
</description>
```

Message type to send for operation call

WSDL document structure

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl">
   <wsdl:types> … </wsdl:types>
   <wsdl:interface> … <wsdl:interface>
   <wsdl:binding> … </wsdl:binding>
   <wsdl:service> … </wsdl:service>
</wsdl:description>
```

- The `<wsdl:binding>` element defines **how** the messages should be transmitted, i.e., what transport protocol should be used to access a given interface. (E.g. SOAP, plain HTTP, …)

- It is possible to give multiple bindings for a single interface. (E.g. support both SOAP and some other binding.)

Defining bindings

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
  <types> … </types>

  <interface name="reservationInterface"> …  </interface>

  <binding name="reservationSOAPBinding"
          interface="tns:reservationInterface"
          type="http://www.w3.org/ns/wsdl/soap"
          wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>
  </binding>
…
</description>
```

Defining bindings

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.
    xmlns:soap="http://www.w3
…
  <types> … </types>

  <interface name="reservationInterface"> …  </interface>

  <binding name="reservationSOAPBinding"
          interface="tns:reservationInterface"
          type="http://www.w3.org/ns/wsdl/soap"
          wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>
  </binding>
…
</description>
```

Expose the tns:reservationInterface defined earlier over SOAP, using the SOAP-over-HTTP transport protocol

Defining bindings

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
  <types> … </types>

  <interface name="reservationInterface"> …  </interface>

  <binding name="reservationSOAPBinding"
        interface="tns:reservationInterface"
        type="http://www.w3.org/ns/wsdl/soap"
        wsoap:protocol="http://www.w3.org/2003/05/soap/bindings/HTTP/">

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

    <fault ref="tns:invalidDataFa
      wsoap:code="soap:Sender"/>
  </binding>
…
</description>
```

Specify that the opCheckAvailability in this interface should be called using HTTP GET

Defining bindings

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
  <types> … </types>

  <interface name="reservationInterface"> …  </interface>

  <binding name="reservationSOAPBinding"
          interface="tns:reservationInterface"
                                              /soap"
                                              rg/2003/05/soap/bindings/HTTP/">

    <operation ref="tns:opCheckAvailability"
      wsoap:mep="http://www.w3.org/2003/05/soap/mep/soap-response"/>

    <fault ref="tns:invalidDataFault"
      wsoap:code="soap:Sender"/>
  </binding>
…
</description>
```

If a fault element is sent, specify that the SOAP error code should be set to “sender”

WSDL document structure

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl">
   <wsdl:types> … </wsdl:types>
   <wsdl:interface> … <wsdl:interface>
   <wsdl:binding> … </wsdl:binding>
   <wsdl:service> … </wsdl:service>
</wsdl:description>
```

- The `<wsdl:service>` element defines **where** the service can be accessed (its endpoint).

- Again, the same binding can be made accessible at multiple places (through multiple endpoint elements)

Defining services

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
  <types> … </types>

  <interface name="reservationInterface" > … </interface>

  <binding name="reservationSOAPBinding"
           interface="tns:reservationInterface" …>
     …
  </binding>

  <service name="reservationService"
           interface="tns:reservationInterface">

     <endpoint name="reservationEndpoint"
               binding="tns:reservationSOAPBinding"
               address ="http://greath.example.com/2004/reservation"/>
  </service>
  </description>
```

Defining services

```
<?xml version="1.0" encoding="utf-8" ?>
<description
    xmlns="http://www.w3.org/ns/wsdl"
    targetNamespace= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:tns= "http://greath.example.com/2004/wsdl/resSvc"
    xmlns:ghns = "http://greath.example.com/2004/schemas/resSvc"
    xmlns:wsoap= "http://www.w3.org/ns/wsdl/soap"
    xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
…
  <types> … </types>

  <interface name="reservationInterface" > … </interface>

  <binding name="reservationSOAPBinding"
           interface="tns:reser
     …
  </binding>

  <service name="reservationService"
           interface="tns:reservationInterface">

     <endpoint name="reservationEndpoint"
               binding="tns:reservationSOAPBinding"
               address ="http://greath.example.com/2004/reservation"/>
  </service>
  </description>
```

