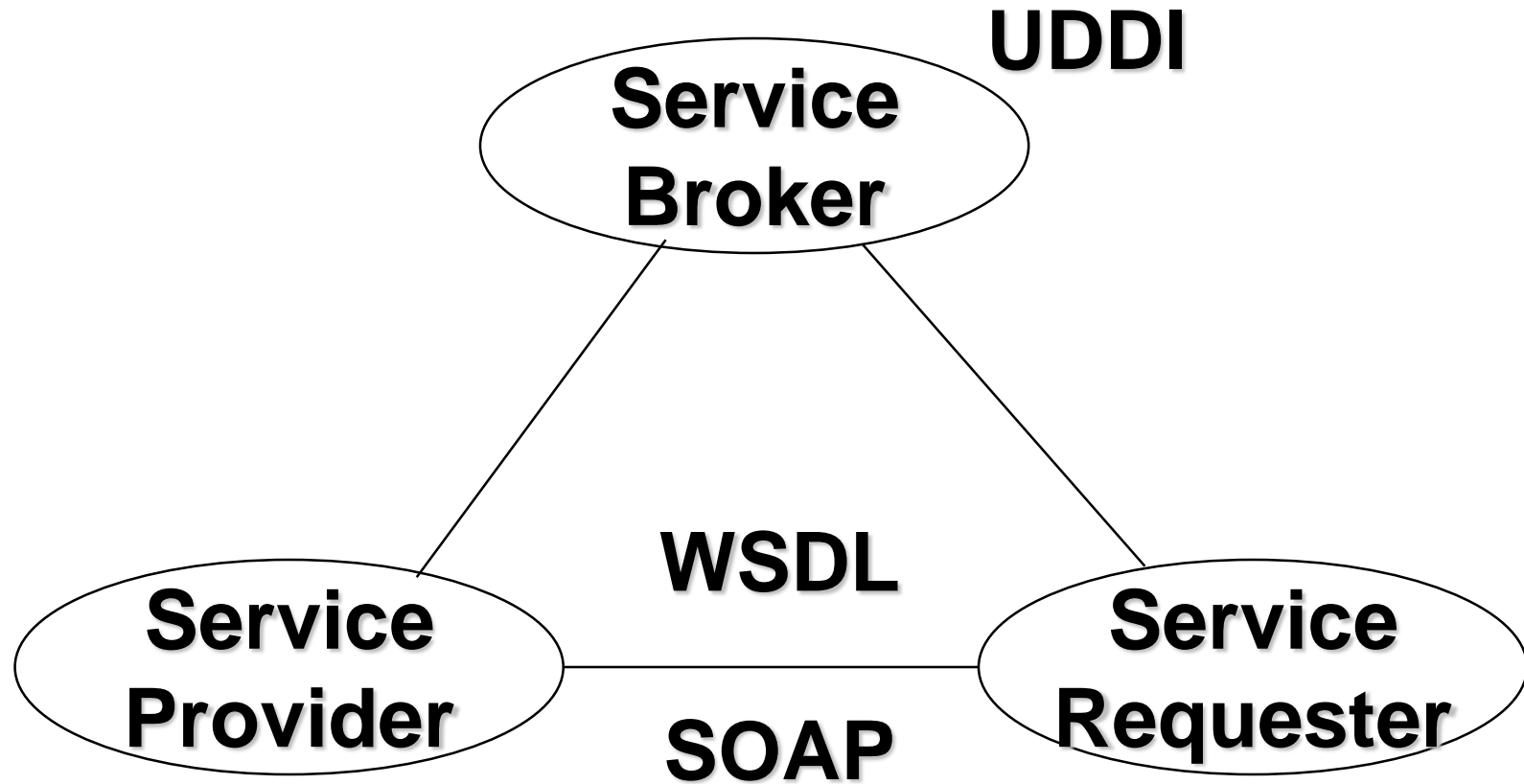


# State of the Art

- Sun RPC: XDR Streams + XDR + Portmapper
- RMI: RPC + Interfaces + Registry
- Corba: IIOP + IDL + ORB
- Web services: SOAP + WSDL + UDDI
  - RESTful:
  - JSON
- References:
  - <http://www.w3.org/TR/soap12-part0/>
  - <http://www.w3.org/TR/wsdl>
  - [http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=uddi-spec](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=uddi-spec)
  - <http://www.w3schools.com/default.asp>

