B2B Application Integration Using Web Services

Problem Statement

- Market Globalization
    - Need to stay ahead of competitors
- Solutions from multiple vendors
    - Unable to share information and isolated functionality

Solution

- Application Integration(AI)
    - Application integration is the real-time controlled sharing of data and business processes among any connected applications and data sources within inter and intra organizations.

Advantages

- Increased productivity
- Controlled procurement processes
- Interoperability with partners
- Reuse of existing systems

Types of AI

- Enterprise Application Integration
  - Integrating applications within an enterprise
  - EDI – based on ANSI standards
  - Point-to-point integration
- B2B Application Integration
  - Integrating applications across enterprises
  - Middlewares – facilitate communication between two or more software systems
  - Point-to-point NOT feasible

Why Web Services?

- Java Middleware Technologies
  - RMI, JMS - Language dependent
- Distributed objects
  - CORBA, DCOM - Platform dependent
- Message Brokers
  - Require sweeping changes in participating applications and hence expensive
- Open standards, platform and language independent, loosely-coupled integration

- HTTP is a protocol used to transfer an HTML page from Web server to Web browser upon a request from the browser using URL.
- Major Components are:
- HTTP request message
- HTTP response message
- HTTP methods
- HTTP status codes

MIME

- Multipurpose Internet Mail Extensions (MIME) is an Internet standard that extends the format of e-mail to support:
- Text in character sets other than ASCII
- Non-text attachments
- Message bodies with multiple parts
- Header information in non-ASCII character sets

MIME

- Internet e-mail transmission protocol, SMTP, supports only 7-bit ASCII characters.
- MIME defines mechanisms for sending other kinds of information in e-mail.
- These include text in languages other than English using character encodings other than ASCII, and 8-bit binary content such as files containing images, sounds, movies, and computer programs.

Introduction

- The format for HTTP request and response message is:
  - Request / Response Line
  - Message Headers
  - <CR><LF> --- Carriage return and Line feed
  - <Message Body>
- For request message
  - User-Agent header is used to indicate the details of the browser.
  - Message body depends on the request method.
  - For GET it is empty and for POST is contains the user-defined values.
  - Format → name1=value1&name2=value2…

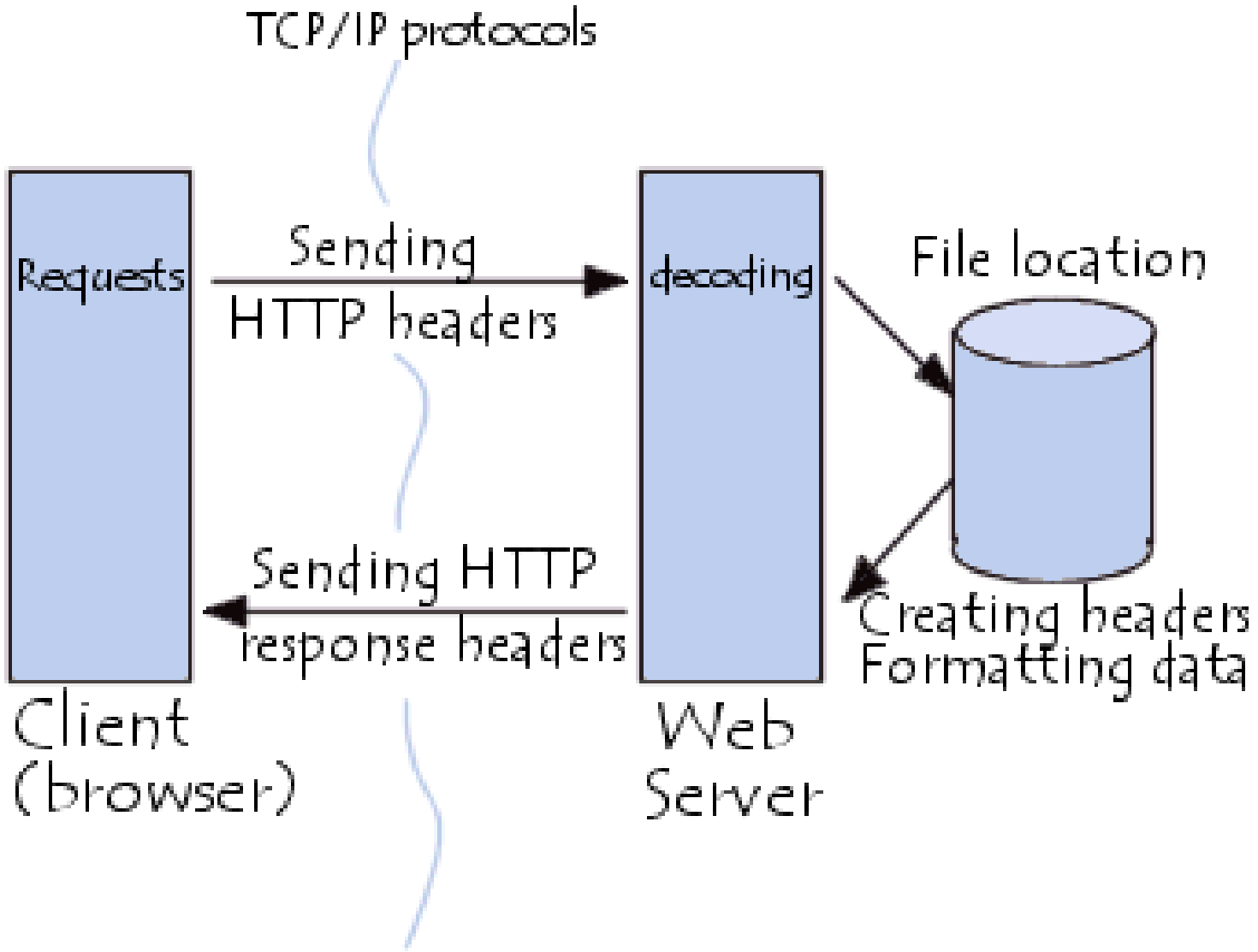
HTTP Request Example

- GET http://en.kioskea.net/ HTTP/1.0
- Accept: text/html
- If-Modified-Since: Saturday, 15-January-2000 14:37:11 GMT
- User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows 95)

HTTP Response Example

- **For response message**
- HTTP/1.0 200 OK
- Date: Sat, 15 Jan 2000 14:37:12 GMT
- Server: Microsoft-IIS/2.0
- Content-Type: text/HTML
- Content-Length: 1245
- Last-Modified: Fri, 14 Jan 2000 08:25:13 GMT

Communication between browser and server



Service-Oriented Architecture (SOA)

- A service-oriented architecture is essentially a collection of services. These services communicate with each other. The communication can involve either simple data passing or it could involve two or more services coordinating some activity. Some means of connecting services to each other is needed.

