A WSDL document uses the following elements in the definition of network services:

- **Types**: a container for non-built-in data type definitions using some type system, e.g., arrays and structures.
- **Message**: an abstract, typed definition of the data being transferred between the requestor and service; method call (request/response): modeled as 2 messages.
- **Port Type**: an abstract set of operations supported by one or more endpoints; an operation specifies a specific input/output message sequence.
- **Operation**: an abstract description of an action supported by the service.
- **Binding**: specifies a concrete protocol and data format for the operations and messages defined by a particular PortType, such as SOAP or Corba.
- **Port**: a single endpoint defined as a combination of a binding and a network address.
- **Service**: a collection of related endpoints.

**Parts of WSDL**

**Relationship of parts**

WSDL is divided into 2 parts:

- **abstract part** which describes what is offered; it consists of types, message, operations and port types.
- **concrete part** which describes how and where it is offered; it consists of bindings, services and ports.

```xml
<definitions>
  <types>
    ...
  </types>
  <message name="">
    ...
  </message>
  <portType name="">
    ...
  </portType>
  <binding name="">
    ...
  </binding>
  <service name="">
    ...
  </service>
</definitions>
```
WSDL definition of a simple service providing stock quotes; the service supports the single operation GetLastTradePrice(ticker symbol) and returns the price as a float.

```xml
<?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/wsd1/soap/"
    xmlns="http://schemas.xmlsoap.org/wsd1/">
    <types>
        <schema targetNamespace="http://example.com/stockquote.xsd"
            xmlns="http://www.w3.org/2001/XMLSchema">
            <element name="TradePriceRequest">
                <complexType>
                    <all>
                        <element name="tickerSymbol" type="string"/>
                    </all>
                </complexType>
            </element>
            <element name="TradePrice">
                <complexType>
                    <all>
                        <element name="price" type="float"/>
                    </all>
                </complexType>
            </element>
        </schema>
    </types>
</definitions>
```

