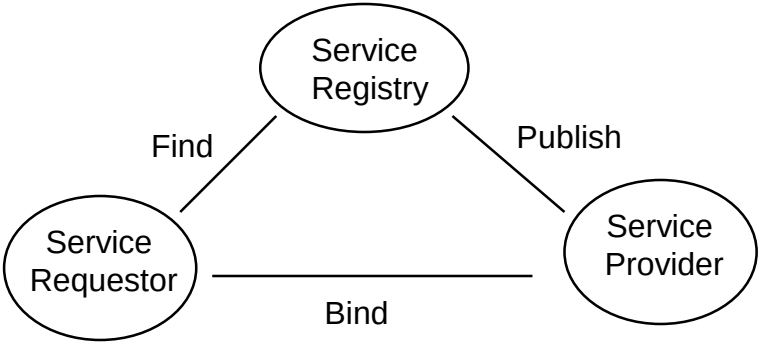
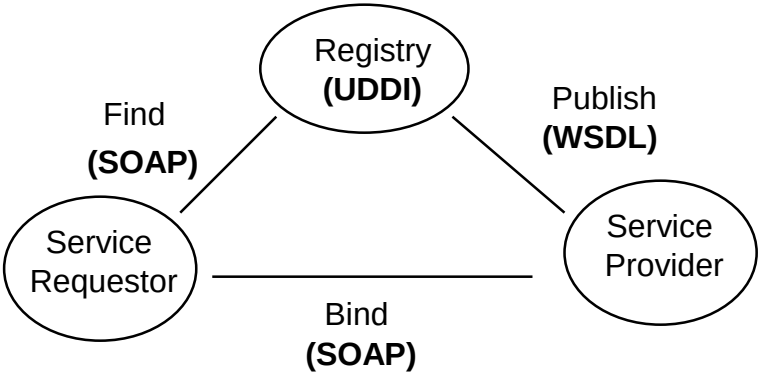
Web Services

Web Service Architecture



- Service-Oriented Architecture

Architecture



- SOAP: Simple Object Access Protocol
- WSDL: Web Services Definition Language
- UDDI: Universal Description, Discovery, and Integration
- All the technologies are **XML based**

SOAP

- **S**imple **O**bject **A**ccess **P**rotocol
- SOAP (Simple Object Access Protocol) is a protocol specification for exchanging structured information in the implementation of web services in computer networks.
- Communicate using HTTP and XML.
- Web service **messaging and invocation**

SOAP

- an application communication protocol
- a format for sending and receiving messages
- platform independent
- based on XML
- W3C recommendation

SOAP History

1998
- Term *SOAP* coined at Microsoft

1999
- Microsoft works with BizTalk to release SOAP 0.9
- Submitted to IETF
- SOAP 1.0 released in December

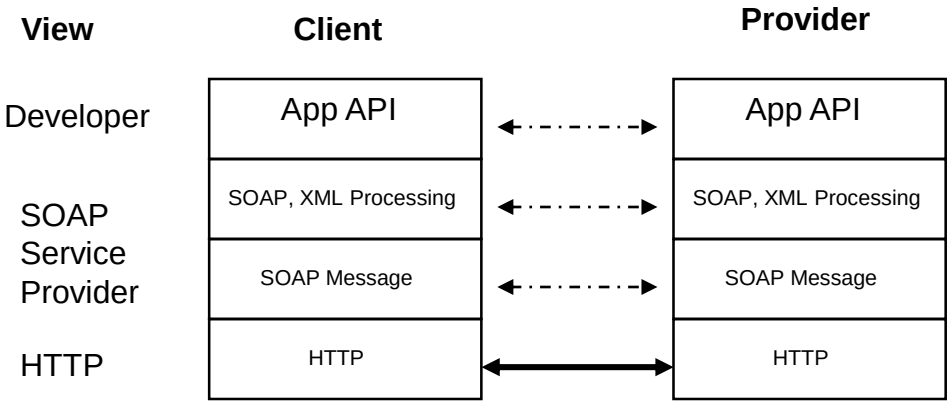
2000
- SOAP 1.1 submitted to W3C with IBM
- IBM releases a Java SOAP implementation
- Sun starts work on Web services in J2EE

2001
- SOAP 1.2 released by XML Protocol working group at W3C

Why SOAP?