Or

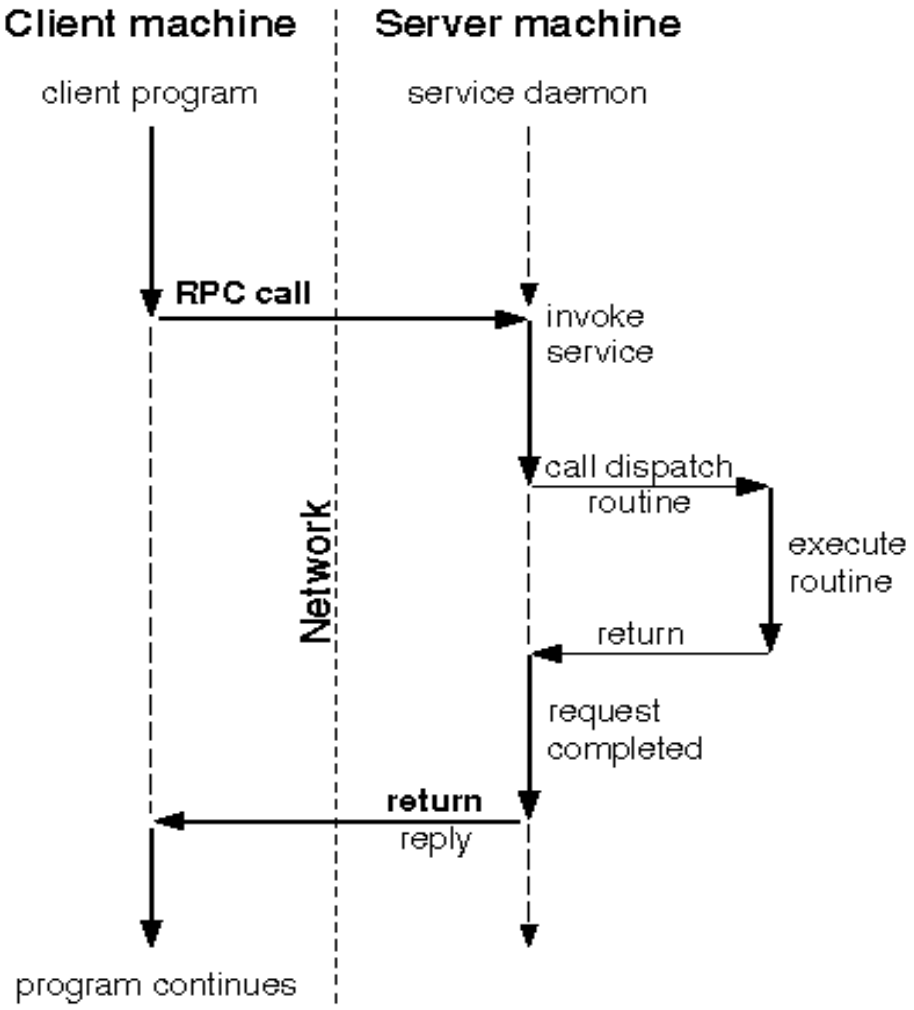
- Service Oriented Architecture or SOA can be explained as a collection of many services in a network which communicate with each other and this involves data exchange with service coordination

SOA

- SOA was based on DCOM or Object Request Brokers (ORBs) but nowadays SOA is based predominantly on Web Services.
- DCOM:DCOM is the acronym for the Distributed Component Object Model, an extension of the Component Object Model (COM).
- DCOM was designed for use across multiple network transports, including Internet protocols such as HTTP
- CORBA: CORBA is the acronym for Common Object Request Broker Architecture. It was developed under the auspices of the Object Management Group (OMG).

Remote Procedure Call (RPC)

- By using RPC, programs on networked platforms can communicate with remote (and local) resources.
- RPC allows network applications to use specialized kinds of procedure calls designed to hide the details of underlying networking mechanisms.
- The complexity involved in the development of distributed processing is reduced by keeping the semantics of a remote call the same whether or not the client and server are collocated on the same system.



DCOM

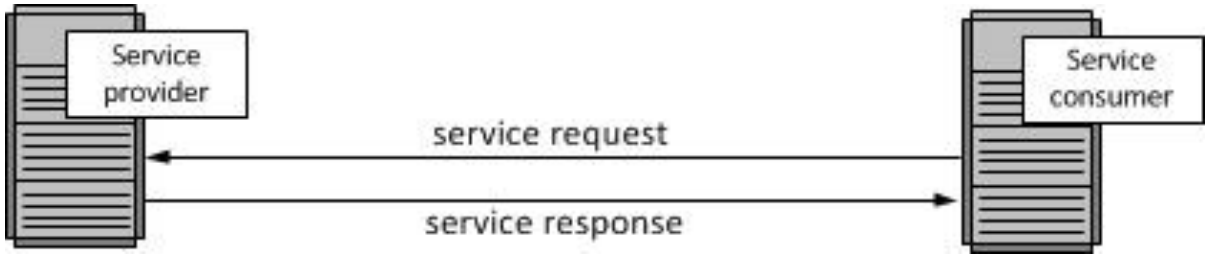
- DCOM (Distributed Component Object Model) is a set of Microsoft concepts and program interfaces in which client program objects can request services from server program objects on other computers in a network.
- DCOM is based on the Component Object Model (COM), which provides a set of interfaces allowing clients and servers to communicate within the same computer.
- For example, you can create a page for a Web site that contains a script or program that can be processed (before being sent to a requesting user) not on the Web site server but on another, more specialized server in the network.
- Using DCOM interfaces, the Web server site program (now acting as a client object) can forward a Remote Procedure Call (RPC) to the specialized server object, which provides the necessary processing and returns the result to the Web server site. It passes the result on to the Web page viewer.

CORBA

- **CORBA (Common Object Request Broker Architecture)** is a standard that enables an object written in one programming language, running on one platform to interact with objects across the network that are written in other programming languages and running on other platforms.
- For example, a **client object written in C++** and running under Windows can communicate with an **object on a remote machine written in Java running under UNIX**.

- A service is nothing but a function or some processing logic or business processing that is well-defined, self-contained, and does not depend on the context or state of other services.
- Example of Services are Loan Processing Services, booking a flight ticket online etc. which can be self-contained unit for process the Loan Applications or could be Weather Services, which can be used to get the weather information. Any application on the network can use the service of the Weather Service to get the weather information and SOAs build applications out of such services.
- Instead of services embedding calls to each other in their source code, there are protocols defined which describe how one or more services can talk to each other. This architecture then relies on a business process expert to link and sequence services, in a process known as orchestration, to meet a new or existing business system requirement.

- Connection
- It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider.
- A service provider can also be a service consumer.



