

Arsitektur Aplikasi Perangkat Enterprise #12



Antonius Rachmat C, S.Kom, M.Cs

What is WSDL?

- ❑ Stands for “Web Service Description Language”
- ❑ WSDL is a document written in XML
- ❑ The document **describes** a Web service
- ❑ Specifies the **location** of the service and the **methods** the service exposes
- ❑ Not W3C Standard
 - Version 1.1 released March 2001
 - Version 2.0 released June 2007
- ❑ MIME: application/wsdl+xml

WSDL

- Once you develop a Web Service:
 - you publish its description
 - and optionally a link to it in a UDDI repository so that potential users can find it...
- When someone wants to use your service, they request the WSDL file in order to:
 - find out the location of the service,
 - the function calls
 - and how to access them
- Then they use this information in your WSDL file to:
 - form a SOAP request to the computer

Why WSDL?

- ❑ Without WSDL, calling syntax must be determined from **documentation** that **must be provided**, or from examining wire messages
- ❑ With WSDL, the generation of proxies for Web services is **automated** in a truly language- and platform-independent way

Working of WSDL

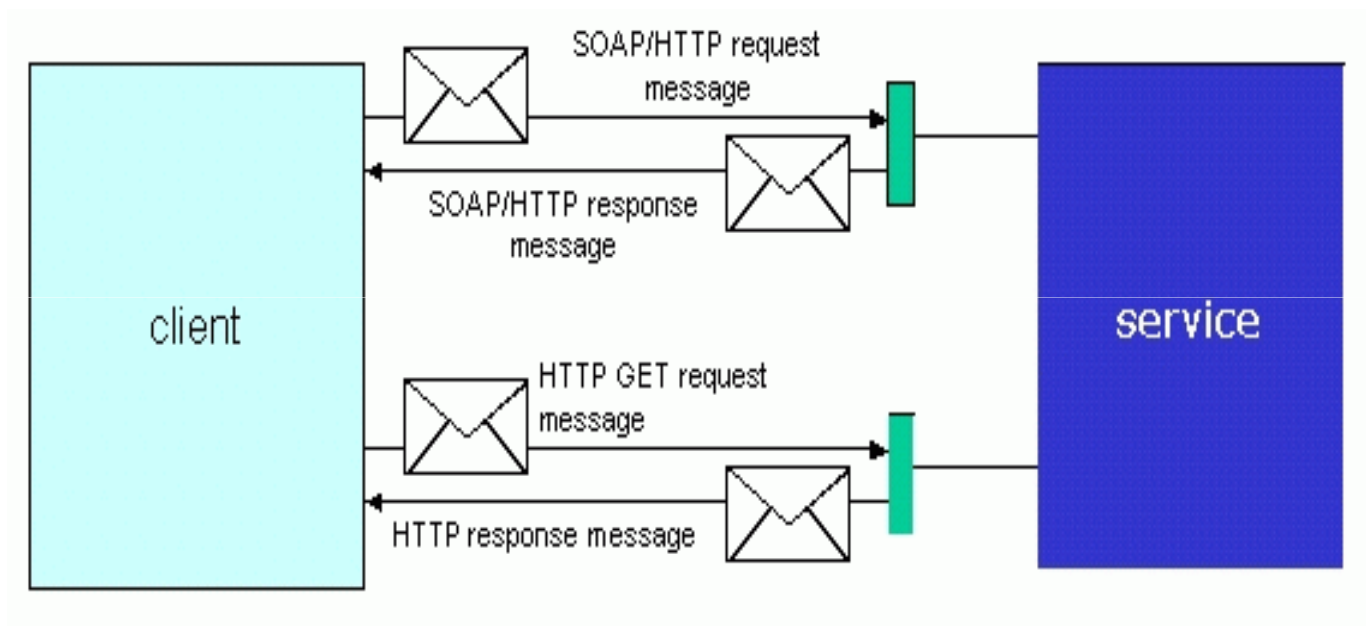


Figure 1. A client invoking a Web service.

