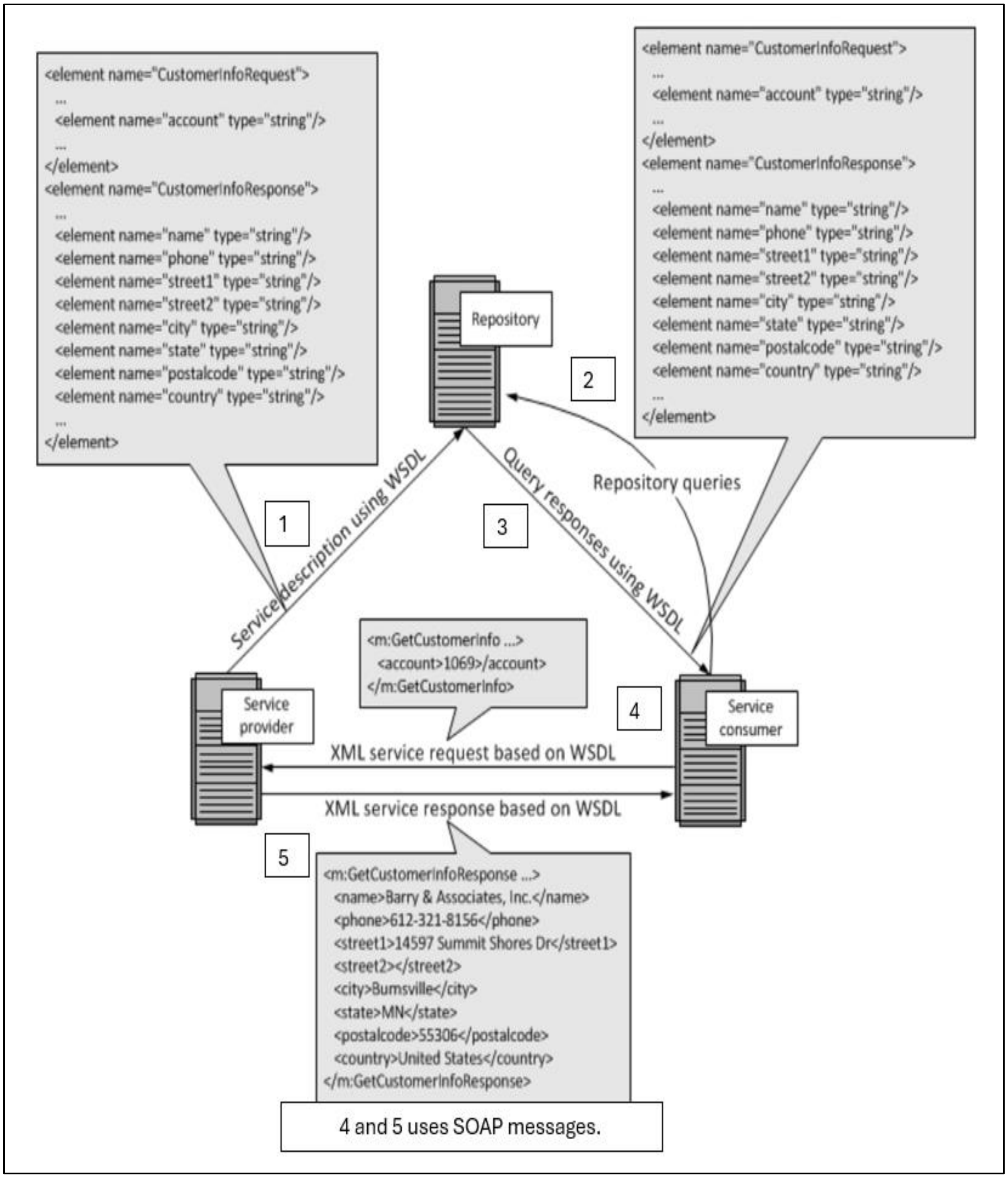

Repository
Service provider
Service consumer
1
2
3
4
5
Service description using WSDL
Repository queries
Query responses using WSDL
XML service request based on WSDL
XML service response based on WSDL
4 and 5 uses SOAP messages.

```
<element name="CustomerInfoRequest">
  ...
  <element name="account" type="string"/>
  ...
</element>
<element name="CustomerInfoResponse">
  ...
  <element name="name" type="string"/>
  <element name="phone" type="string"/>
  <element name="street1" type="string"/>
  <element name="street2" type="string"/>
  <element name="city" type="string"/>
  <element name="state" type="string"/>
  <element name="postalcode" type="string"/>
  <element name="country" type="string"/>
  ...
</element>
```

```
<element name="CustomerInfoRequest">
  ...
  <element name="account" type="string"/>
  ...
</element>
<element name="CustomerInfoResponse">
  ...
  <element name="name" type="string"/>
  <element name="phone" type="string"/>
  <element name="street1" type="string"/>