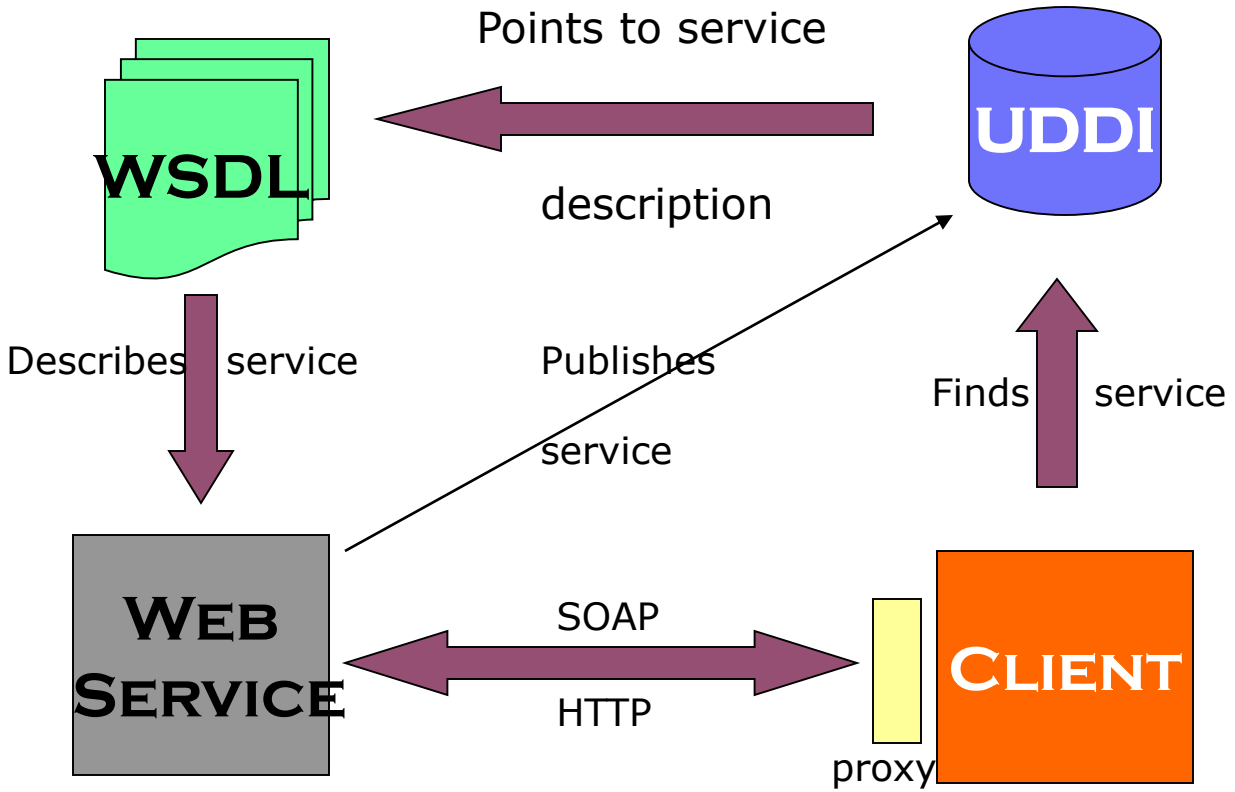
Web Services

- Web services are middleware components that implement business logic via services and expose these services programmatically over the web, which could be invoked by service clients using SOAP over HTTP
- Based on open standards like UDDI, WSDL, SOAP, XML, HTTP
- Separation of specification from implementation

Web Services Stack

- Transport protocol – HTTP, SMTP
- Data encoding – XML
- Standard Message Structure – SOAP
- Service Description – WSDL
- Service Discovery - UDDI

Web Services Framework



Web Services

- Web services are open standard (XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data.
- Web Services can convert your existing applications into Web-applications.
- definitions to Web Services.
- A web service is any piece of software that makes itself available over the internet and uses a standardized XML messaging system. XML is used to encode all communications to a web service. For example, a client invokes a web service by sending an XML message, then waits for a corresponding XML response. As all communication is in XML, web services are not tied to any one operating system or programming language--Java can talk with Perl; Windows applications can talk with Unix applications

XML !!!

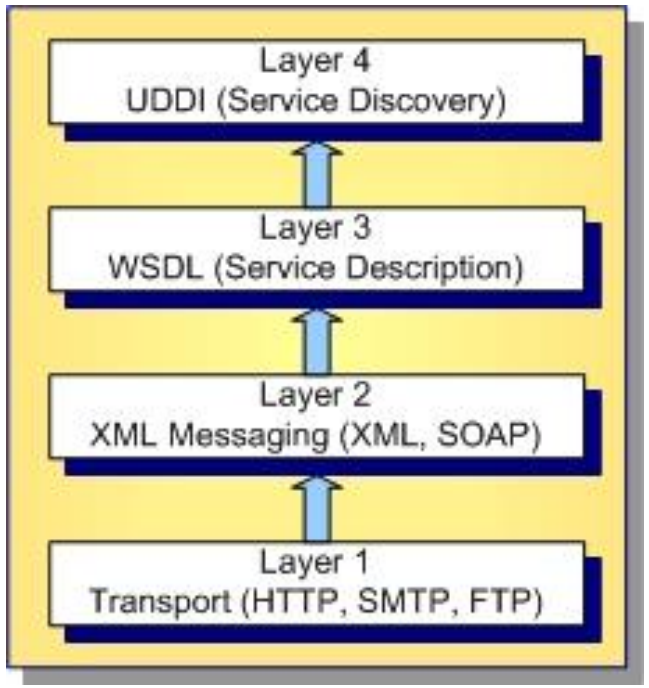
- Xml stands for extensible markup language.
- Its standards are defined by W3C.
- It is flexible text format to create structured data.
- It is design to transport and stored the data.

- Example

A complete web service is, therefore, any service that:

- Is available over the Internet or private (intranet) networks
- Uses a standardized XML messaging system
- Is not tied to any one operating system or programming language
- Is self-describing via a common XML grammar
- Is discoverable via a simple find mechanism

Web Service Protocol Stack



- Service Transport
- This layer is responsible for transporting messages between applications. Currently, this layer includes Hyper Text Transport Protocol (HTTP), Simple Mail Transfer Protocol (SMTP), File Transfer Protocol (FTP)

- XML Messaging
- This layer is responsible for encoding messages in a common XML format so that messages can be understood at either end. Currently, this layer includes XML-RPC and SOAP.

- Service Description
- This layer is responsible for describing the public interface to a specific web service. Currently, service description is handled via the Web Service Description Language (WSDL).

- Service Discovery
- This layer is responsible for centralizing services into a common registry and providing easy publish/find functionality. Currently, service discovery is handled via Universal Description, Discovery, and Integration (UDDI).

- Components of Web Services
- The basic web services platform is XML + HTTP. All the standard web services work using the following components
- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)
- Task
- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

Example

- account-management and order processing system
- accounting personnel use a client application built with Visual Basic
- The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

Benefits of using Web Services:

- Exposing the Existing Function on the network: Web services allows you to expose the functionality of your existing code over the network
- Interoperability :Web services allow various applications to talk to each other and share data and services among themselves.
- Standardized Protocol: This standardization of protocol stack gives the business many advantages such as a wide range of choices, reduction in the cost due to competition, and increase in the quality.
- Low Cost of Communication :Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services

Web services special behavioral characteristics

- XML-Based: XML eliminates any networking, operating system, or platform binding.
- Loosely Coupled
- Ability to be Synchronous or Asynchronous
- Supports Document Exchange

SOAP

The Problem

Interoperability between Internet applications

- Today there are countless different operating systems, different firewalls, different methods of making remote procedure calls, and different platforms. In order to interoperate across the Internet both the client and server need to understand each others security types and trusts, service deployment schemas, and implementation details.
- Remote objects can give a program lots of power over the Internet, but most firewalls block non-HTTP requests.