# Service Oriented Application Architecture (SOA)





# Web Service Technology Stack

<b>Service Publication/Discovery</b>	<b>→ UDDI</b>
<b>Service Description WSDL</b>	<b>→ WSDL</b>
<b>Service Interaction</b>	<b>→ SOAP</b>
<b>Transport Network</b>	<b>→ HTTP, SMTP, FTP</b>

# What is SOAP?

- SOAP stands for Simple Object Access Protocol
- SOAP is a communication protocol
- SOAP is for communication between applications
- SOAP is a format for sending messages
- SOAP is designed to communicate via Internet
- SOAP is platform independent
- SOAP is language independent
- SOAP is based on XML
- SOAP is simple and extensible

# Why SOAP?

- It is important for application development to allow Internet communication between programs.
- Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.
- A better way to communicate between applications is over HTTP, because HTTP is supported by all Internet browsers and servers. SOAP was created to accomplish this.
- SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages.

# Background

- Microsoft and SOAP
  - SOAP is a key element of Microsoft's .NET architecture for future Internet application development.
- SOAP 1.1 was Proposed to W3C
  - UserLand, Ariba, Commerce One, Compaq, Developmentor, HP, IBM, IONA, Lotus, Microsoft, and SAP proposed to W3C, in May 2000, the SOAP Internet protocol that they hope will revolutionize application development by connecting graphic user interface desktop applications to powerful Internet servers using the standards of the Internet; HTTP and XML.
- W3C is Working on SOAP 1.2
  - The first public Working Draft on SOAP was published from W3C in December 2001. To read more about the SOAP activities at W3C please visit our W3C School.

# SOAP Building Blocks

- A SOAP message is an ordinary XML document containing the following elements:
  - A required Envelope element that identifies the XML document as a SOAP message
  - An optional Header element that contains header information
  - A required Body element that contains call and response information
  - An optional Fault element that provides information about errors that occurred while processing the message
  - All the elements above are declared in the default namespace for the SOAP envelope:

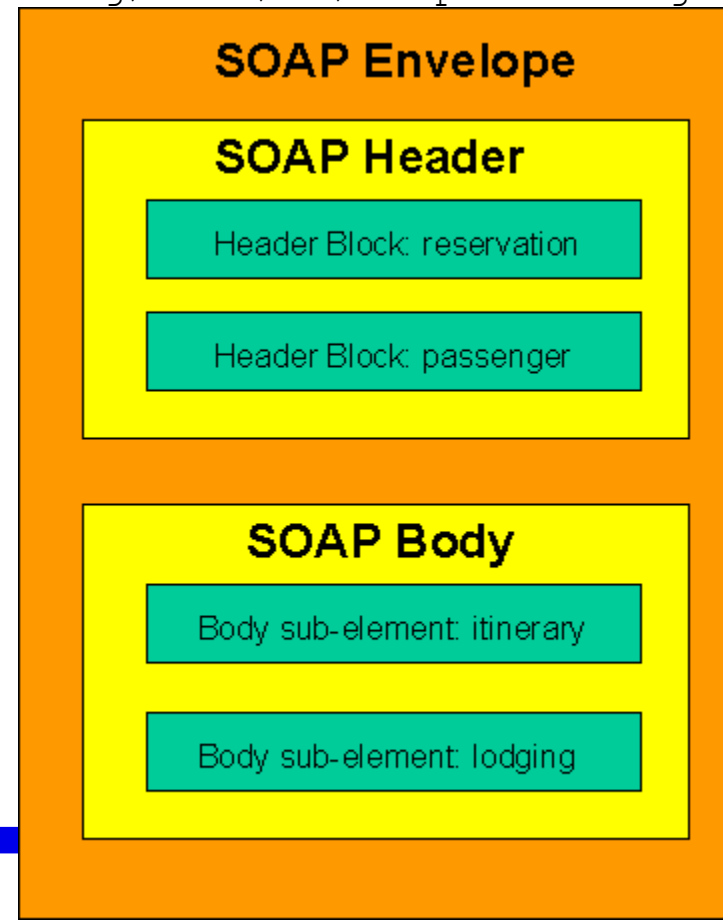


# Syntax Rules

- Here are some important 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

# Skeleton of SOAP Messages

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
    ...
    ...
  </soap:Header>
  <soap:Body>
    ...
    ...
    <soap:Fault>
      ...
      ...
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```



# Example of a SOAP Envelop: Request

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
    </m:GetPrice>
  </soap:Body>
</soap:Envelope>
```

# Example of a SOAP Envelope: Response

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
  soap:encodingStyle="http://www.w3.org/2001/12/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>
```

# What is WSDL?

- WSDL stands for Web Services Description Language
- WSDL is written in XML
- WSDL is an XML document
- WSDL is used to describe Web services
- WSDL is also used to locate Web services
- WSDL is not yet a W3C standard

# WSDL Describes Web Services

- WSDL stands for
  - Web Services Description Language.
- WSDL is a document written in XML.
  - It contains a set of definitions to define a web service.
  - It specifies the location of the service and the operations (or methods) the service exposes.