  <element name="street2" type="string"/>
  <element name="city" type="string"/>
  <element name="state" type="string"/>
  <element name="postalcode" type="string"/>
  <element name="country" type="string"/>
  ...
</element>
```

```
<m:GetCustomerInfo ...>
  <account>1069>/account>
</m:GetCustomerInfo>
```

```
<m:GetCustomerInfoResponse ...>
  <name>Barry & Associates, Inc.</name>
  <phone>612-321-8156</phone>
  <street1>14597 Summit Shores Dr</street1>
  <street2></street2>
  <city>Burnsville</city>
  <state>MN</state>
  <postalcode>55306</postalcode>
  <country>United States</country>
</m:GetCustomerInfoResponse>
```

A WEB SERVICE SCENARIO

What are 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.

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)

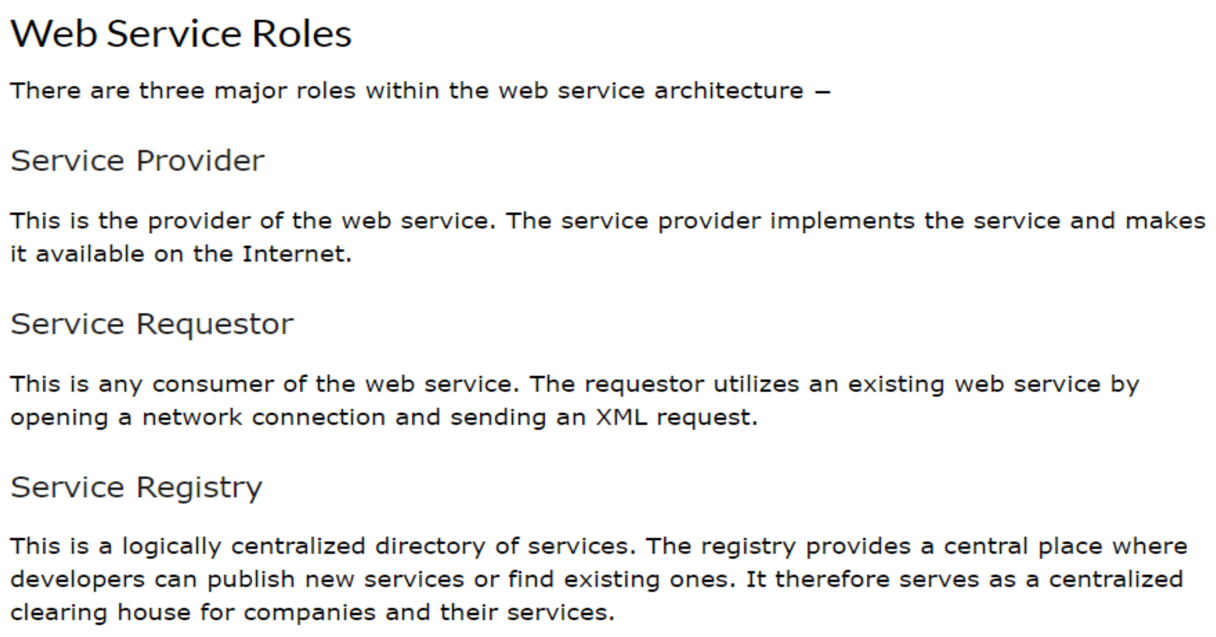
How Does a Web Service Work?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP. A web service takes the help of –

- XML to tag the data
- SOAP to transfer a message
- WSDL to describe the availability of service.