- It is important for web applications to be able to communicate over the Internet.
- The best way to communicate between applications is over HTTP
- HTTP is supported by all Internet browsers and servers.
- SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

SOAP Messaging Layers



SOAP Message

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope>
  <Header>

  </Header>
  <Body>

  </Body>
</Envelope>
```

SOAP Building Blocks

- An **Envelope** element that identifies the XML document as a SOAP message
- A **Header** element that contains header information
- A **Body** element that contains call and response information
- A **Fault** element containing errors and status information

Syntax Rules

- A SOAP message MUST be encoded using XML
- A SOAP message MUST use the SOAP Envelope namespace
- A SOAP message MUST use the SOAP Encoding namespace
- A SOAP message must NOT contain a DTD reference
- A SOAP message must NOT contain XML Processing Instructions

Sample

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Header>
...
</soap:Header>
<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

The SOAP Envelope Element

- SOAP Envelope element is the **root element of a SOAP message.**
- It defines the XML document as a SOAP message.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
  ...
  Message information goes here
  ...
</soap:Envelope>
```

The xmlns:soap Namespace

- It should always have the value of: "http://www.w3.org/2003/05/soap-envelope/".
- The namespace defines the Envelope as a SOAP Envelope.

The encodingStyle Attribute

- The encodingStyle attribute is used to define the data types used in the document.
- A SOAP message has no default encoding.

soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

The SOAP Header Element

- The optional SOAP Header element contains application-specific information (like authentication, payment, etc) about the SOAP message.
- First Child element of Envelope

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Header>
  <m:Trans xmlns:m="http://www.w3schools.com/transaction/"
  soap:mustUnderstand="1">
  </m:Trans>
</soap:Header>
...
...
</soap:Envelope>
```

SOAP (attributes default namespace)

- SOAP defines three attributes in the default namespace.
- These attributes are:
  - mustUnderstand,
  - actor, and
  - encodingStyle.
- These defines how a recipient should process the SOAP message.

The mustUnderstand Attribute

- The SOAP mustUnderstand attribute can be used to indicate whether a header entry is mandatory or optional for the recipient to process.
- Syntax : soap:mustUnderstand="0|1"
- mustUnderstand="1" indicates the receiver processing the Header must recognize the element.

The actor Attribute

- The SOAP actor attribute is used to address the Header element to a specific endpoint.
- Syntax: soap:actor="URI"

An actor is an application that can both receive SOAP messages and forward them to the next actor. The ability to specify one or more actors as intermediate recipients makes it possible to route a message to multiple recipients and to supply header information that applies specifically to each of the recipients.

The encodingStyle Attribute

- The encodingStyle attribute is used to define the data types used in the document.
- Syntax: soap:encodingStyle="*URI*"

The SOAP Body Element

- It contains the actual SOAP message intended for the ultimate endpoint of the message.
- The example below **requests** the price of apples.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
    <m:Item>Apples</m:Item>
  </m:GetPrice>
</soap:Body>
</soap:Envelope>
```

m:GetPrice and the **Item elements** above **are application-specific elements.**

A SOAP response

```
<?xml version="1.0"?>

<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body>
  <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
    <m:Price>1.90</m:Price>
  </m:GetPriceResponse>
</soap:Body>

</soap:Envelope>
```

The SOAP Fault Element

- The optional SOAP Fault element is used to indicate error messages.
- The SOAP Fault element has the following sub elements:

Sub Element	Description
<faultcode>	A code for identifying the fault
<faultstring>	A human readable explanation of the fault
<faultactor>	Information about who caused the fault to happen
<detail>	Holds application specific error information related to the Body element

The HTTP Protocol

- HTTP communicates over TCP/IP.
- After establishing a connection, the client can send a HTTP request message to the server:
- POST /item HTTP/1.1
  Host: 189.123.255.239
  Content-Type: text/plain
  Content-Length: 200

  200 OK
  Content-Type: text/plain
  Content-Length: 200

SOAP Protocol Binding: HTTP

```
POST /ServiceLoc HTTP/1.1
Host: www.foo.com
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Directory/Service"

<?xml version="1.0" encoding="UTF-8"?>
<Envelope>

  <Header>

  </Header>

  <Body>
    <LookupPerson …>

    </LookupPerson>
  </Body>

</Envelope>
```

Out-of-message context

In-message context

Sample RPC Call

SOAP Binding

- SOAP bindings are mechanisms which allow SOAP messages to be effectively exchanged using a transport protocol.
- HTTP (widely used) or SMTP.