Example for SOAP Request/Response

```xml
<complexType>
    <sequence>
        <element name="tickerSymbol" type="string"/>
    </sequence>
</complexType>
</schema>
</types>

<-- Parameter der Nachricht -->
<message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest"/>
</message>

<-- Parameter der Antwort -->
<message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice"/>
</message>

<portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
        <input message="tns:GetLastTradePriceInput"/>
        <output message="tns:GetLastTradePriceOutput"/>
    </operation>
</portType>
```
Example for SOAP Request/Response

```xml
<message>
  <!-- Parameter der Antwort -->
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd:TradePrice"/>
  </message>
</message>

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

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

Generating code from WSDL

Use of a WSDL compiler to automatically generate code (e.g., a Java interface) from a WSDL file.

WSL documents can be generated from APIs (1).
Stubs and skeletons can be generated from WSDL document (2).

Web Services Description Language (WSDL)

Ian Forster states: “Web service have little value if others cannot discover, access, and make sense of them.”

**Definition:** A **WSDL** document defines **services** as collections of network endpoints, or ports.

A **WSDL** has a purpose similar to that of IDLs in conventional middleware platforms. A **WSDL** description describes 3 fundamental properties of a Web Service:

- **What** a service does: operations and the arguments needed to invoke them.
- **How** a service is accessed: details of data formats and protocols.
- **Where** a service is located: details of the protocol-specific network address, such as a URI.

**WSDL Information Model**

- **Example for SOAP Request/Response**
- **Generating code from WSDL**
- **Common bad Practices**

**Generated by Typemunk**

Analysis of existing WSDL documents shows that functionality of many Web services are hard to understand due to bad practices:

- developers take insufficient care of names and comments.
- port types are tied to concrete protocols.
- semantically unrelated operations are placed in a single port type.
- overload output messages to transport results and error information.

**Generated by Typemunk**
Universal Description, Discovery, and Integration (UDDI)

provides the definition of a set of services supporting the description and discovery of businesses, organizations, and Web Service providers, the Web services they make available, the technical interface to access those services.

UDDI itself is a Web Service; has a WSDL interface and can be described by a UDDI registry.

**UDDI Business Registry System**

**UDDI Entities**

**UDDI Registry API**

UDDI registry [methods] for publicly available Web Services.

UDDI allows to store and manipulate four main types of entities

- **businessEntity**: represents the owner of a Web Service.
  - Attributes: name, unique key, zero or more services, descriptions, ...

- **businessService**: represents a group of one or more Web Services.
  - Attributes: name, unique key, one binding template per Web Service, descriptions, ...

- **bindingTemplate**: represents a single Web Service; contains all the information to locate and invoke the service.
  - Attributes: unique key, an access point that indicates the URL of the Web Service

- **TModel**: represents WSDL interface type.
  - Attributes: name, unique key, an URL that points to the data associated with the TModel, description,
Universal Description, Discovery, and Integration (UDDI)

provides the definition of a set of services supporting the description and discovery of businesses, organizations, and Web Service providers, the Web services they make available, the technical interface to access those services.

UDDI itself is a Web Service; has a WSDL interface and can be described by a UDDI registry.

UDDI Business Registry System
UDDI Entities
UDDI Registry API

UDDI registry [methods] for publicly available Web Services.
Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardized by the W3C community.

Motivation - Example

Service Oriented Architecture - SOA
Web Services - Characteristics
Web Services Architecture
Simple Object Access Protocol (SOAP)
Web Services Description Language (WSDL)
Universal Description, Discovery, and Integration (UDDI)
REST
Web Service Composition
Adopting Web Services
Mashups

REST (Representational State Transfer) is an architectural style of distributed applications.

REST is not a standard; it is a set of principles how to use Web standards, such as HTTP, URLs and Mime Types.

The Web is a REST system.

REST is based on the following key principles:
give every relevant resource an ID; use URLs to identify everything that is an item of interest.

URL: http://www.boeing.com/aircraft/747

A representation of the resource is returned (e.g., Boeing747.html). The representation places the client application in a state.

Ix resources together: navigating links results in state transfers of the client application.

use standard methods: such as get, post, put, delete

communication is stateless.

resources with multiple representations: client may specify the formats which it accepts

GET /customers/1234 HTTP/1.1

Accept: text/x-vcard
Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the WSC community.

**Motivation - Example**

Service Oriented Architecture - SOA

Web Services - Characteristics

Web Services Architecture

Simple Object Access Protocol (SOAP)

Web Services Description Language (WSDL)

Universal Description, Discovery, and Integration (UDDI)

REST

Web Service Composition

Adopting Web Services

Mashups
Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardized by the W3C community.

**Motivation - Example**

**Service Oriented Architecture - SOA**

**Web Services - Characteristics**

**Web Services Architecture**

**Simple Object Access Protocol (SOAP)**

**Web Services Description Language (WSDL)**

**Universal Description, Discovery, and Integration (UDDI)**

**REST**

**Web Service Composition**

**Adopting Web Services**

**Mashups**

---

**Dimension to handle complexity**

**Web Service Orchestration**

Component model: defines the sub-services.

Orchestration model: defines the order in which the sub-services are invoked.