The first option for handling a web service:

Steps (4 and 5)



Web Service Roles

There are three major roles within the web service architecture –

Service Provider

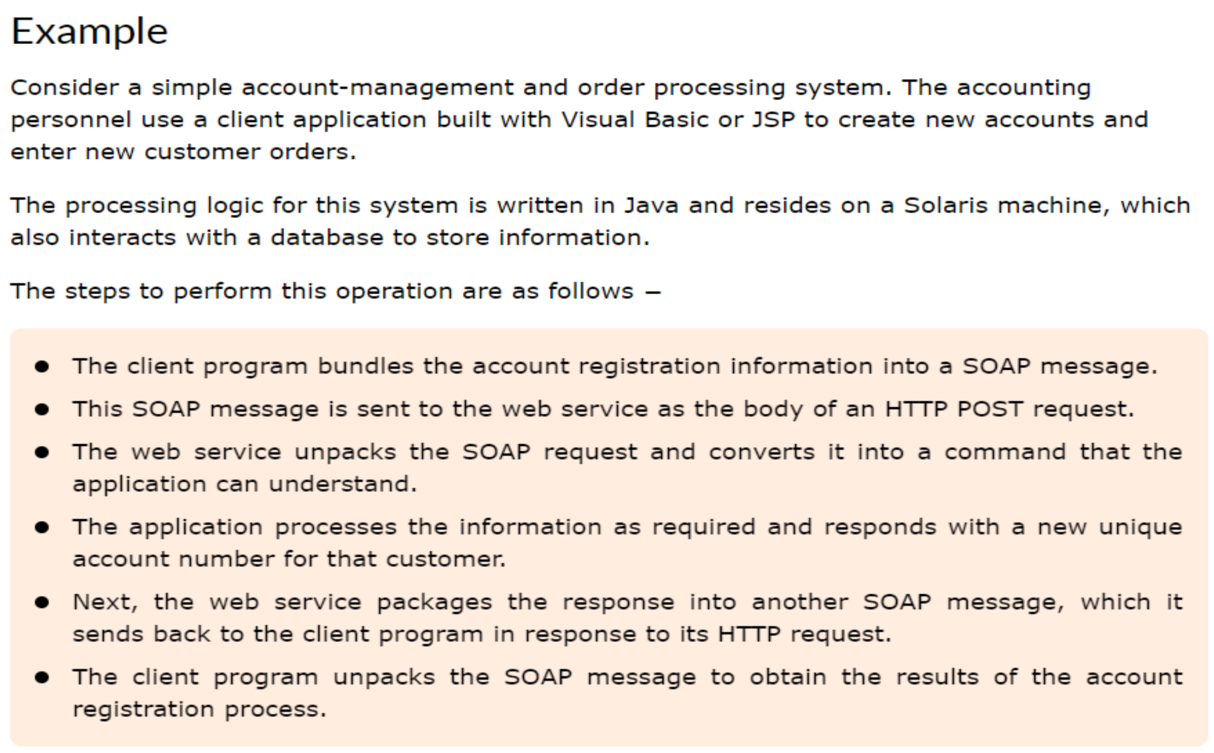
This is the provider of the web service. The service provider implements the service and makes it available on the Internet.

Service Requestor

This is any consumer of the web service. The requestor utilizes an existing web service by opening a network connection and sending an XML request.

Service Registry

This is a logically centralized directory of services. The registry provides a central place where developers can publish new services or find existing ones. It therefore serves as a centralized clearing house for companies and their services.



Example

Consider a simple account-management and order processing system. The accounting personnel use a client application built with Visual Basic or JSP to create new accounts and enter new customer orders.