# WSDL Development History at W3C

- WSDL 1.1 was submitted as a W3C Note by Ariba, IBM and Microsoft for describing services for the W3C XML Activity on XML Protocols in March 2001.
  - (a W3C Note is made available by the W3C for discussion only. Publication of a Note by W3C indicates no endorsement by W3C or the W3C Team, or any W3C Members)
- The first Working Draft of WSDL 1.2 was released by W3C in July 2002.

# The WSDL Document Structure

- A WSDL document defines a web service using these major elements:
  - **types**, which provides data **type** definitions used to describe the messages exchanged.
  - **message**, which represents an abstract definition of the data being transmitted.
    - ▶ A message consists of logical parts, each of which is associated with a definition within some type system.
    - ▶ The parts are just like the **parameters** of a function call in a programming language.
  - **portType**, which is a set of abstract operations.
    - ▶ Each operation refers to an input message and output messages.
    - ▶ It defines a web service, the operations that can be performed, and the messages that are involved.
    - ▶ It is just like a **function library** (or a module, or a class) in a programming language.
  - **binding**, which specifies concrete **protocol** and **data format** specifications for the operations and messages defined by a particular portType.
  - **port**, which specifies an **address** for a binding, thus defining a single communication **endpoint**.
  - **service**, which is used to aggregate **a set of related ports**.



# WSDL Ports

- The <portType> element is the most important WSDL element.
- Operation Types
  - The request-response type is the most common operation type, but WSDL defines four types:
- Type Definition
  - One-way
    - ▶ The operation can receive a message but will not return a response
  - Request-response
    - ▶ The operation can receive a request and will return a response
  - Solicit-response
    - ▶ The operation can send a request and will wait for a response
  - Notification
    - ▶ The operation can send a message but will not wait for a response

# Example of Java Web Services

## ● Java Interface

```
package helloservice;  
  
import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
public interface HelloIF extends Remote {  
    public String sayHello(String s) throws RemoteException;  
}
```

# Config of WSDL in J2EE

```
<?xml version="1.0" encoding="UTF-8"?>
<webservices xmlns="http://java.sun.com/xml/ns/j2ee" version="1.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
    http://www.ibm.com/webservices/xsd/j2ee_web_services_1_1.xsd">
  <webservice-description>
    <display-name>MyHelloService</display-name>
    <webservice-description-name>MyHelloService</webservice-description-name>
    <wsdl-file>WEB-INF/wsdl/MyHelloService.wsdl</wsdl-file>
    <jaxrpc-mapping-file>build/mapping.xml</jaxrpc-mapping-file>
    <port-component>
      <display-name>HelloIF</display-name>
      <port-component-name>HelloIF</port-component-name>
      <wsdl-port xmlns:wsdl-port_ns__="urn:Foo">wsdl-port_ns__:HelloIFPort</wsdl-port>
      <service-endpoint-interface>helloservice.HelloIF</service-endpoint-interface>
      <service-impl-bean>
        <servlet-link>HelloImpl</servlet-link>
      </service-impl-bean>
    </port-component>
  </webservice-description>
</webservices>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyHelloService" targetNamespace="urn:Foo" xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></output></operation></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyHelloService" targetNamespace="urn:Foo" xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></output></operation></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>
```

Description name:  
MyHelloService

```

<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyHelloService" targetNamespace="urn:Foo" xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></output></operation></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>

```

Port & Binding:  
HelloIF

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyHelloService" targetNamespace="urn:Foo" xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></output></operation></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>
```

Operation name:  
sayHello

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyHelloService" targetNamespace="urn:Foo" xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></output></operation></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>
```