Content-Type

- The Content-Type header for a SOAP request and response defines the MIME type (Multipurpose Internet Mail Extensions) for the message and the character encoding (optional) used for the XML body of the request or response.
- **Syntax:**
  Content-Type: MIMEType; charset=character-encoding
- **Example**

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
```

Content-Length

- It specifies the number of bytes in the body of the request or response.

```
POST /item HTTP/1.1
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 250
```

A SOAP Example (A SOAP request)

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 80

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>
</soap:Envelope>
```

The SOAP response

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: 80

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

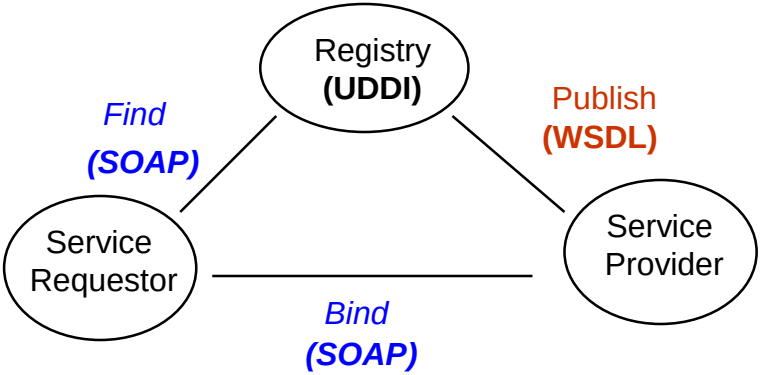
Data encoding in SOAP

- SOAP provides default encoding schema
- Simple data types
  - Use “xsi:type”
  - String, floats, etc
- Complex data types
  - SOAP arrays
  - Structs: compound types
- Data referencing
  - Href and id attributes

Ref

- http://www.w3.org/2003/05/soap-envelope
- http://www.w3.org/2003/05/soap-encoding
- http://www.w3schools.com/xml/xml_soap.asp

Roadmap


Registry
(UDDI)
Find
(SOAP)
Publish
(WSDL)
Service
Requestor
Service
Provider
Bind
(SOAP)

WSDL

- **W**eb **S**ervices **D**efinition **L**anguage
- WSDL is used to **describe web services**
- WSDL is written in XML
- WSDL is a W3C recommendation from 26 June 2007
- https://www.w3.org/TR/wsdl

WSDL

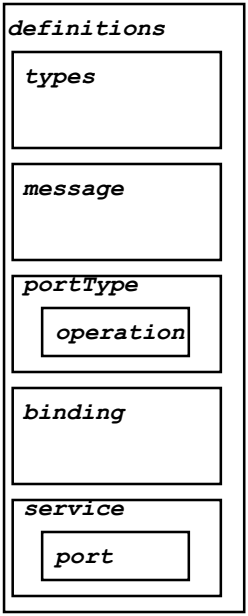
- Define a web service in WSDL by
  - Writing an XML document conforming to the WSDL specs
- Describes three fundamental properties
  - What a service does
    - Operations (methods) provided by the service
  - How a service is accessed
    - Data format and protocol details
  - Where a service is located
    - Address (URL) details

WSDL Documents

- An WSDL document describes a web service using:

Element	Description
<types>	Defines the (XML Schema) **data types** used by the web service
<message>	Defines the **data elements** for each operation
<portType>	Describes the operations that can be performed and the messages involved.
<binding>	Defines the protocol and data format for each port type

WSDL Structure



All the data types used by the Web service

Parameters and messages used by method

Abstract interface definition – each *operation* element defines a method signature

Binds abstract methods to specific protocols

A service is a collection of ports.
A port is a specific method and its URI

structure of a WSDL document

```
<definitions>

<types>
  data type definitions........
</types>

<message>
  definition of the data being communicated....
</message>

<portType>
  set of operations......
</portType>

<binding>
  protocol and data format specification....
</binding>
```