WSDL Goal

- **Service description:** documentation for distributed systems
 - Language- and platform-independent
- **Service automation:** recipe for automating the details involved in the service invocation

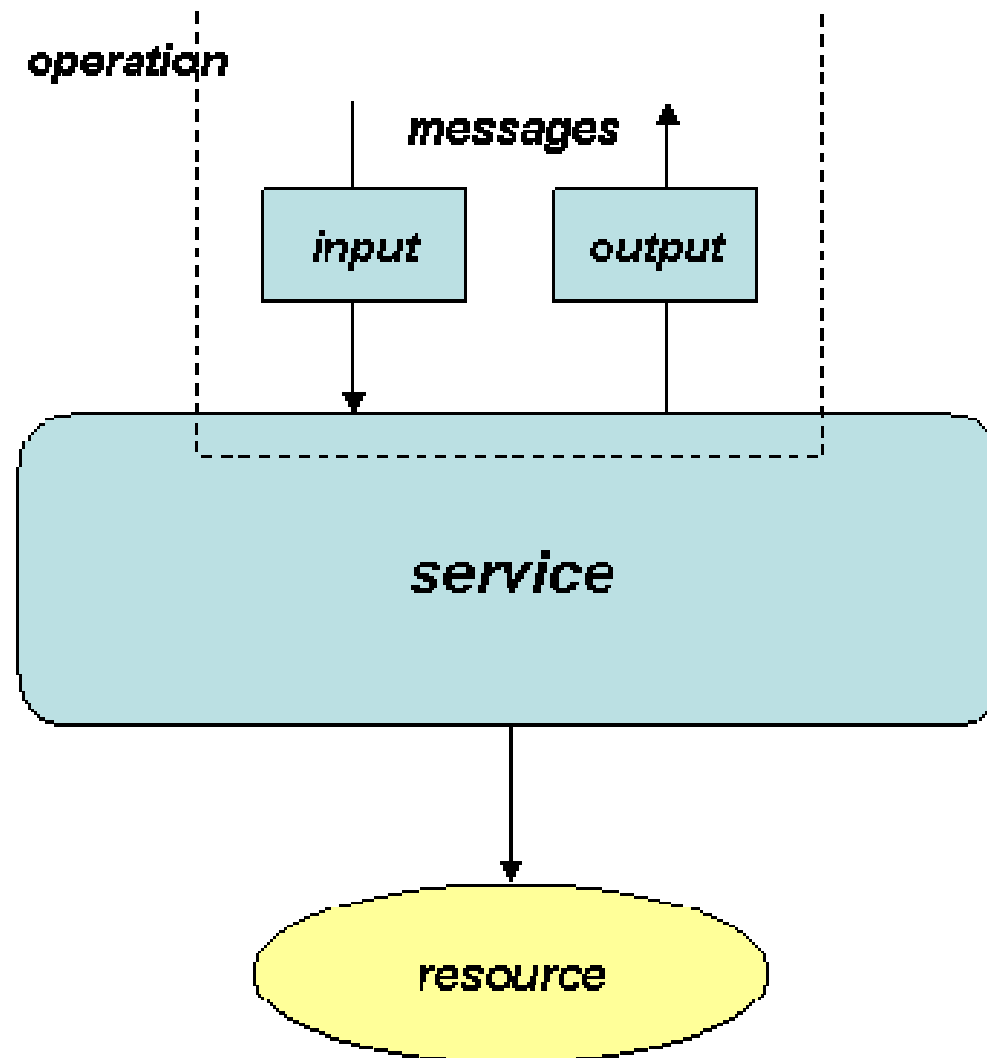
WSDL Elements



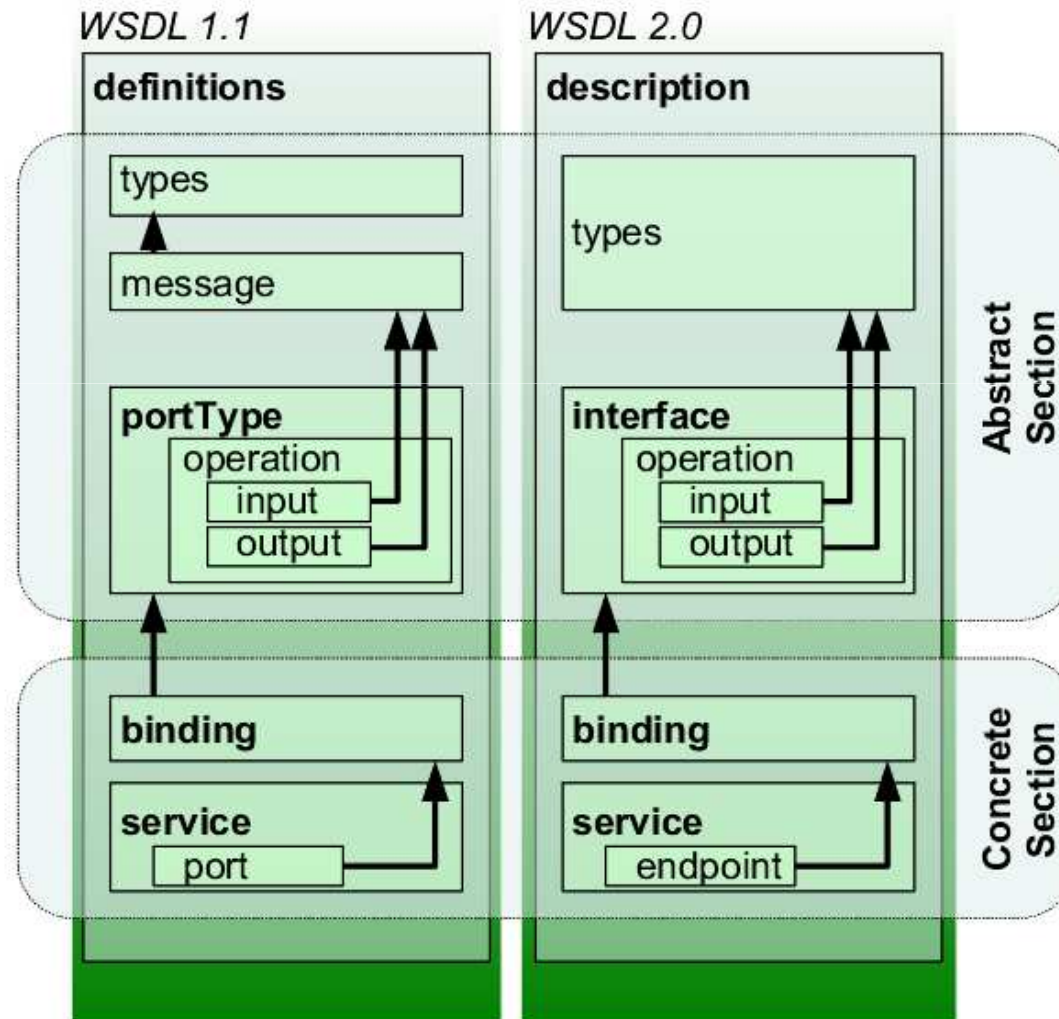
Element	Defines
<definitions>	Merupakan root element, mendefinisikan nama web service
<portType>	Operasi-operasi yang dilakukan oleh web service dan messages yang terlibat. Merupakan elemen terpenting dan dapat dianalogikan sebagai function library dalam bahasa pemrograman.
<message>	Message yang digunakan oleh web service, mendefinisikan elemen data dari operasi-operasi yang dilakukan. Dapat dianalogikan sebagai parameters dari fungsi-fungsi operasi.
<types>	Tipe data yang digunakan oleh web service
<binding>	Protocol komunikasi yang digunakan oleh web service, mendefinisikan message format dan detail protocol untuk setiap port
<service>	Specifies port address(es) of each binding.



Big Picture of WSDL Elements



WSDL 1.1 dan 2.0



The Main Structure of WSDL

<definition **namespace** = "http/... ">

Bagian Abstrak

<**type**> xschema types </type>

<**message**> ... </message>

<**port**> a set of operations </port>

Bagian Konkret

<**binding**> communication protocols </binding>

<**service**> a list of binding and ports </service>

<definition>

Namespace used

- ❑ The XML namespace prefix are used to indicate the **namespace** of the element being defined
- ❑ All WSDL elements belong to the WSDL namespace, defined as
 - xmlns=http://schemas.xmlsoap.org/wsdl/
 - xmlns:xsd=http://www.w3.org/2001/XMLSchema
 - xmlns:soap=http://schemas.xmlsoap.org/wsdl/soap/
 - xmlns:soapenc=http://schemas.xmlsoap.org/soap/encoding/
 - xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"