The Solution

- SOAP bridges the gap between competing object RPC (Remote Procedure Calls) technologies and provides a light-weight messaging format that works with any operating system, any programming language, and any platform.
- SOAP is able to provide intra-process communication across machines.

- Simple Object Access Protocol, a lightweight, message-based protocol built on XML and standard Internet protocols, such as HTTP and SMTP for information exchange in a decentralized environment

- Defines specification for message structure and data encoding

- Facilitates structured and typed messages to be exchanged

- SOAP message must contain a SOAP envelope, a SOAP body and optional SOAP header
- Encoding - serialization of data inside a SOAP message
- SOAP encoding is based on XML Schemas and relies on the XML Schema data types, namespace and the type attribute

SOAP envelope

Container to hold the SOAP message

```
<soap:envelope>
   <soap:header>
    ……….
   </soap:header>
   <soap:body>
     <add>
       <num1 xsi:type="xsd:int">5</num1>
       <num2 xsi:type="xsd:int">10</num2>
     </add>
   </soap:body>
</soap:envelope>
```

SOAP header

Sequence numbers, authentication credentials

SOAP payload

Method calls, parameters, Application-specific data

WSDL

- Describes Web Service methods to heterogeneous clients in a platform and language independent manner
- SOAP toolkits generate proxy classes using WSDL
- Service contract which specifies the methods available and type information needed to properly compose SOAP requests

```
<?xml version="1.0" encoding="utf8" ?>
<definitions >
        <types />
        <message name="addSoapIn">
            <part name="num1" type="s:int" />
            <part name="num2" type="s:int" />
        </message>
        <message name="addSoapOut">
            <part name="addResult" type="s:int" />
        </message>
        <portType name="sampleSoap">
            <operation name="add">
               <input message="tns:addSoapIn" />
               <output message="tns:addSoapOut" />
            </operation>
        </portType>
        <binding name="sampleSoap" type="tns:sampleSoap">
             <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc" />
            <operation name="add">
            <soap:operation soapAction="http://tempuri.org/add" style="rpc" />
             <input>
        <soap:body use="encoded" namespace="http://tempuri.org/" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </input>
        <output>
        <soap:body use="encoded" namespace="http://tempuri.org/" encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
        </output>
        </operation>
        </binding>
        <service name="Sample">
            <port name="sampleSoap" binding="tns:sampleSoap">
            <soap:address location="http://agena.cis.ksu.edu:8080/axis/services/Sample" />
            </port>
        </service>
        </definitions>
```

Describes custom or complex Data types

public int add(int num1, int num2)

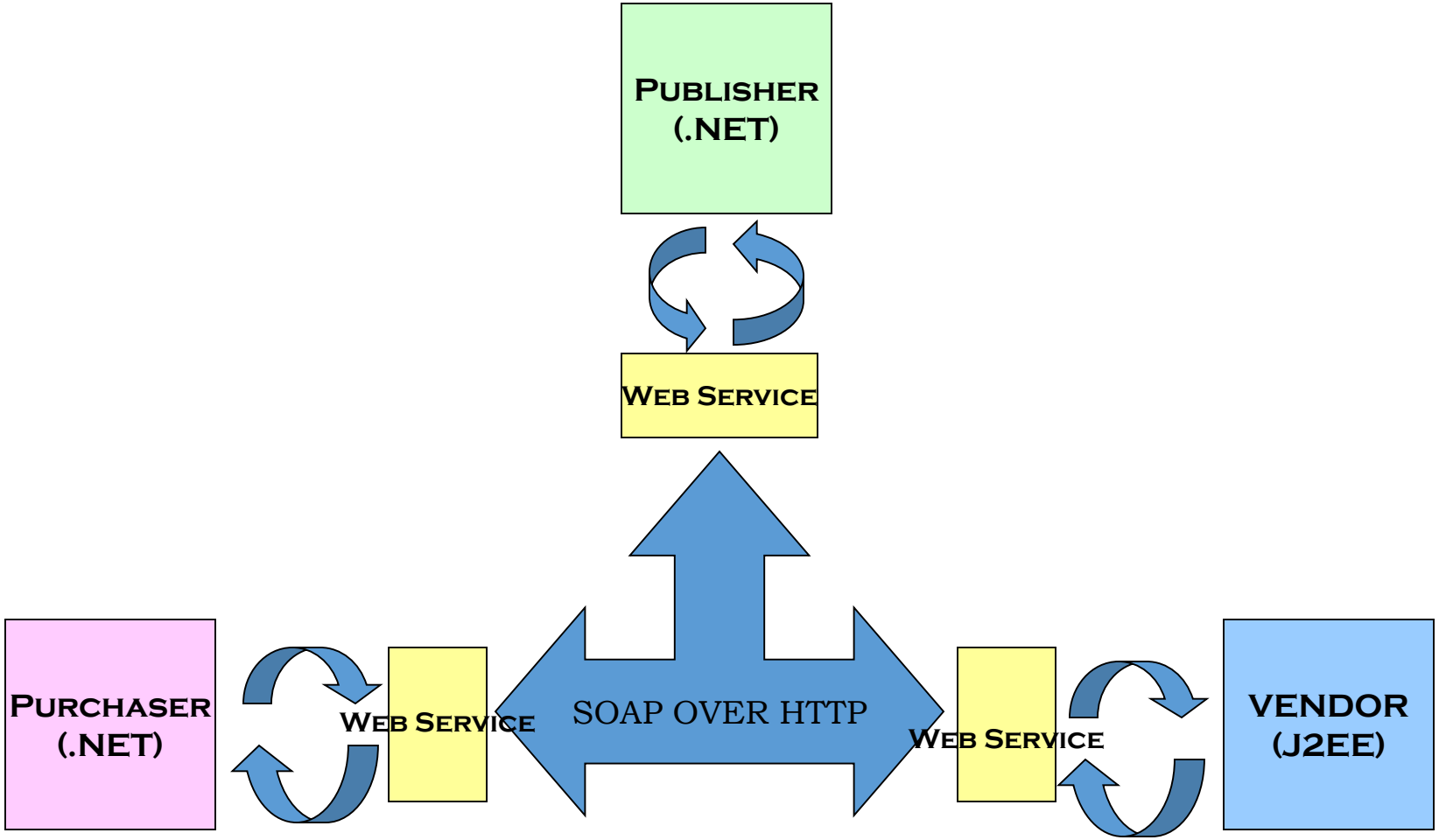
Number and type of input and output parameters