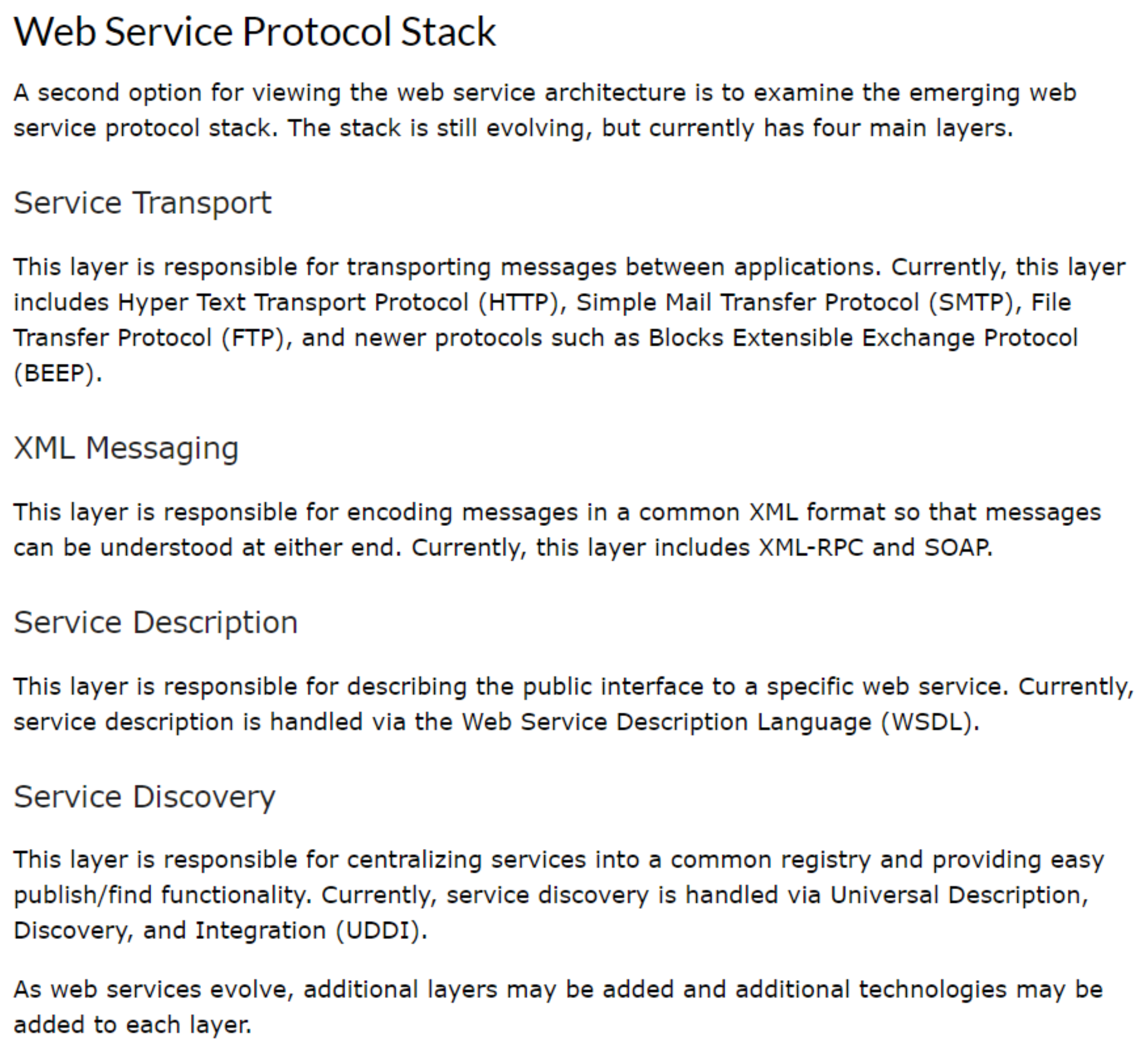
The processing logic for this system is written in Java and resides on a Solaris machine, which also interacts with a database to store information.

The steps to perform this operation are as follows –

- The client program bundles the account registration information into a SOAP message.
- This SOAP message is sent to the web service as the body of an HTTP POST request.
- The web service unpacks the SOAP request and converts it into a command that the application can understand.
- The application processes the information as required and responds with a new unique account number for that customer.
- Next, the web service packages the response into another SOAP message, which it sends back to the client program in response to its HTTP request.
- The client program unpacks the SOAP message to obtain the results of the account registration process.

The second option for handling a web service:

Steps (1, 2 and 3)



Web Service Protocol Stack

A second option for viewing the web service architecture is to examine the emerging web service protocol stack. The stack is still evolving, but currently has four main layers.

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), and newer protocols such as Blocks Extensible Exchange Protocol (BEEP).