Amazon WSDL

```
<wsdl:definitions
  xmlns:typens="http://soap.amazon.com"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://soap.amazon.com"
  name="AmazonSearch">
```

Google WSDL

```
<definitions name="GoogleSearch"
  targetNamespace="urn:GoogleSearch"
  xmlns:typens="urn:GoogleSearch"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
```

Types Section

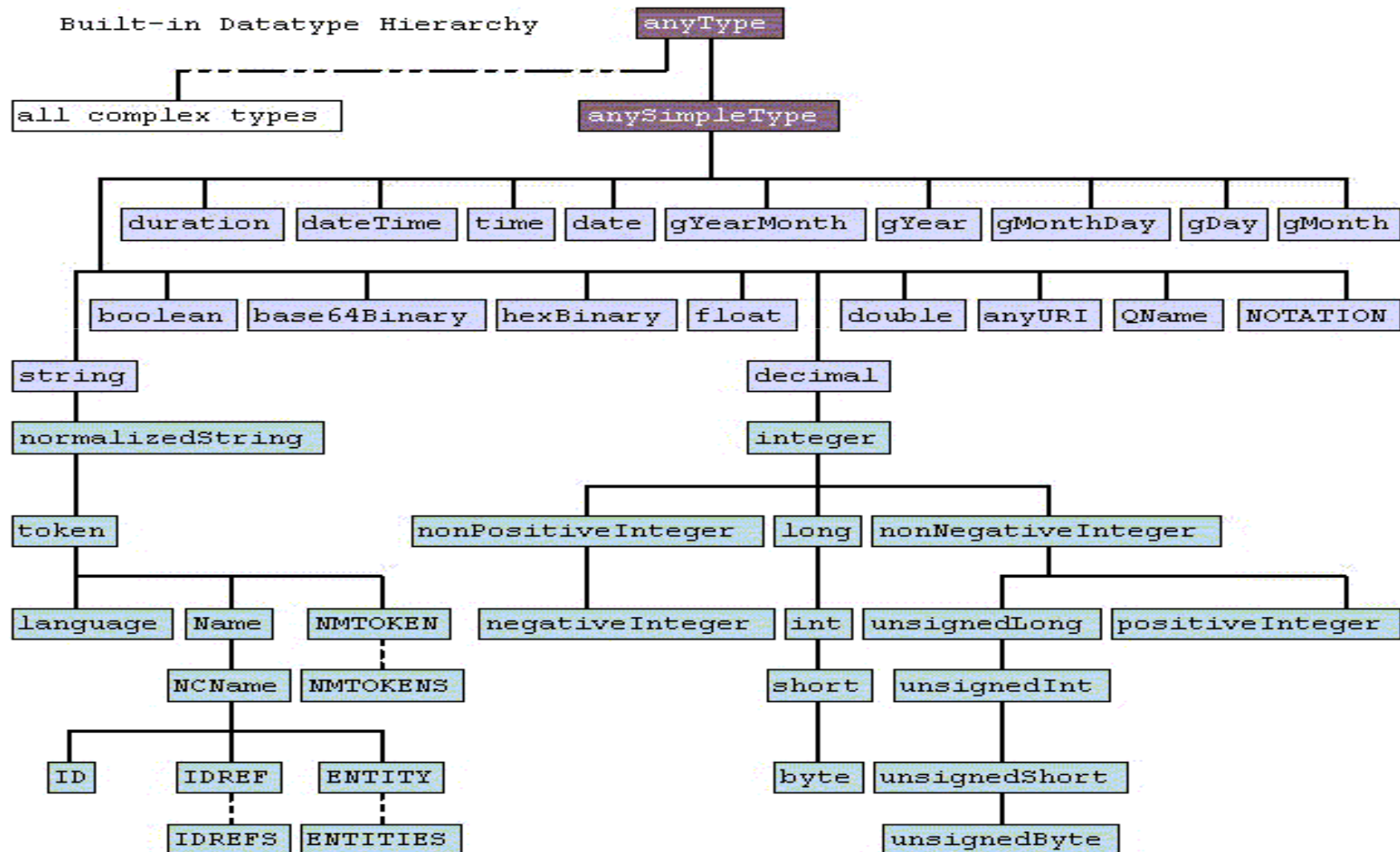
- The ***type*** element defines **the data types** that are used by the web service.
- ```
<xsd:complexType name="PERSON">
 <xsd:sequence>
 <xsd:element name="firstName" type="xsd:string"/>
 <xsd:element name="lastName" type="xsd:string"/>
 <xsd:element name="ageInYears" type="xsd:int"/>
 </xsd:sequence>
</xsd:complexType>
```