Methods available along with input and output messages

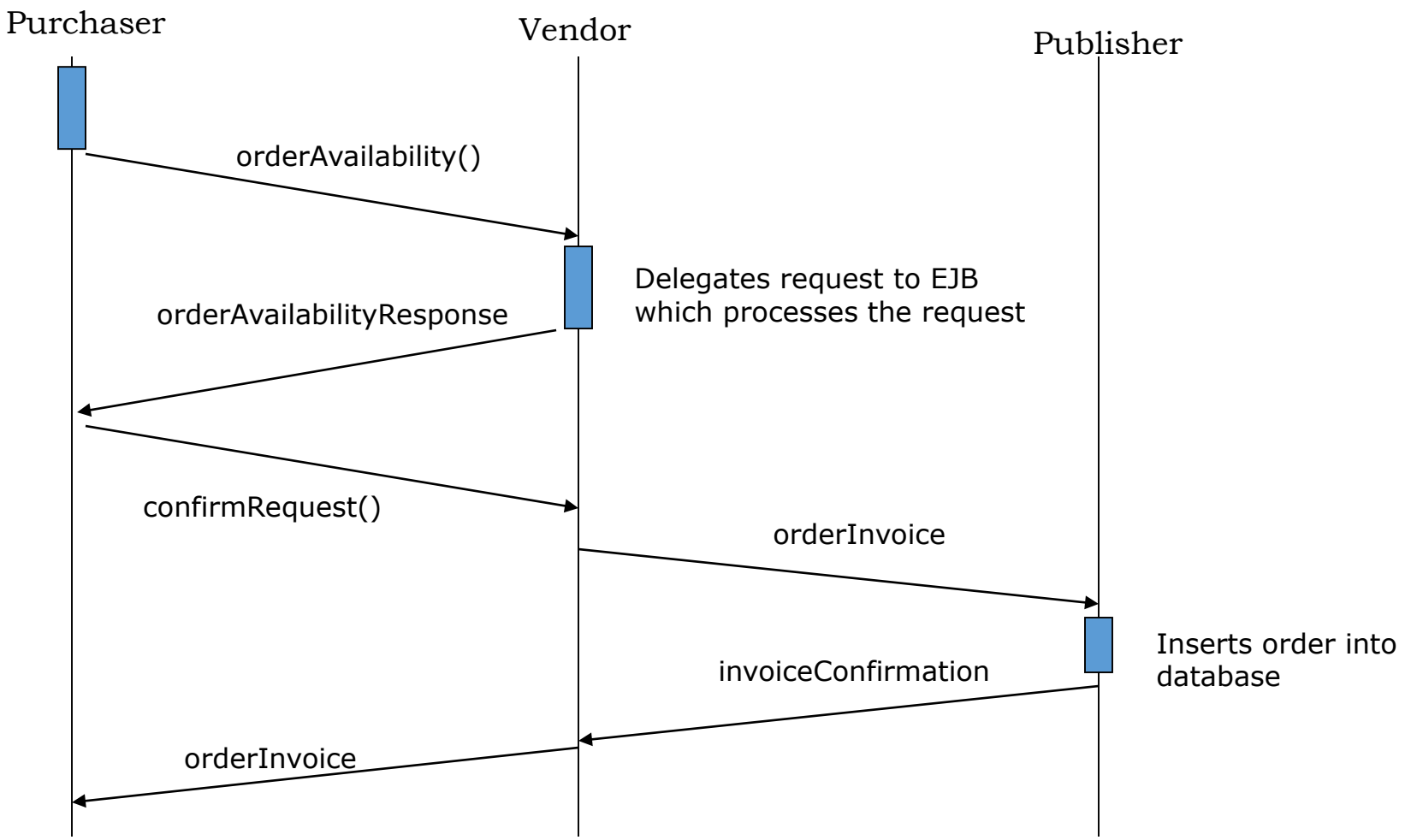
Describes style and encoding of the SOAP messages for each operation

Describes entry points to web service HTTP-POST, HTTP-GET, SOAP

System Architecture



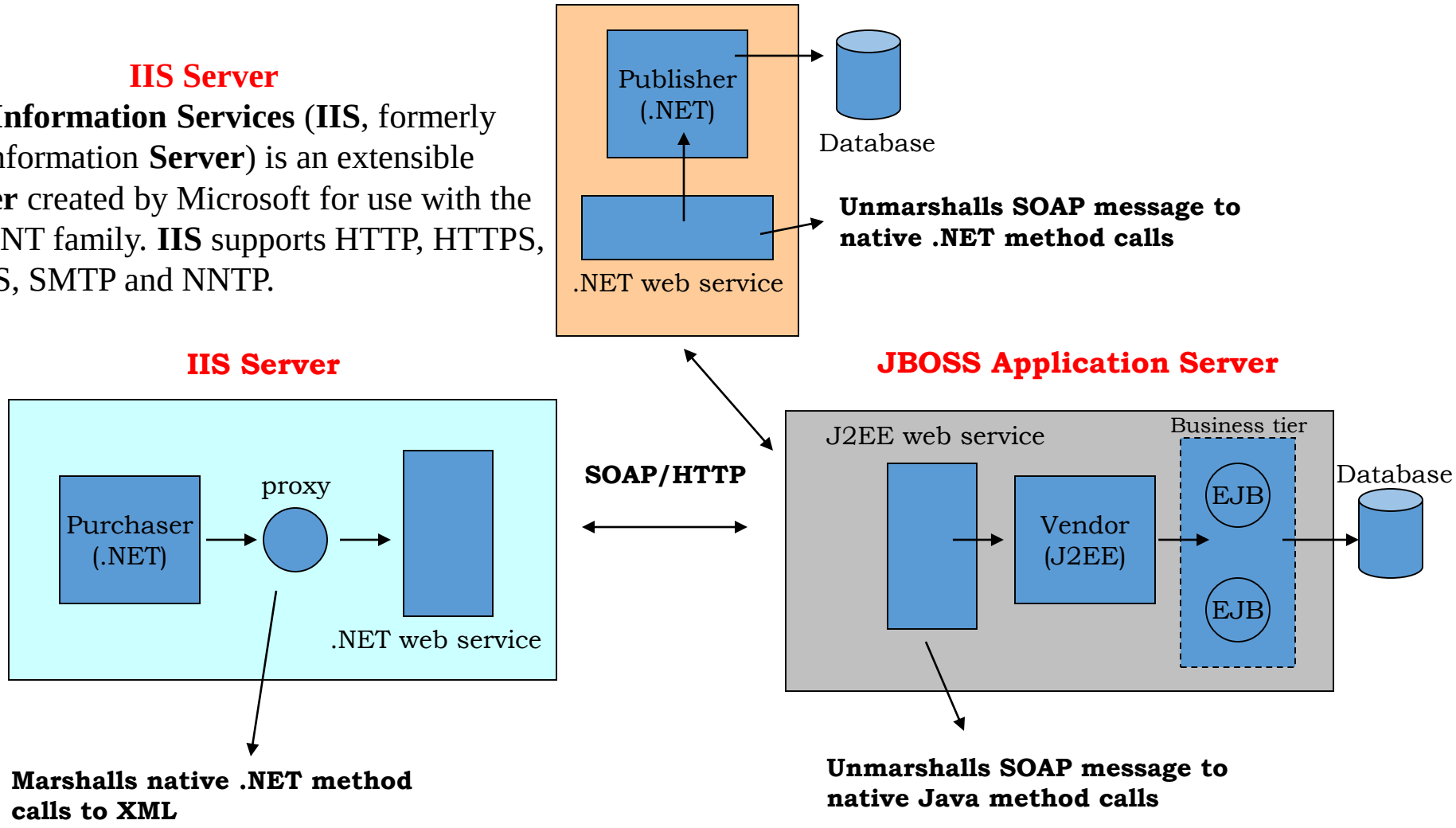
Message Flow



Detailed System Architecture

IIS Server

Internet Information Services (**IIS**, formerly Internet Information **Server**) is an extensible web **server** created by Microsoft for use with the Windows NT family. **IIS** supports HTTP, HTTPS, FTP, FTPS, SMTP and NNTP.