WSDL Example

```
<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> element defines "glossaryTerms" as the name of a **port**, and "getTerm" as the name of an **operation**.
The "getTerm" operation has an **input message** called "getTermRequest" and an **output message** called "getTermResponse".
The **<message>** elements define the **parts** of each message and the associated data types.

The <portType> Element

The <portType> element defines **a web service**, the **operations** that can be performed, and the **messages** that are involved.

Type		Definition
One-way	input	The operation can receive a message but will not return a response
Request-response	input,output	The operation can receive a request and will return a response
Solicit-response	output,input	The operation can send a request and will wait for a response
Notification	output	The operation can send a message but will not wait for a response

WSDL One-Way Operation

```
<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 >
```

The "setTerm" operation allows input of new glossary terms messages using a "newTermValues" message with the input parameters "term" and "value".

WSDL Request-Response Operation

```
<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>
```

WSDL Binding to SOAP

- The **binding** element has two attributes - name and type.
- The name attribute (you can use any name you want) defines the name of the binding, and the type attribute points to the port for the binding
- The **soap:binding** element has two attributes - style and transport.
- The style attribute can be "rpc" or "document".
- The transport attribute defines the SOAP protocol to use.
- The **operation** element defines each operation that the portType exposes.
- literal➔ how the input and output are encoded

WSDL Binding to SOAP

```
<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>
```

```
<binding type="glossaryTerms" name="b1">
  <soap:binding style="document"
        transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation soapAction="http://example.com/getTerm"/>
    <input><soap:body use="literal"/></input>
    <output><soap:body use="literal"/></output>
  </operation>
</binding>
```

- The **binding** element- name and type.➔the "glossaryTerms" port.
- The **soap:binding** element has two attributes - style and transport.
- The style attribute ➔document.
- The transport attribute ➔HTTP.
- literal➔ how the input and output are encoded

Sample WSDL: getQuote

```
<?xml version="1.0" encoding="UTF-8" ?>

<definitions
name="net.xmethods.services.stockquote.StockQuote"
targetNamespace="http://www.themindelectric.com/wsdl/net.xmet
hods.services.stockquote.StockQuote/"
xmlns:tns="http://www.themindelectric.com/wsdl/net.xmethods.s
ervices.stockquote.StockQuote/"
   xmlns:electric="http://www.themindelectric.com/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">

<message name="getQuoteResponse1">
    <part name="Result" type="xsd:float" />
</message>

<message name="getQuoteRequest1">
    <part name="symbol" type="xsd:string" />
</message>
```

Sample WSDL: getQuote

```
<portType name="net.xmethods.services.stockquote.StockQuotePortType">
   <operation name="getQuote" parameterOrder="symbol">
     <input message="tns:getQuoteRequest1" />
     <output message="tns:getQuoteResponse1" />
   </operation>
</portType>

<binding name="net.xmethods.services.stockquote.StockQuoteBinding"
         type="tns:net.xmethods.services.stockquote.StockQuotePortType">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="getQuote">
       <soap:operation soapAction="urn:xmethods-delayed-quotes#getQuote" />
         <input>
           <soap:body use="encoded" namespace="urn:xmethods-delayed-quotes"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
         </input>
         <output>
           <soap:body use="encoded" namespace="urn:xmethods-delayed-quotes"
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
         </output>
    </operation>
</binding>
```

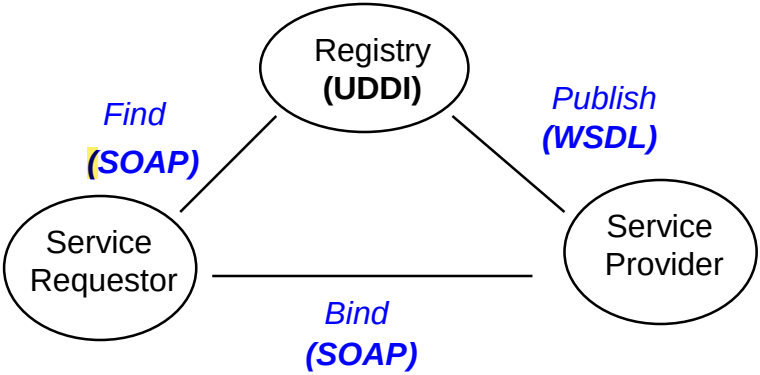
Sample WSDL: getQuote