# Google Data Type

---

```
<xsd:complexType name="ResultElement">
 <xsd:all>
 <xsd:element name="summary" type="xsd:string"/>
 <xsd:element name="URL" type="xsd:string"/>
 <xsd:element name="snippet" type="xsd:string"/>
 <xsd:element name="title" type="xsd:string"/>
 <xsd:element name="cachedSize" type="xsd:string"/>
 <xsd:element name="relatedInformationPresent" type="xsd:boolean"/>
 <xsd:element name="hostName" type="xsd:string"/>
 <xsd:element name="directoryCategory" type="typens:DirectoryCategory"/>
 <xsd:element name="directoryTitle" type="xsd:string"/>
 </xsd:all>
</xsd:complexType>
```

# Remember: Schema Built In Types



# Message

---

- A message is **protocol independent**
- There is an **input** or **request** message, which is sent from the client to the service, and there is a **output** or **response** message, which is sent back the opposite way
- Each **<message>** element contains one or more **<part>** elements.
- **<part>** element corresponds to the **parameter** or a return value in the RPC call.

# Messages Section

---

- ❑ A *message* element defines **function name**
- ❑ The name of an output message element ends in "**Response**" by convention
- ❑ 

```
<message name="doGetCachedPage">
 <part name="key" type="xsd:string"/>
 <part name="url" type="xsd:string"/>
</message>
```
- ❑ 

```
<message name="doGetCachedPageResponse">
 <part name="return" type="xsd:base64Binary"/>
</message>
```
- ❑ 

```
<message name="doSpellingSuggestion">
 <part name="key" type="xsd:string"/>
 <part name="phrase" type="xsd:string"/>
</message>
```
- ❑ 

```
<message name="doSpellingSuggestionResponse">
 <part name="return" type="xsd:string"/>
</message>
```

# PortTypes Section

---

- ❑ Defines the **operations that can be performed**, and the messages that are involved.
- ❑ Operation defines which message is the **input** and which message is the **output**

```
<portType name="GoogleSearchPort">

 <operation name="doGetCachedPage">
 <input message="typens:doGetCachedPage"/>
 <output message="typens:doGetCachedPageResponse"/>
 </operation>

 <operation name="doSpellingSuggestion">
 <input message="typens:doSpellingSuggestion"/>
 <output message="typens:doSpellingSuggestionResponse"/>
 </operation>

 <operation name="doGoogleSearch">
 <input message="typens:doGoogleSearch"/>
 <output message="typens:doGoogleSearchResponse"/>
 </operation>

</portType>
```

# WSDL portTypes

---

- WSDL **messages** are only **abstract** messages.
  - We bind them to ***operations*** within the portType.
- The structure of the portType specifies (**still abstractly**) how the messages are to be used.
  - operations->programming methods
  - portTypes->interfaces

# PortType Operation

---

- Tipe operasi PortType:
  - One-way: operation dapat menerima message tapi tidak mengembalikan response

```
<message name="newTermValues">
 <part name="term" type="xs:string"/>
 <part name="value" type="xs:string"/>
</message>
```

```
<portType name="glossaryTerms">
 <operation name="setTerm">
 <input name="newTerm" message="newTermValues"/>
 </operation>
</portType>
```

# PortType Operation

---

- Request-response: operation dapat menerima message dan akan mengembalikan response

```
<message name="getTermRequest" >
 <part name="term" type="xs:string" />
</message>

<message name="getTermResponse" >
 <part name="value" type="xs:string" />
</message>

<portType name="glossaryTerms" >
 <operation name="getTerm" >
 <input message="getTermRequest" />
 <output message="getTermResponse" />
 </operation>
</portType>
```

# PortType Operation

---

- Notification: operation hanya memberikan response saja

```
<message name="getBilAcak">
 <part name="bil" type="xsd:int">
</message>
```

```
<portType name="BilanganAcak">
 <operation name="getBilanganAcak">
 <output message="getBilAcak"/>
 </operation>
</portType>
```

# SOAP Binding tags

---

- Binding defines **how message are transmitted**, and the location of the service.
  - SOAP, HTTP GET/POST, and MIME are provided in the WSDL specification.
- Bindings refer back to **portTypes** by name, just as **operations** point to **messages**.