WS-Coordination is an extensible framework that describes how different Web Services work together reliably. Coordination framework contains:

- Activation, Registration and Coordination services
- Data access model: specifies the data exchange between the sub-services.
- Transactional model: transactional semantics of the composed service.

WS-Transaction specifies the protocol for each coordination type (used by WS-Coordination):

- Atomic Transactions: all-or-nothing property, 2-phase commit.

Business Activity: handle long-lived activities and to apply business logic to handle business exceptions; Business Agreement Protocol.

Exception handling: handling of errors in the sub-services.
Web services provide a standard means of communication among distributed software applications based on the Web technology. Standardization by the WSC community.

**Motivation - Example**

**Service Oriented Architecture - SOA**

**Web Services - Characteristics**

**Web Services Architecture**

**Simple Object Access Protocol (SOAP)**

**Web Services Description Language (WSDL)**

**Universal Description, Discovery, and Integration (UDDI)**

**REST**

**Web Service Composition**

**Adopting Web Services**

**Mashups**

---

**Example Web Services**

The following lists some available Web services; often registration necessary in order to use them.

**Amazon E-Commerce Service (ECS)**

**FedEx Office and Printing Service**

- Printing of online documents and distribution of paper documents as commercial Web Service.
- Free print plug-in for standard office application; plug-in added to list of printers.
- Pick up your document at any U.S. location or ship via FedEx for added convenience.
- Location independent use of printing service.

**ViaMichelin**

- Reverse Geocoding Web Service allows users to obtain the closest road segment (named or not) for each supplied geographic coordinates (WGS84).

**XMethode**

- Clearinghouse for Web Services

---

**Amazon E-Commerce Service (ECS)**

**FedEx Office and Printing Service**

- Printing of online documents and distribution of paper documents as commercial Web Service.
- Free print plug-in for standard office application; plug-in added to list of printers.
- Pick up your document at any U.S. location or ship via FedEx for added convenience.
- Location independent use of printing service.

**ViaMichelin**

- Reverse Geocoding Web Service allows users to obtain the closest road segment (named or not) for each supplied geographic coordinates (WGS84).

**XMethode**

- Clearinghouse for Web Services
Apache Axis supports an environment to implement and provide Web services.

A set of client-side APIs for dynamically invoking SOAP Web services (with or without WSDL descriptions), tools to translate WSDL documents into Java frameworks, mechanisms for hosting Web services either within a servlet container (e.g., Tomcat) or via standalone server, a set of APIs for manipulating SOAP envelopes, bodies, and headers, and using them inside Message objects, data binding which enables mapping of Java classes into XML schemas and vice versa, a transport framework that allows usage of a variety of underlying transport mechanisms (e.g., JMS, email, etc).

Axis2

In the meantime there exists already Apache Axis2. Axis2 is more flexible, efficient and configurable in comparison to Axis1.x

Axis2 not only supports SOAP messages, but it also supports RESTful Web services.

Java provides a number of APIs implementing the Web Services standards

SAAJ (SOAP with Attachments API for Java)
SOAP messages as Java objects
JAX-WS (Java API for XML based Web-Services)
programming model for Web Services, replaces JAX-RPC
JAXRPC: Accessing WSDL descriptions
JAXR (Java API for XML Registries)
Accessing Web Services Registries, e.g., UDDI
JAXP (Java API for XML Processing)
Abstract XML API-Specification implemented by e.g., Apache Xalan(XSLT), Apache Xerces2 (XML Parsing (DOM, SAX...)), XWSS (Java Web-Services Security)
Signature, Encryption (roughly for SOAP what SSL is for HTTP)

Integration and WS Standards

<table>
<thead>
<tr>
<th>process management</th>
<th>meta data &amp; additional services</th>
</tr>
</thead>
<tbody>
<tr>
<td>process modeling, execution: BPEL4WS</td>
<td>meta database: UDDI</td>
</tr>
<tr>
<td>process control:</td>
<td>system management:</td>
</tr>
<tr>
<td>user interface integration: WSUWWSDL</td>
<td>security services:</td>
</tr>
<tr>
<td>message management:</td>
<td>WS-Security, SAML,</td>
</tr>
<tr>
<td>transformation services: XSLT</td>
<td>XML-Encryption, XML-Signature</td>
</tr>
<tr>
<td>synchronization:</td>
<td>development support:</td>
</tr>
<tr>
<td>transaction services: WS-Correlation, WS Transactions</td>
<td></td>
</tr>
</tbody>
</table>

Supporting - Restraining Forces

The adoption of Web Services in organization depends on

Supporting Forces
interoperable networked applications, i.e. independence of hardware, operating system, application server, ...
easier exchange of distributed data
easier access of enterprise wide data
availability of external services, encapsulation of legacy applications
cross-organizational computing
reduced maintenance cost, easier reuse of components
emerging industry-wide standard

Restraint Forces
different formats and semantics of data sources
security issues due to network access
standards are evolving and not fixed yet
lack of understanding of effects on operational systems
In order to allow for **automatic discovery of appropriate** web services and of **automatic interaction / chaining / incorporation** with web services, we need semantic meta-data for web services: Web-Service Ontologies, DataTypes with rich semantics, etc.

**Example:** Map-Service

**Input:** (int, int)

**Output:** APPLICATION/GIF

Input: (int, int):

- (x,y) of center of map?
- of corner of map? which corner?
- what coordinate system? Wgs84? Gauss-Krüger? ...

Output: APPLICATION/GIF:

- Units of measure?

**Candidate technology:** OWLS (Ontology Web Language for Web Services)

OWL-based Web service ontology, which supplies web service providers with a core set of constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form.

There exist already a variety of free or commercial Web services; provided especially by Internet companies, such as Google, Amazon or Yahoo.

**Example Web Services**

- Apache Axis
- Web Services and Java
- Integration and WS Standards
- Supporting - Restraining Forces
- Distributed Process Architecture
- Semantic Web Services