Expose the tns:reservationSOAPBinding binding at this URI

RESTful WSDL

- With WSDL 2.0 it is also possible to describe RESTful web services.

```
<?xml version="1.0" encoding="utf-8" ?>
<description xmlns="http://www.w3.org/ns/wsdl"
             xmlns:whttp="http://www.w3.org/ns/wsdl/http" >
…
  <binding name="reservationHTTPBinding"
      interface="tns:reservationInterface"
      type="http://www.w3.org/ns/wsdl/http"
      whttp:methodDefault="GET">

    <operation ref="tns:opCheckAvailability"
               whttp:location="{bookingDesiderata}"  />
  </binding>

  <service name="reservationService"
           interface="tns:reservationInterface">

    <!-- HTTP 1.1 GET End Point -->
    <endpoint name="reservationEndpoint"
        binding="tns:reservationHTTPBinding"
        address="http://greath.example.com/2004/checkAvailability/"/>
  </service>
…
</description>
```

RESTful WSDL

- With WSDL 2.0 it is also possible to describe RESTful web services.

Support the operations of reservationInterface by the HTTP transport protocol, using GET as the method

```
<?xml version="1.0" encoding="utf-8
<description xmlns="http://www.w3.c
             xmlns:whttp="http://ww
…
  <binding name="reservationHTTPBinding"
      interface="tns:reservationInterface"
      type="http://www.w3.org/ns/wsdl/http"
      whttp:methodDefault="GET">

    <operation ref="tns:opCheckAvailability"
             whttp:location="{bookingDesiderata}"  />
  </binding>

  <service name="reservationService"
          interface="tns:reservationInterface">

    <!-- HTTP 1.1 GET End Point -->
    <endpoint name="reservationEndpoint"
        binding="tns:reservationHTTPBinding"
        address="http://greath.example.com/2004/checkAvailability/"/>
  </service>
…
</description>
```

RESTful WSDL

- With WSDL 2.0 it is also possible to describe RESTful web services.

```
<?xml version="1.0" encoding="utf-8
<description xmlns="http://www.w3.c
             xmlns:whttp="http://ww
…
  <binding name="reservationHTTPBinding"
      interface="tns:reservationInterface"
      type="http://www.w3.org/ns/wsdl/http"
      whttp:methodDefault="GET">

    <operation ref="tns:opCheckAvailability"
             whttp:location="{bookingDesiderata}"  />
  </binding>

  <service name="reservationServ
           interface="tns:reserva

    <!-- HTTP 1.1 GET End Point -
    <endpoint name="reservationEndpoint"
        binding="tns:reservationHTTPBinding"
        address="http://greath.example.com/2004/checkAvailability/"/>
  </service>
…
</description>
```

Support the operations of reservationInterface by the HTTP transport protocol, using GET as the method

The check availability operation can be called by appending the bookingDesiderata in application/x-www-form-urlencoded form to the endpoint URL

RESTful WSDL

- With WSDL 2.0 it is also possible to describe RESTful web services.

```
<?xml version="1.0" encoding="utf-8
<description xmlns="http://www.w3.c
             xmlns:whttp="http://ww
…
  <binding name="reservationHTTPBinding"
        interface="tns:reservationInterface"
        type="http://www.w3.org/ns/wsdl/http"
        whttp:methodDefault="GET">

    <operation ref="tns:opCheckAvailability"
               whttp:location="{bookingDesiderata}"  />
  </binding>

  <service name="reservationServ
           interface="tns:reserva

    <!-- HTTP 1.1 GET End Point -
    <endpoint name="reservationEndpoint"
          binding="tns:reservationHTTPBinding"
          address="http://greath.example.com/2004/checkAvailability/"/>
  </service>
…
</
```

Support the operations of reservationInterface by the HTTP transport protocol, using GET as the method