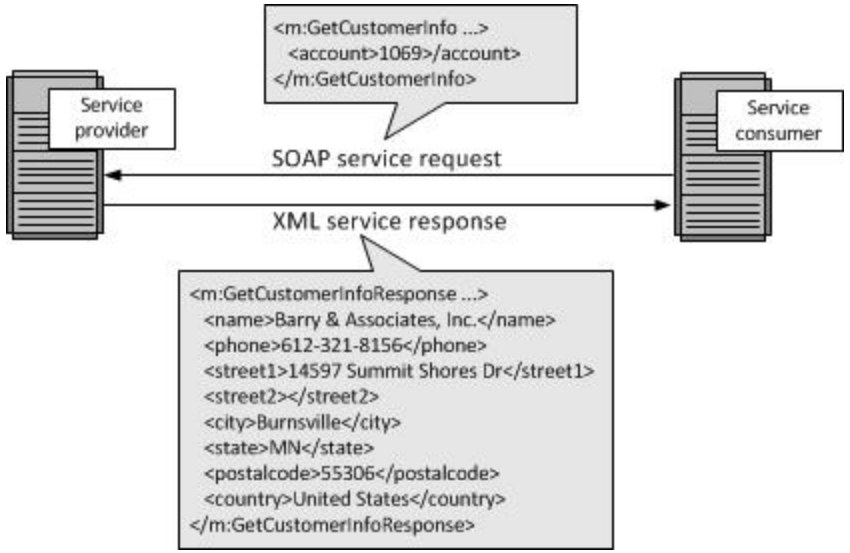


Web Services

- SOAP
- SOAP is a lightweight protocol intended for exchanging structured information in a decentralized, distributed environment.
- SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols.
- The framework has been designed to be independent of any particular programming model and other implementation-specific semantics.

- SOAP was originally part of the specification that included the Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI). It is used now without WSDL and UDDI.

- SOAP provides the envelope for sending Web Services messages over the Internet/Internet. It is part of the set of standards specified by the W3C. SOAP is an alternative to Representational State Transfer (REST) and JavaScript Object Notation (JSON).



3 Main Components

- The SOAP envelope construct defines an overall framework for expressing **what** is in a message; **who** should deal with it, and **whether** it is optional or mandatory.
- The SOAP encoding rules defines a serialization mechanism that can be used to exchange instances of application-defined data types.
- The SOAP RPC representation defines a convention that can be used to represent remote procedure calls and responses.

SOAP Skeleton

```
<?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>
```

- **Envelope –** Defines the start and the end of the message. It is a mandatory element.
- **Header –** An optional header providing information on authentication, encoding of data, or how a recipient of a SOAP message should process the message.
- **Body –** Contains the XML data comprising the message being sent. It is a mandatory element.
- **Fault –** An optional Fault element that provides information about errors that occur while processing the message.
- All these elements are declared in the default namespace for the SOAP envelope −

- The required SOAP Envelope element is the root element of a SOAP message.
- It defines the XML document as a SOAP message.
- Note the use of the xmlns:soap namespace. It should always have the value of:
- http://www.w3.org/2001/12/soap-envelope and it defines the Envelope as a SOAP Envelope:

- The optional SOAP Header element contains application specific information (like authentication, payment, etc) about the SOAP message.
- If the Header element is present, it must be the first child element of the Envelope element.
- **Note:** All immediate child elements of the Header element must be namespace-qualified.

SOAP Skeleton

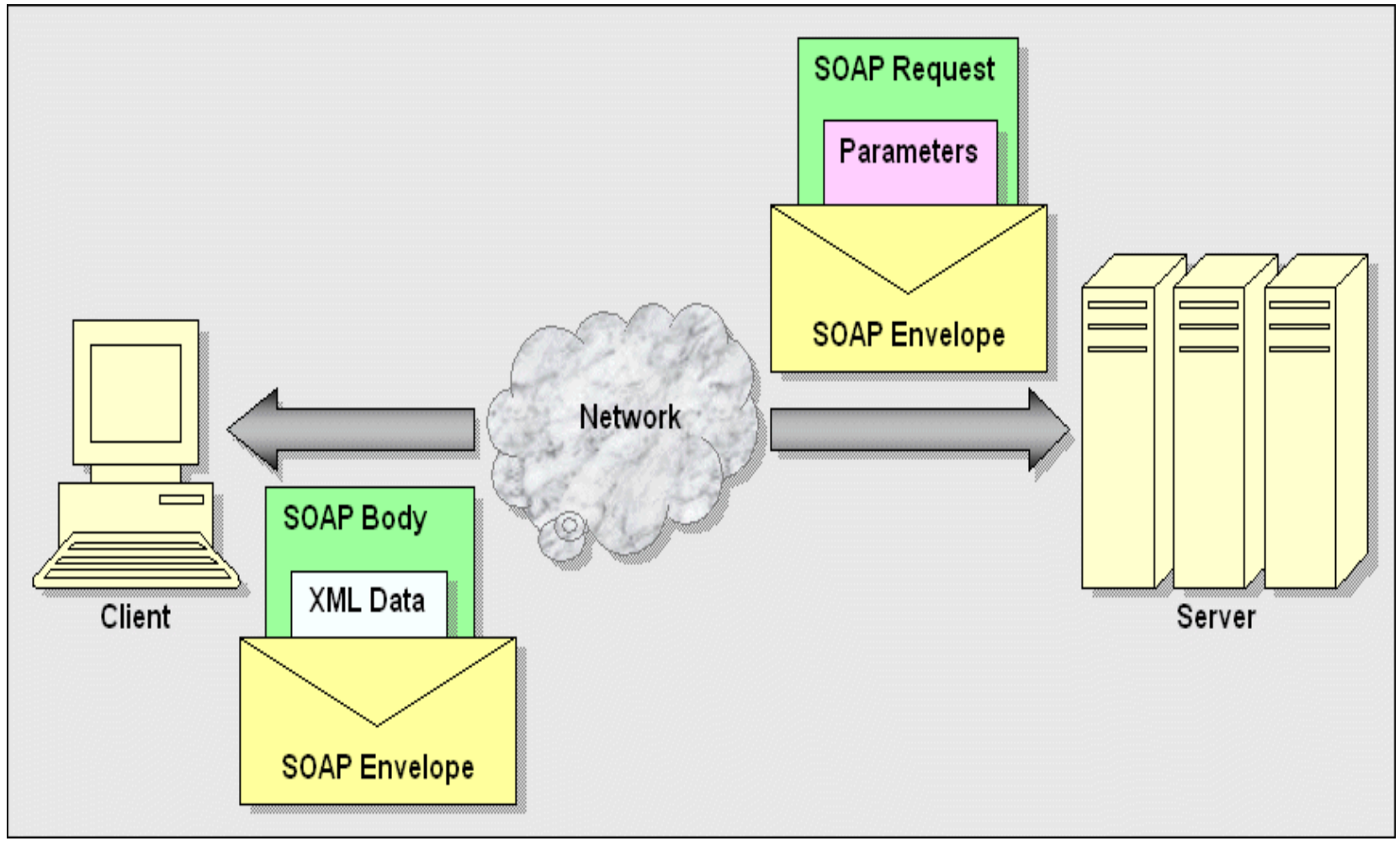
Simple Example

MESSAGE

```
<?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>
    <n:movie>Star Wars</n:movie>
</soap:Header>

<soap:Body>
    <p:day>Wednesday</p:day>
    <p:times>7 p.m-10 p.m.</p:times>
</soap:Body>
</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:Header>
    <n:movie>Star Wars</n:movie>
</soap:Header>

<soap:Body>
    <p:day>Wednesday</p:day>
    <p:times>9 p.m.</p:times>
</soap:Body>

</soap:Envelope>
```