# SOAP Binding

---

- **<soap:binding>** - Signifies that the binding is bound to the SOAP protocol format: Envelope, Header and Body

```
<binding ...>
```

```
 <soap:binding transport="uri"? Style="rpc|document"?>
```

```
</binding>
```

- **<soap:operation>** - Provides information for the document as a whole

```
<binding ...>
```

```
 <operation ...>
```

```
 <soap:operation soapAction="uri"? Style="rpc|document"?>
```

```
 <input> </input>
```

```
 <output> </output>
```

```
 </operation>
```

```
</binding>
```

# SOAP Binding

---

- **<soap:body>** - Specifies how the message parts appear inside the SOAP Body element

```
<input>
```

```
 <soap:body use="literal|encoded"?
 encodingStyle="uri-list"? Namespace="uri"?>
```

```
</input>
```

- **<soap:fault>** - Specifies the contents of the contents of the SOAP fault

```
<fault>
```

```
 <soap:fault name="nmtoken" use="literal|encoded"
 encodingStyle="uri-list"? Namespace="uri"?>
```

```
</fault>
```

# SOAP binding

---

- **<soap:header> and <soap:headerfault>** - Allow headers to be defined that are transmitted inside the Header element of the SOAP Envelope

```
<input>
```

```
 <soap:header message="qname" part="nmtoken" use="literal|encoded"?
 encodingStyle="uri-list"? Namespace="uri"?>
```

```
 <soap:headerfault message="qname" part="nmtoken"
 use="literal|encoded"? encodingStyle="uri-list"? Namespace="uri"?>
```

```
</input>
```

- **<soap:address>** - Used to give a port an address (a URI)

```
<binding ...>
```

```
 <soap:address location="uri" />
```

```
</binding>
```

# Google Binding

---

```
<binding name="GoogleSearchBinding" type="typens:GoogleSearchPort">
 <soap:binding style="rpc"
 transport="http://schemas.xmlsoap.org/soap/http"/>

 <operation name="doGoogleSearch">
 <soap:operation soapAction="urn:GoogleSearchAction"/>
 <input>
 <soap:body use="encoded"
 namespace="urn:GoogleSearch"
 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
 </input>
 <output>
 <soap:body use="encoded"
 namespace="urn:GoogleSearch"
 encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
 </output>
 </operation>
</binding>
```

# Element Service

---

- Digunakan untuk menamai keseluruhan service pada sebuah web service beserta nama port, binding, dan lokasi web service tersebut

```
<service name="GoogleSearchService">
 <port name="GoogleSearchPort"
 binding="typens:GoogleSearchBinding">
 <soap:address location="http://api.google.com/search/beta2"/>
 </port>
</service>
```

# WSDL 2.0

---

- ❑ Adding further semantics to the description language
- ❑ Removal of message constructs
- ❑ No support for operator overloading
- ❑ PortTypes renamed to interfaces
- ❑ Ports renamed to endpoint.