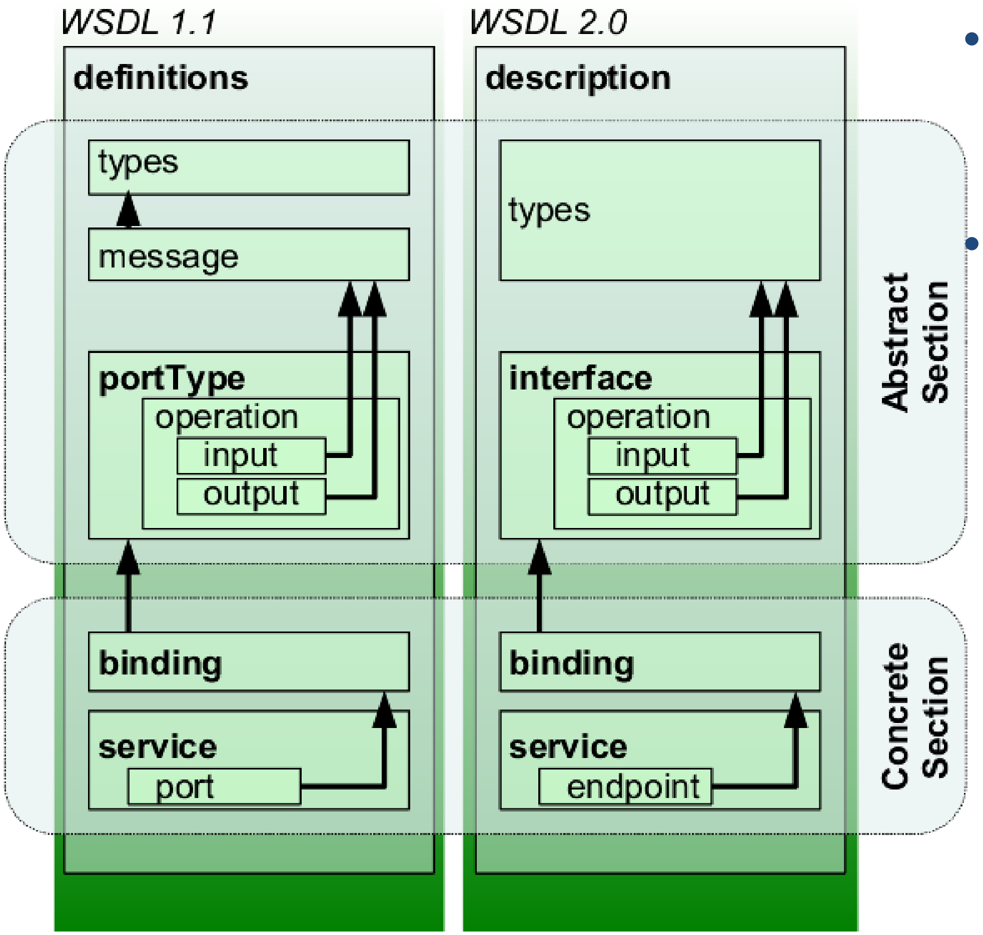
The check availability operation can be called by appending the bookingDesiderata in application/x-www-form-urlencoded form to the endpoint URL

This gives, e.g.,
`http://greath.example.com/2004/checkAvailability/5-5-5?checkOutDate=6-6-5&roomType=foo.`

Phew …

- That seems complicated and a lot of work. Why bother?
- Automated tooling support! (E.g. Java -> WSDL, WSDL -> Java)

WSDL 1.1 versus 2.0



- WSDL 1.1 is still used a lot in practice (and most Java tools do not support 2.0)
- WSDL 1.1 has an extra `message` element. And it uses `portType` instead of `interface` and `port` instead of `endpoint`.

References

- M. P. Papazoglou, Web Services: Principles and Technology 2nd edition, Prentice Hall

- World Wide Web Consortium, SOAP Version 1.2 Part 0: Primer (Second Edition) http://www.w3.org/TR/soap12-part0/

- World Wide Web Consortium, Web Services Description Language (WSDL) Version 2.0 Part 0: Primer http://www.w3.org/TR/wsdl20-primer