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).

As web services evolve, additional layers may be added and additional technologies may be added to each layer.

Over the past few years, three primary technologies have emerged as worldwide standards that make up the core of today's web services technology. These technologies are discussed below.

XML-RPC

This is the simplest XML-based protocol for exchanging information between computers.

- XML-RPC is a simple protocol that uses XML messages to perform RPCs.
- Requests are encoded in XML and sent via HTTP POST.
- XML responses are embedded in the body of the HTTP response.
- XML-RPC is platform-independent.
- XML-RPC allows diverse applications to communicate.
- A Java client can speak XML-RPC to a Perl server.
- XML-RPC is the easiest way to get started with web services.

SOAP

SOAP is an XML-based protocol for exchanging information between computers.

- 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 simple and extensible.
- SOAP allows you to get around firewalls.
- SOAP will be developed as a W3C standard.

WSDL

WSDL is an XML-based language for describing web services and how to access them.

- WSDL stands for Web Services Description Language.
- WSDL was developed jointly by Microsoft and IBM.
- WSDL is an XML based protocol for information exchange in decentralized and distributed environments.
- WSDL is the standard format for describing a web service.
- WSDL definition describes how to access a web service and what operations it will perform.
- WSDL is a language for describing how to interface with XML-based services.
- WSDL is an integral part of UDDI, an XML-based worldwide business registry.
- WSDL is the language that UDDI uses.
- WSDL is pronounced as 'wiz-dull' and spelled out as 'W-S-D-L'.