# WSDL Tool

---

□ <http://xmmethods.net/ve2/Tools.po>

## X METHODS

### WSDL Analyzer : Service Definitions

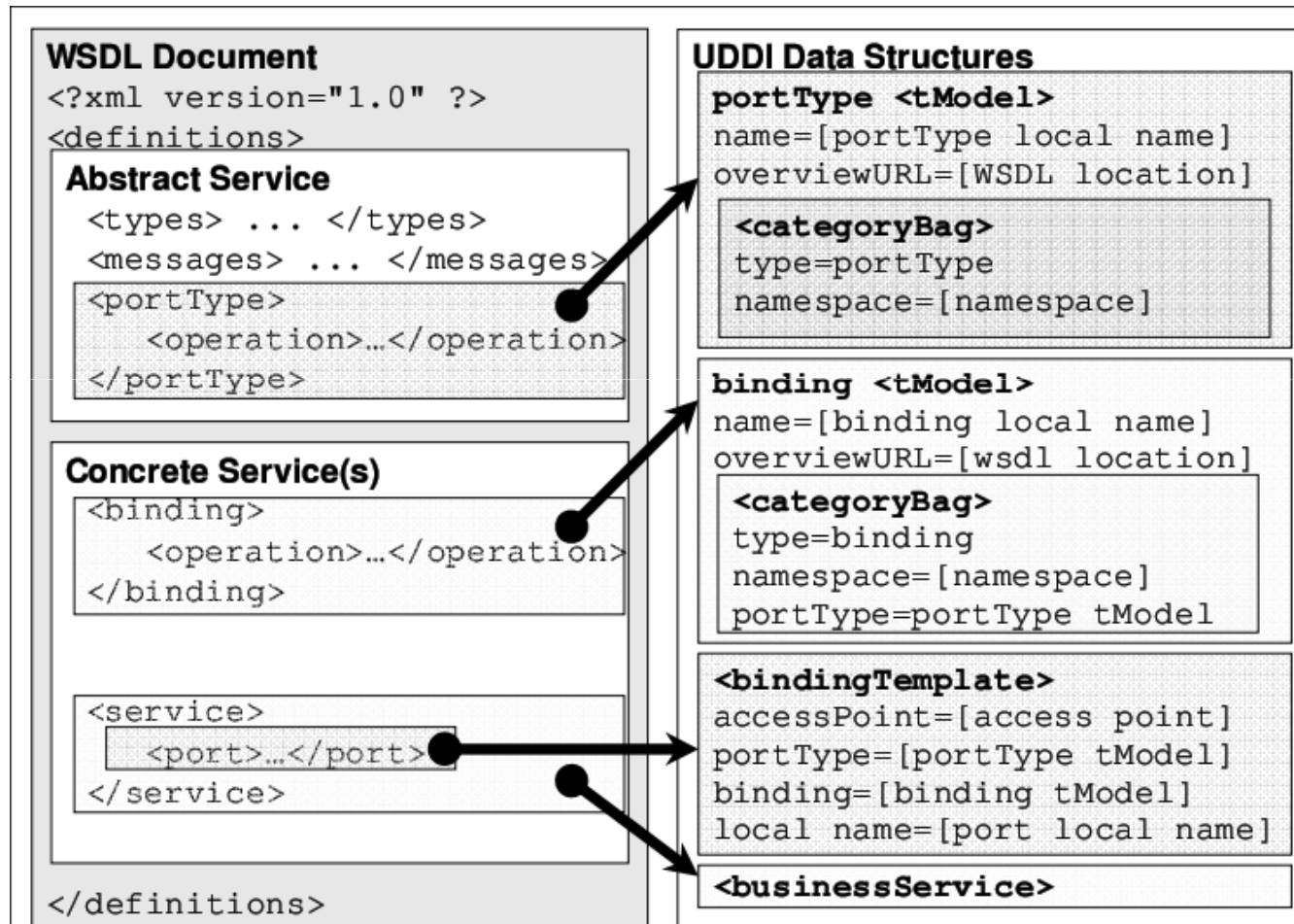
for the WSDL file <http://www.yukawa.de/infect/infectOpen.wsdl>

The following table lists the service(s) defined in the WSDL file. You can drill down into the operations (methods) defined for that service by clicking on the **operations** link.

Service [Port]	Operations	Default Style	Transport	Endpoint
InfectOpenWebService [InfectOpenWebPort]	<a href="#">5 operations</a>	Document	HTTP/S	<a href="http://service.yukawa.de/chain/infect/ws/InfectOpenWebService">http://service.yukawa.de/chain/infect/ws/InfectOpenWebService</a>

Operation / Method Name	SOAPAction <sup>+</sup>	Style	Input Message	Output Message
createImmunity	[Empty String]	document	<a href="#">Input Msg</a>	
killInfection	[Empty String]	document	<a href="#">Input Msg</a>	
setInfectedEmail	[Empty String]	document	<a href="#">Input Msg</a>	
spreadInfection	[Empty String]	document	<a href="#">Input Msg</a>	
whatsMyInfection	[Empty String]	document	<a href="#">Input Msg</a>	

# Hubungan WSDL & UDDI



# NEXT

---

- Web Service Implementation in VB.NET
  - Tipe data primitif
  - Tipe data array dan record
  - Tipe data class
  - OOP di VB.NET