```
<service
name="net.xmethods.services.stockquote.StockQuoteService">
  <documentation>net.xmethods.services.stockquote.StockQuote web
service
  </documentation>
   <port name="net.xmethods.services.stockquote.StockQuotePort"

binding="tns:net.xmethods.services.stockquote.StockQuoteBinding"
>
     <soap:address location="http://64.39.29.211:9090/soap" />
   </port>
</service>

</definitions>
```

WSDL to Code

- Translators available that can
  - Convert WSDL document to code
    - IBM's WSTK Toolkit
    - Apache AXIS WSDL2java program
    - Soapy.py in Python
  - Derive WSDL from Java classes
    - Apache WSDL program

Roadmap


Registry
(UDDI)
Find
(SOAP)
Publish
(WSDL)
Service
Requestor
Service
Provider
Bind
(SOAP)

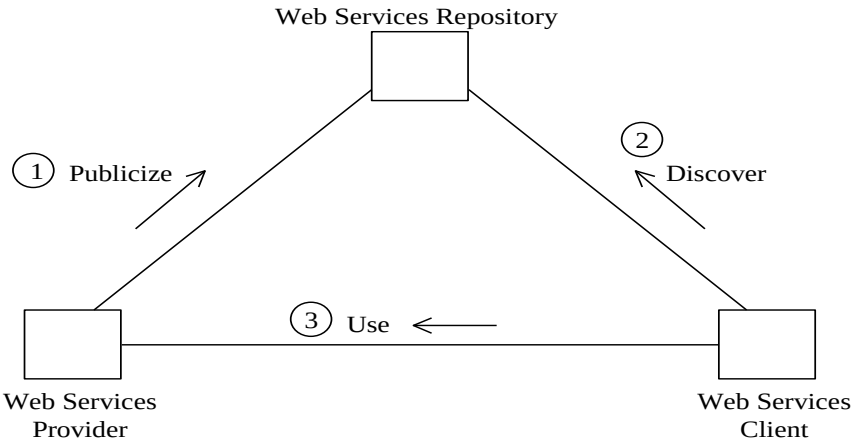
UDDI

- The Universal Description, Discovery, and Integration specs define a way to publish and discover information about Web services.
- The UDDI business registration is an XML file that describes a business entity and its Web services

UDDI

- Universal Description, Discovery, and Integration
- API for a Web based registry
- Implemented by an *Operator* Site
  - Replicate each others' information
- Formally announced in Sept, 2000
  - Collaboration between IBM, Microsoft, Ariba
  - Community of 310 companies

UDDI Schema


Web Services Repository
1
Publicize
2
Discover
3
Use
Web Services Provider
Web Services Client

UDDI Schema

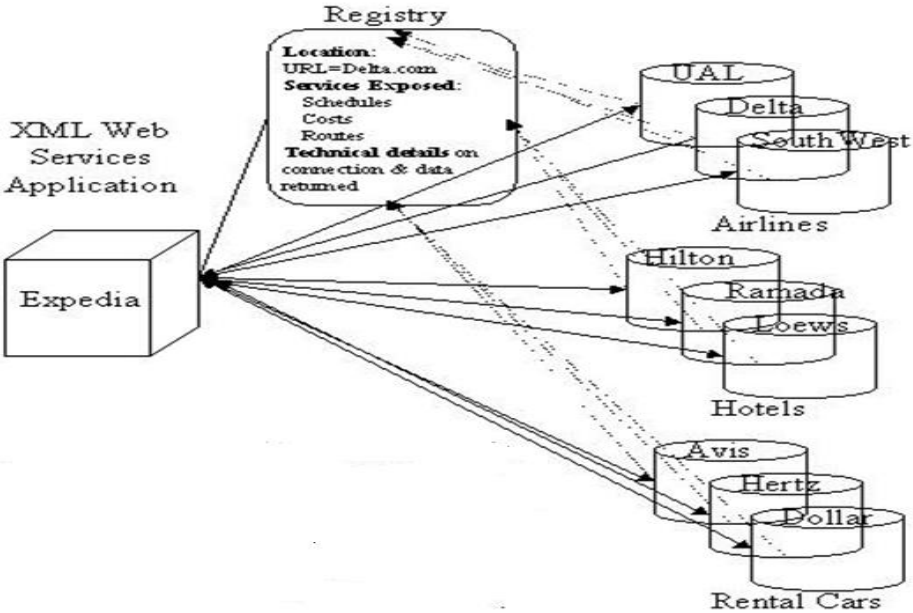


Fig 1

A UDDI Registry

- Who?
  - Basic business information
    - Name, contact information
- What?
  - Get classification
    - Standard Industry Codes, NA Industry Code Std
- Where?
  - Service URI
- How?
  - Describes a how a given interface functions

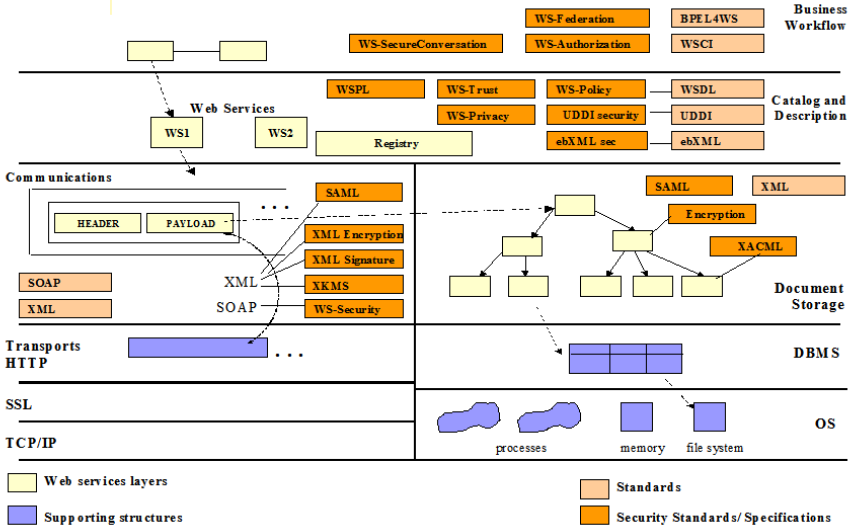
UDDI Data Structures

- businessEntity:
  - Basic business information
  - Used by UDDI for “yellow” pages
- businessService:
  - Services provided by that business