SOAP Request
Parameters
SOAP Envelope
Network
Client
SOAP Body
XML Data
SOAP Envelope
Server

WSDL : **Service Contract**

- WSDL stands for *Web Services Description Language*.
- It describes the "what", "how" and "where" parts of web services.
- The **what** part of WSDL deals with the interface of a web service. The interface describes the supported operations and the input/output parameters accepted by them.
- The interface part essentially tells the service
- consumer what services are available.
- WSDL defines individual services as "ports types".

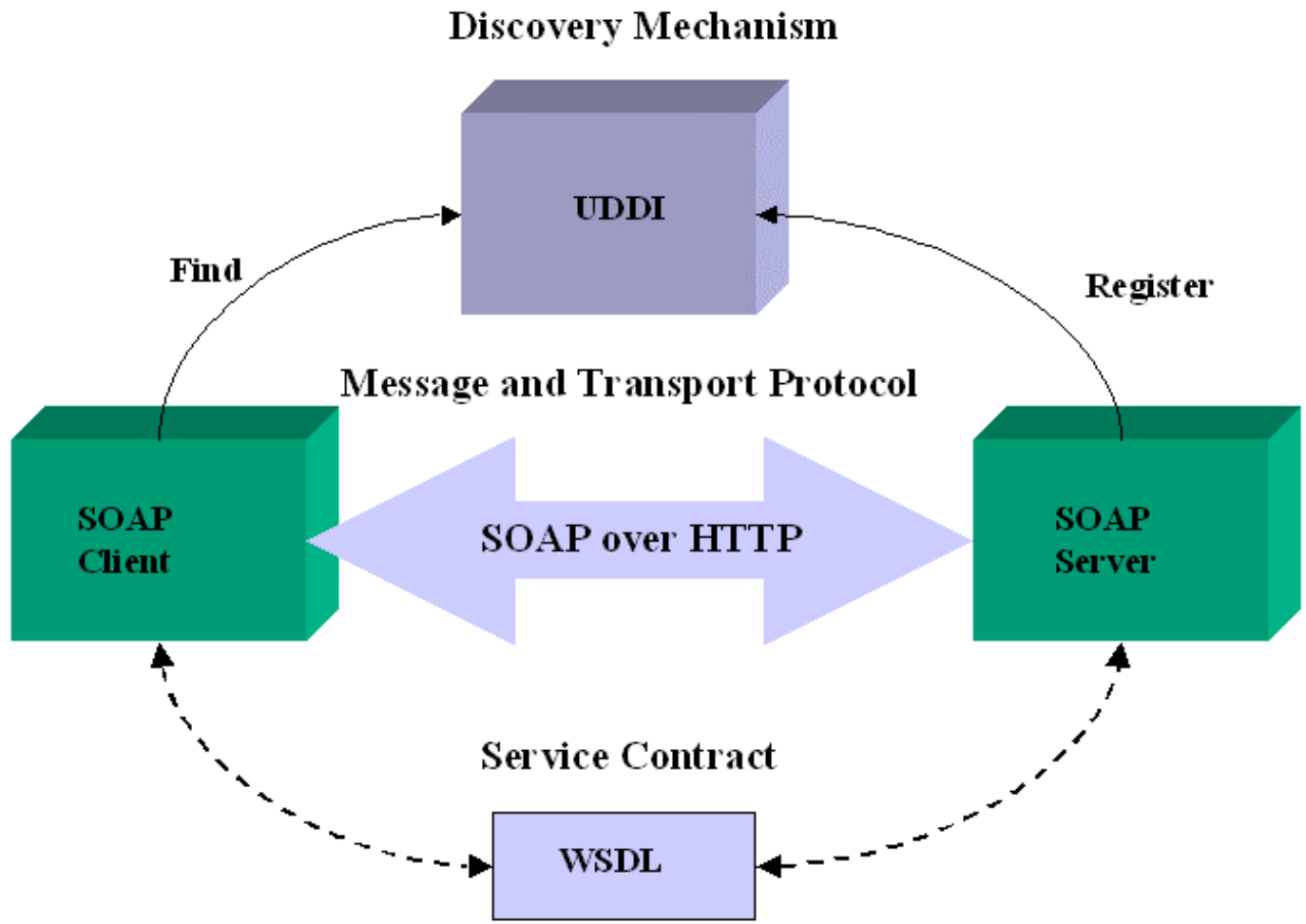
- The **how** part of WSDL deals with how a client is supposed to communicate with the server.
- It indicates what kind of transport is used to send messages, what style of messages is to be sent (RPC or document) and how the messages are encoded on the wire.
- WSDL defines this information in "binding".
- The **where** part of the WSDL describes where the service resides. For SOAP over HTTP implementations, this information could be provided as the service URL.
- WSDL defines service location information in "port".

UDDI

- UDDI stands for ***Universal Description, Discovery and Integration***.
- Advertising services in a UDDI server makes it easy for partners to find both business and technical information about them.
- UDDI clients are able to discover and reuse services instead of building services from scratch.

- For Web services to be meaningful, there is a need to provide information about them beyond the technical specifications of the service itself.
- Central to UDDI’s purpose is **the representation of data and metadata about Web services**.
- The other principle behind UDDI design is **"Data is worthless if it is lost within a mass of other data and cannot be distinguished or discovered".**
- Based on this principle, UDDI allows definition of any number of classification schemes (taxonomies), allows such classification schemes to be used on every component in the information model.

Service Oriented Architecture using Web Services



Web Feeds

- Web feed is a document often based on XML, which provides content such as news, items, weather and blogs.
- Real-time Simple Syndication
- Rich Site Summary

1. RSS simply allows a website to automatically publish a basic file of “what’s new”. A user (i.e. you) can subscribe to this file (called a feed) and be notified of anything new that has been written on that particular website since you last received an update. It’s a quick and easy way of finding what is new on a site without necessarily having to visit the site yourself. Using a feed aggregator (more on these in a second), you can quickly keep up-to-date on the latest updates to many websites all in one place.
2. Let’s have a quick look at Life Goggles. When visiting this site, you can see the large orange RSS icon at the top right of the page.