Input/Output  
Messages



```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="MyHelloService" targetNamespace="urn:Foo" xmlns:tns="urn:Foo"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
  <types/>
  <message name="HelloIF_sayHello">
    <part name="String_1" type="xsd:string"/></message>
  <message name="HelloIF_sayHelloResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="HelloIF">
    <operation name="sayHello" parameterOrder="String_1">
      <input message="tns:HelloIF_sayHello"/>
      <output message="tns:HelloIF_sayHelloResponse"/></operation></portType>
  <binding name="HelloIFBinding" type="tns:HelloIF">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"/>
    <operation name="sayHello">
      <soap:operation soapAction=""/>
      <input>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></input>
      <output>
        <soap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
          namespace="urn:Foo"/></output></operation></binding>
  <service name="MyHelloService">
    <port name="HelloIFPort" binding="tns:HelloIFBinding">
      <soap:address location="REPLACE_WITH_ACTUAL_URL"/></port></service></definitions>
```

Parameters of  
sayHello

# Examples of C# Web Service

- Example 1:
  - C# source code.
  - The corresponding WSDL (generated)
  - The corresponding SOAP.
- Example 2:
  - C# source code.
  - The corresponding WSDL (generated)
  - The corresponding SOAP.

# What is UDDI

- UDDI is a platform-independent framework for describing services, discovering businesses, and integrating business services by using the Internet.
- UDDI stands for Universal Description, Discovery and Integration
- UDDI is a directory for storing information about web services
- UDDI is a directory of web service interfaces described by WSDL
- UDDI communicates via SOAP
- UDDI is built into the Microsoft .NET platform

## What is UDDI Based On?

- UDDI uses World Wide Web Consortium (W3C) and Internet Engineering Task Force (IETF) Internet standards such as XML, HTTP, and DNS protocols.
- UDDI uses WSDL to describe interfaces to web services
- Additionally, cross platform programming features are addressed by adopting SOAP, known as XML Protocol messaging specifications found at the W3C Web site.

# UDDI Benefits

- Any industry or businesses of all sizes can benefit from UDDI
- Before UDDI, there was no Internet standard for businesses to reach their customers and partners with information about their products and services. Nor was there a method of how to integrate into each other's systems and processes.
- Problems the UDDI specification can help to solve:
  - Making it possible to discover the right business from the millions currently online
  - Defining how to enable commerce once the preferred business is discovered
  - Reaching new customers and increasing access to current customers
  - Expanding offerings and extending market reach
  - Solving customer-driven need to remove barriers to allow for rapid participation in the global Internet economy
  - Describing services and business processes programmatically in a single, open, and secure environment

## How can UDDI be Used

- If the industry published an UDDI standard for flight rate checking and reservation, airlines could register their services into an UDDI directory.
- Travel agencies could then search the UDDI directory to find the airline's reservation interface.
- When the interface is found, the travel agency can communicate with the service immediately because it uses a well-defined reservation interface.

# Who is Supporting UDDI?

- UDDI is a cross-industry effort driven by all major platform and software providers like
  - Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP, and Sun,
  - as well as a large community of marketplace operators, and e-business leaders.
- Over 220 companies are members of the UDDI community.

# REST (RESTful)

- Representational State Transfer (REST)
- A simpler alternative to SOAP and WSDL based Web services.
  - Widespread acceptance across the Web
  - Especially well known and used as the Internet of Things (IoT).
- Return data in XML and/or JSON format,
  - JSON has been gaining more and more support
    - ▶ the data format of choice for many of the newer REST implementations.



# JSON

- JSON (JavaScript Object Notation)

- an open standard format
  - ▶ uses human-readable text to transmit data objects
  - ▶ consisting of attribute–value pairs.
- Primarily transmit data between a server and web application,
- An alternative to XML.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    }
  ],
  "children": [],
  "spouse": null
}
```

PIVOTAL

autodesk

COMPAQ

TOSHIBA

IBM

Network Programming

Web Services

SAP

accenture

McAfee

UNITED

ORACLE

intel

Peregrine  
SYSTEMS  
Fortitudo Business

Microsoft

FUJITSU

KPMG Consulting

GRAND CENTRAL  
COMMUNICATIONS

DAIMLERCHRYSLER

flamenco  
networks

Ford Motor Company

amcracker

LOUDCLOUD

WS

hp  
invent

IONA

POSCO

epicentric

KANA

bea

CommerceQuest

CAPE CLEAR

epicor

REUTERS

BUSINESS OBJECTS

Rational  
the software development company

corechange

SYBASE

plumtree

REED ELSEVIER

FrontRange  
SOLUTIONS

Borland

COMMERCE ONE

sas

Sabre

FileNET

macromedia

groove NETWORKS

Qwest  
ride the light

Akamai

webMethods

J D EDWARDS

DASSAULT SYSTEMES

RealNames

VeriSign

versata  
Automating e-Business