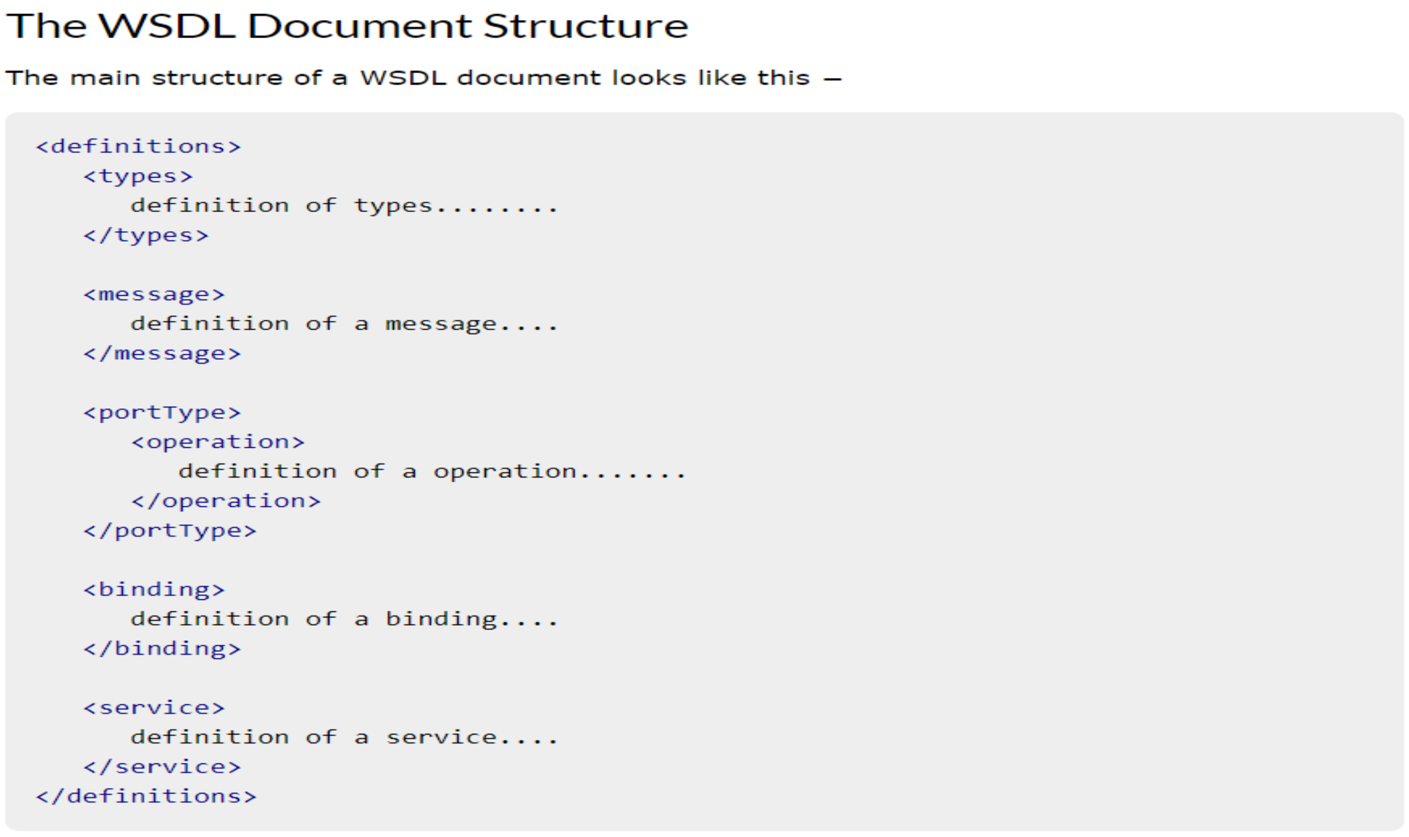
UDDI

UDDI is an XML-based standard for describing, publishing, and finding web services.

- UDDI stands for Universal Description, Discovery, and Integration.
- UDDI is a specification for a distributed registry of web services.
- UDDI is platform independent, open framework.
- UDDI can communicate via SOAP, CORBA, and Java RMI Protocol.
- UDDI uses WSDL to describe interfaces to web services.
- UDDI is seen with SOAP and WSDL as one of the three foundation standards of web services.
- UDDI is an open industry initiative enabling businesses to discover each other and define how they interact over the Internet.

Security is critical to web services. However, neither XML-RPC nor SOAP specifications make any explicit security or authentication requirements.

1- WSDL:



The WSDL Document Structure

The main structure of a WSDL document looks like this —

```
<definitions>
   <types>
      definition of types........
   </types>

   <message>
      definition of a message....
   </message>

   <portType>
      <operation>
         definition of a operation.......
      </operation>
   </portType>

   <binding>
      definition of a binding....
   </binding>

   <service>
      definition of a service....
   </service>
</definitions>
```

Example

Contents of HelloService.wsdl file –

```
<definitions name = "HelloService"
   targetNamespace = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns = "http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns = "http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd = "http://www.w3.org/2001/XMLSchema">

   <message name = "SayHelloRequest">
      <part name = "firstName" type = "xsd:string"/>
   </message>

   <message name = "SayHelloResponse">
      <part name = "greeting" type = "xsd:string"/>
   </message>

   <portType name = "Hello_PortType">
      <operation name = "sayHello">
         <input message = "tns:SayHelloRequest"/>
         <output message = "tns:SayHelloResponse"/>
      </operation>
   </portType>
```

```xml
   <binding name = "Hello_Binding" type = "tns:Hello_PortType">
      <soap:binding style = "rpc"
         transport = "http://schemas.xmlsoap.org/soap/http"/>
      <operation name = "sayHello">
         <soap:operation soapAction = "sayHello"/>
         <input>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </input>

         <output>
            <soap:body
               encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/"
               namespace = "urn:examples:helloservice"
               use = "encoded"/>
         </output>
      </operation>
   </binding>

   <service name = "Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding = "tns:Hello_Binding" name = "Hello_Port">
         <soap:address
            location = "http://www.examples.com/SayHello/" />
      </port>
   </service>
</definitions>
```

Example Analysis

- **Definitions –** HelloService
- **Type –** Using built-in data types and they are defined in XMLSchema.
- **Message –**
  - sayHelloRequest – firstName parameter
  - sayHelloresponse – greeting return value
- **Port Type –** sayHello operation that consists of a request and a response service.
- **Binding –** Direction to use the SOAP HTTP transport protocol.
- **Service –** Service available at http://www.examples.com/SayHello/
- **Port –** Associates the binding with the URI http://www.examples.com/SayHello/ where the running service can be accessed.