Summary

- **S**imple **O**bject **A**ccess **P**rotocol
- Communication Protocol
- Communication between applications
- Format for sending messages
- Designed to communicate via Internet
- Platform independent
- Language independent
- Based on XML
- Simple and Extensible
- Get around Firewalls
- W3C standard

A SOAP request:

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

  <?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: nnn

  ```
  <?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>
  ```

JSON

JavaScript Object Notation

- JSON or JavaScript Object Notation is a lightweight text-based open standard designed for human-readable data interchange.

Uses of JSON :

- It is used while writing JavaScript based applications that includes browser extensions and websites.
- JSON format is used for serializing and transmitting structured data over network connection.
- It is primarily used to transmit data between a server and web applications.
- Web services and APIs use JSON format to provide public data.
- It can be used with modern programming languages.

Characteristics of JSON:

- JSON is easy to read and write.
- It is a lightweight text-based interchange format.
- JSON is language independent.

Basic syntax of JSON

- Data is represented in name/value pairs.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).
- Square brackets hold arrays and values are separated by ,(comma).

```
{
   "book": [

      {
         "id":"01",
         "language": "Java",
         "edition": "third",
         "author": "Herbert Schildt"
      },

      {
         "id":"07",
         "language": "C++",
         "edition": "second"
         "author": "E.Balagurusamy"
      }

   ]
}
```

Creating Simple Objects

- JSON objects can be created with JavaScript. Let us see the various ways of creating JSON objects using JavaScript −
- Creation of a new Object
- var JSONObj = new Object();
- Lets create object for bookname and price.

- JSON and XML are human readable formats and are language independent.
- Compairisions
- Verbose:XML is more verbose than JSON, so it is faster to write JSON for programmers.
- Arrays Usage :XML is used to describe the structured data, which cJSON s whereas JSON XML

```
{
   "company": Volkswagen,
   "name": "Vento",
   "price": 800000
}
```

```
<car>
   <company>Volkswagen</company>
   <name>Vento</name>
   <price>800000</price>
</car>
```

JSON with PHP

how to encode and decode JSON objects using PHP programming language.

JSON Functions

Function	Libraries
json_encode	Returns the JSON representation of a value.
json_decode	Decodes a JSON string.
json_last_error	Returns the last error occurred.

Encoding JSON in PHP (json_encode)

- PHP json_encode() function is used for encoding JSON in PHP.
- This function returns the JSON representation of a value on success or FALSE on failure.
- Syntax
- string json_encode ($value [, $options = 0])

Decoding JSON in PHP (json_decode)

- PHP json_decode() function is used for decoding JSON in PHP
- syntax
- mixed json_decode ($json [,$assoc = false])
- Paramaters
- **json_string** − It is an encoded string which must be UTF-8 encoded data.
- **assoc** − It is a boolean type parameter, when set to TRUE, returned objects will be converted into associative arrays.

Representational State Transfer (REST)

- REST stands for **RE**presentational **S**tate **T**ransfer. REST is web standards based architecture and uses HTTP Protocol for data communication.
- every component is a resource and a resource is accessed by a common interface using HTTP standard methods.
- In REST architecture, a REST Server simply provides access to resources and REST client accesses and presents the resources.
- **Here each resource is identified by URIs/ global IDs**
- REST uses various representations to represent a resource like text, JSON and XML.
- JSON is the most popular format being used in web services.

- HTTP Methods
- Following well known HTTP methods are commonly used in REST based architecture.

- **REST uses the available HTTP methods as a resource interface:**
- **Create (C) → HTTP POST**
- **Read (R) → HTTP GET**
- **Update (U) →HTTP PUT**
- **Delete (D) → HTTP DELETE**

- Web services based on REST Architecture are known as RESTful web services
- What is a Resource?
- REST architecture treats every content as a resource. These resources can be text files, html pages, images, videos or dynamic business data
- Here each resource is identified by URIs/ global IDs.

- REST, unlike SOAP, is not a WS (web service) standard but an architectural style for web applications.
- REST is not a standard or protocol, REST is an architectural style.
- REST makes use of existing web standards (HTTP, URL, XML, JSON, MIME types).
- REST is resource oriented. Resources (pieces of information) are addressed by URIs and passed from server to client (or the other way round).

Example

- What is Representational State Transfer?
- **To understand the REST principle, look at what happens in a web access of a browser .**
- REST is based on existing web (WWW, HTTP) principles and protocols: Resources: Application state and functionality are abstracted into resources (everything is a resource).
- Addressability of resources: Every resource is uniquely addressable using hyperlinks.
- Uniform interface for accessing resources: All resources share a uniform interface for the transfer of state between client and resource, consisting of - a constrained (=limited) set of well-defined operations (GET, PUT, POST, DELETE). - a constrained set of content types (text/html, text/xml etc.).

- Architectural constraints for REST
- 1. Client-server paradigm: A client retrieves resources from a server or updates resources on a server.
  - Separation of concerns such as presentation (client) from data storage (server).
  - Portability (UI may be ported to different platforms).
- 2. Stateless: A client request contains all information necessary for the server to understand the request.
  - No need for storing context (state) on the server.
  - Better scalability.
- 3. Cacheable: Data (resources) need to be labeled as cacheable or non-cacheable.
  - Improve network performance.

REST protocol’

- REST is not a protocol like SOAP. But REST defines some core characteristics that make a system REST-ful.
- REST does not define something new, it simply makes use of existing protocols and standards (HTTP, URI).
- Addressing resources: REST uses plain *URIs* (actually URLs) to address and name resources.
- Access to resources: REST uses the available HTTP methods as a resource interface:
- **Create (C) → HTTP POST**
- **Read (R) → HTTP GET**
- **Update (U) →HTTP PUT**
- **Delete (D) → HTTP DELETE**

- Resource representations:
- REST uses standard resource representations like HTML, XML, JSON, GIF, JPEG. Commonly used representations are XML and JSON (preferable to XML if the data needs to be transferred in a more compact and readable form).
- Media types:
- REST uses the HTTP header *Content-type* (MIME types like text/html, text/plain, text/xml, text/javascript for JSON etc.) to indicate the encoding of the resource.
- State:
- Application state is to be maintained on the client. The server does not have to maintain a state variable for each client (this improves scalability).
- Resource state (resource creation, update, deletion), however, is maintained on the server.

RESTful in JAVA

- **Jersey** RESTful **Web Services** framework is open source, production quality, framework for developing RESTful **Web Services** in Java that provides support for JAX-RS APIs and serves as a JAX-RS (JSR 311 & JSR 339) Reference Implementation. **Jersey** framework is more than the JAX-RS Reference Implementation.

#	SOAP	REST
1	A XML-based message protocol	An architectural style protocol
2	Uses WSDL for communication between consumer and provider	Uses XML or JSON to send and receive data
3	Invokes services by calling RPC method	Simply calls services via URL path
4	Does not return human readable result	Result is readable which is just plain XML or JSON
5	Transfer is over HTTP. Also uses other protocols such as SMTP, FTP, etc.	Transfer is over HTTP only
6	JavaScript can call SOAP, but it is difficult to implement	Easy to call from JavaScript
7	Performance is not great compared to REST	Performance is much better compared to SOAP - less CPU intensive, leaner code etc.