  - Grouping of related businesses
- bindingTemplate:
  - What the service looks like (tModel element)
  - Where to access the service

UDDI Data Structures

- tModel
  - Technology model
  - Could contain just about anything
  - Has service details
    - Abstract industry specs
    - Service specs
- Designed to be reusable
- Can contain pointer to WSDL document

Layers and Web Services Standards



Other UDDI Issues

- Security
  - No global standard
  - Each operator site must select/implement an authentication protocol that still allows publishing
- Versioning
  - Numbers not used
  - *Generic* element used in function calls

Open UDDI Issues

- Effective search
  - Classification and Categorization
- Private UDDI registries
  - E-marketplace
  - Portal
  - Partner catalog
  - Internal Application Integration

Overall Issues

- Interoperability
- Web Services Everywhere
  - Peer to peer vs centralized

Query Pattern

- Browse:
  - UDDI yellow page data has hierarchy
  - Search via Web/standalone client app
- Drill down:
  - Given a specific candidate, get all details
- Invocation

Inquiry API

- Generally accessible
  - Find_binding
  - Find_business, find_relatedBusiness
  - Find_service
  - Find_tmodel
  - Get_bindingDetail
  - Get_businessDetail
  - Get_serviceDetail
  - Get_tModelDetail
- Use SOAP to access

Publishing API

- Restricted access
  - Save_service, save_business, save_binding, save_tModel
  - Delete_service, delete_business, delete_binding, delete_tModel
  - Get_binding, get_registeredInfo, get_authToken
  - Add_publisherAssertions, get …, delete …

UDDI security

- Not specified in detail, only general policies
- Only authorized individuals can publish or change information in the registry
- Changes or deletions can only be made by the originator of the information
- Each instance of a registry can define its own user authentication mechanism