2- UDDI:

UDDI includes an XML Schema that describes the following data structures −

- businessEntity
- businessService
- bindingTemplate
- tModel
- publisherAssertion

Here is an example of a fictitious business's UDDI registry entry –

```
<businessEntity businessKey = "uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40"
   operator = "http://www.ibm.com" authorizedName = "John Doe">
   <name>Acme Company</name>
   <description>
      We create cool Web services
   </description>

   <contacts>
      <contact useType = "general info">
         <description>General Information</description>
         <personName>John Doe</personName>
         <phone>(123) 123-1234</phone>
         <email>jdoe@acme.com</email>
      </contact>
   </contacts>

   <businessServices>
      ...
   </businessServices>
   <identifierBag>
      <keyedReference tModelKey = "UUID:8609C81E-EE1F-4D5A-B202-3EB13AD01823"
         name = "D-U-N-S" value = "123456789" />
   </identifierBag>

   <categoryBag>
      <keyedReference tModelKey = "UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2"
         name = "NAICS" value = "111336" />
   </categoryBag>
</businessEntity>
```

Here is an example of a business service structure for the Hello World web service.

```xml
<businessService serviceKey = "uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
   businessKey = "uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
   <name>Hello World Web Service</name>
   <description>A friendly Web service</description>
   <bindingTemplates>
      ...
   </bindingTemplates>
   <categoryBag />
</businessService>
```

Here is an example of a binding template for Hello World.

```xml
<bindingTemplate serviceKey = "uuid:D6F1B765-BDB3-4837-828D-8284301E5A2A"
   bindingKey = "uuid:C0E6D5A8-C446-4f01-99DA-70E212685A40">
   <description>Hello World SOAP Binding</description>
   <accessPoint URLType = "http">http://localhost:8080</accessPoint>

   <tModelInstanceDetails>
      <tModelInstanceInfo tModelKey = "uuid:EB1B645F-CF2F-491f-811A-4868705F5904">
         <instanceDetails>
            <overviewDoc>
               <description>
                  references the description of the WSDL service definition
               </description>

               <overviewURL>
                  http://localhost/helloworld.wsdl
               </overviewURL>
            </overviewDoc>
         </instanceDetails>
      </tModelInstanceInfo>
   </tModelInstanceDetails>
</bindingTemplate>
```

Here is an example of a tModel representing the Hello World Interface port type.

```
<tModel tModelKey = "uuid:xyz987..." operator = "http://www.ibm.com"
   authorizedName = "John Doe">
   <name>HelloWorldInterface Port Type</name>
   <description>
      An interface for a friendly Web service
   </description>

   <overviewDoc>
      <overviewURL>
         http://localhost/helloworld.wsdl
      </overviewURL>
   </overviewDoc>
</tModel>
```

The keyedReference designates the asserted relationship type in terms of a keyName keyValue pair within a tModel, uniquely referenced by a tModelKey.

```
<element name = "publisherAssertion" type = "uddi:publisherAssertion" />
<complexType name = "publisherAssertion">
   <sequence>
      <element ref = "uddi:fromKey" />
      <element ref = "uddi:toKey" />
      <element ref = "uddi:keyedReference" />
   </sequence>
</complexType>
```

SOAP Message Structure

The following block depicts the general structure of a SOAP message –

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

   <SOAP-ENV:Header>
      ...
      ...
   </SOAP-ENV:Header>
   <SOAP-ENV:Body>
      ...
      ...
      <SOAP-ENV:Fault>
         ...
         ...
      </SOAP-ENV:Fault>
      ...
   </SOAP-ENV:Body>
</SOAP_ENV:Envelope>
```

SOAP Example:

Here is the SOAP request –

```
POST /Quotation HTTP/1.0
Host: www.xyz.org
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

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

   <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotations">
      <m:GetQuotation>
         <m:QuotationsName>MiscroSoft</m:QuotationsName>
      </m:GetQuotation>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

A corresponding SOAP response looks like –

```
HTTP/1.0 200 OK
Content-Type: text/xml; charset = utf-8
Content-Length: nnn

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

   <SOAP-ENV:Body xmlns:m = "http://www.xyz.org/quotation">
      <m:GetQuotationResponse>
         <m:Quotation>Here is the quotation</m:Quotation>
      </m:GetQuotationResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```